Knowledge-Based Systems 223 (2021) 107058

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Comparison of neutrosophic approach to various deep learning models for sentiment analysis

Mayukh Sharma, Ilanthenral Kandasamy *, W.B. Vasantha

School of Computer Science and Engineering, VIT, Vellore, Tamil Nadu, 632014, India

ARTICLE INFO

Article history:
Received 12 December 2020
Received in revised form 4 March 2021
Accepted 17 April 2021
Available online 23 April 2021

Keywords:
Neutrosophy
Sentiment analysis
BiLSTM
ALBERT
RoBERTa
BERT
MPNet
Stacked ensemble

ABSTRACT

Deep learning has been widely used in numerous real-world engineering applications and for classification problems. Real-world data is present with neutrality and indeterminacy, which neutrosophic theory captures clearly. Though both are currently developing research areas, there has been little study on their interlinking. We have proposed a novel framework to implement neutrosophy in deep learning models. Instead of just predicting a single class as output, we have quantified the sentiments using three membership functions to understand them better. Our proposed model consists of two blocks, feature extraction, and feature classification. Having a separate feature extraction block enables us to use any model as a feature extractor. We experimented with BiLSTM using GloVe (Global Vectors for word representation), BERT (Bidirectional Encoder Representations from Transformers), ALBERT (A Lite BERT), RoBERTa (Robustly optimized BERT approach), MPNet, and stacked ensemble models. Feature classification performs prediction and dimensionality reduction of features. Experimental analysis was done on the SemEval 2017 Task 4 dataset (Subtask A). We used the intermediate layer features to define membership functions of Single Valued Neutrosophic Sets (SVNS). We used these membership functions for prediction as well. We have compared our models with the top five teams of the task and recent state-of-the-art systems. Our proposed stacked ensemble model achieved the best recall (0.733) score.

1. Introduction

Neural networks that had piped much interest in artificial intelligence reached a stagnation stage in the 1990s because they were mostly shallow networks due to lesser computation powers. With the advent of better hardware like GPU, internet, and the massive amount of data, neural networks in the form of deep learning (neural networks with more than three layers) have taken off credibly. Deep learning uses cascading different layers of processing units for transformation and feature extraction, which are non-linear. Layers near the input layer learn simple basic features whereas, the higher layers learn more complicated features from the previous layers. Deep learning has produced fantastic results in many engineering and application-oriented scientific problems from image processing to Natural Language Processing (NLP). In recent years, deep learning has been used extensively in sentiment analysis, NLP, and Natural Language Generation(NLG).

Sentiment analysis analyzes the polarity of the sentiment behind a sentence or a social media post. It helps to understand social media on specific subjects. The existing conventional sentiment analysis tools that work on various platforms aim at categorizing the target subject into negative or positive polarity. Ascertaining a tweet's polarity is an essential exercise in NLP and data science. It is widely used in several practical applications ranging from scrutinizing popular events to mining trading indicators by scrutinizing tweets about companies/corporates. These systems learn to utilize syntactic and semantic features and their impact on the sentiment of natural language. Work done in [1] discusses the applications of sentiment analysis in the real world and discusses knowledge-based, statistical, and hybrid techniques for sentiment analysis and affective computing. For the past ten years, SemEval (Semantic Evaluation) competitions/workshops are organized for various tasks to evaluate computational semantic analysis systems. The SemEval-2017: Task 4 came with a similar challenge of sentiment analysis on tweets. An important aspect of this task involved classifying the sentiment into the positive, negative, or neutral class rather than just finding the positive or negative polarity. For an exhaustive report on SemEval-2017 Task 4, see [2]. Recently, deep learning methods have drastically surpassed conventional techniques in numerous

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail addresses: mayukh.sharma2016@vitalum.ac.in (M. Sharma), ilanthenral.k@vit.ac.in (I. Kandasamy), vasantha.wb@vit.ac.in (W.B. Vasantha).

https://doi.org/10.1016/j.knosys.2021.107058
0950-7051/

NLP tasks [3–6], and opinion mining is no exemption [7,8], and they have been used extensively in SemEval tasks.

The neutrosophy concept was proposed by Smarandache [9], which was shortly transformed into SVNS by Wang et al. [10]. SVNS is a generalization of fuzzy sets [11]. Neutrosophic refined sets were introduced by Smarandache [12] and have been applied to sentiment analysis in [13,14]. The existing conventional sentiment analysis or fuzzy sentiment analysis does not capture the indeterminacy present in the tweets/opinion.

In the social media context, the concept of neutrosophy is a formidable tool for sentiment analysis. Opinions in social media are based on a user's perception of the topic. He/She can take an assortment of attitudes on an issue that can vary from strongly positive to strongly negative. Human sentiment is a complex combination of emotions. Current sentiment analysis systems try to predict the most powerful one, completely ignoring other sentiments. SVNS overcomes this by assigning membership functions to each sentiment. The final sentiment is represented as a set of each sentiment corresponding to its independent membership function. Moreover, the user may not have a strong opinion, and the sentiment falls in the indeterminate/neutral category. Neutrosophy bases its estimates on these indeterminacies of a sentiment present in a sentence.

Two of the popular theory in artificial intelligence and soft computing are deep learning techniques and neutrosophy. Consequently, to build a state-of-the-art sentiment classifier, we integrated these techniques to build a system that combines both. We have proposed a deep learning system that predicts the sentiment of a sentence as an SVNS, that is in terms of three membership values. Our proposed architecture makes use of feature extraction and feature classification components. Feature extraction acts as a black-box for generating features from the text allowing us to experiment with different models while using the same architecture for generating SVNS values. The proposed model's high-level architecture is given in Fig. 1.

One of our proposed models makes use of Recurrent Neural Networks (RNN) based on Long Short Term Memory (LSTM) and Gated Recurrent Units (GRU) on pre-trained GloVe embeddings for learning the class features. We used optimization techniques like Batch Normalization (BatchNorm) and Dropout for performance enhancement. We also experimented with BERT (Bidirectional Encoder Representations from Transformers), ALBERT (A Lite BERT), RoBERTa (Robustly optimized BERT approach), MPNet, and stacked ensemble models. K-means clustering on features learned by intermediate layers of our neural network was used for calculating the SVNS values.

The paper organization is as follows: Section 1 is introductory in nature. Section 2 justifies the use of neutrosophy along with deep learning for sentiment analysis. In Section 3, a brief overview of neutrosophy, deep learning models, and SemEval 2017 Task 4 is presented. Section 4 provides the dataset description. Then, Section 5 describes the various model architectures in detail along with the neutrosophic approach. Section 6 explains the experimental setup. Section 7 provides the experimental results of the six models (BiLSTM with GloVe, BERT, ALBERT, RoBERTa, MPNet, and Stacked Ensemble) along with the error analysis. Section 8 compares the model with the first five teams of the SemEval 2017 Task 4 (Subtask A), current state-of-the-art systems, and it outperforms all competing teams and models. Finally, Section 9 concludes the research study.

2. Why use neutrosophy with deep learning for sentiment analysis

1. Deep learning models have turned out to be state-of-the-art for solving problems related to NLP. They perform classification tasks efficiently but are incapable of quantifying the input over the output classes. Multi-label classification problems use softmax as the final activation. Softmax ensures that the sum of the probabilities for output classes is 1. Multi-label classification problems use cross-entropy loss which is, defined as:

$$-\sum_{i}^{C} t_i log(s_i) \tag{1}$$

where t_i is the ground truth, s_i is predicted value and C represents the number of classes. It alters the loss value only with respect to the truth value of the current sample, trying to maximize its probability against other classes for which the *cross entropy* value will be zero. The problem with *cross-entropy* for sentiment analysis is that it completely ignores other sentiments in natural language. Neutral sentiment may be a partial combination of both positive and negative sentiments. The neural network will learn to identify the neutral component and maximize its probability but at the cost of losing information about other sentiments. Neutrosophy on the other hand deals with neutralities. SVNS can be used to define the membership function for each class. So instead of simply predicting the correct class, we can quantify the sample into SVNS values. This allows us to capture all the sentiments in the final output. This can help in improving the understanding of language using neural networks.
2. Understanding language and emotion is a difficult task. Especially on social media platforms where people use varied emotions and sarcasm. RNN's have shown success in language modeling tasks such as sentiment analysis, machine translation, and classification. However, they fail to understand the combination of emotions and simply assign the polarity with maximum probability. However, language is never a single emotion. It is a combination of emotions that the person uses to present his belief. Thus, instead of labeling a sentence as positive, negative, or neutral, we can use SVNS to represent them as a combination of sentiments using membership functions. So in our approach, we used state-of-the-art machine learning algorithms and combined them with neutrosophy to achieve the results.
3. A survey on machine learning in neutrosophic environments done in [15] covers the machine learning algorithms used to date with neutrosophy. For classification tasks, models were trained using support vector machines (SVM). Although SVM's [16] attain good results, they fail to capture the sequential nature of the text. We employed RNN's with attention which are best known for such tasks. We also performed experiments with pre-trained language models based on transformer architecture [17] which attain the current state-of-the-art results. Neural networks [18] have been employed earlier for predicting the membership functions given the input data. They used separate neural networks for each membership function which makes the task computationally expensive. Our proposed method employs neural networks to learn feature vectors of different dimensions and use clustering on them for calculating the SVNS values. We will describe the approach in more detail in the following sections.
4. As we all know, big data has become an important part of machine learning systems. Moreover, machine learning algorithms are highly dependent on large datasets for better performance. Deep learning systems are developed keeping this in mind. Libraries like TensorFlow[1] and PyTorch[2] allow us to train on large datasets. These libraries

[1] https://www.tensorflow.org.
[2] https://pytorch.org.

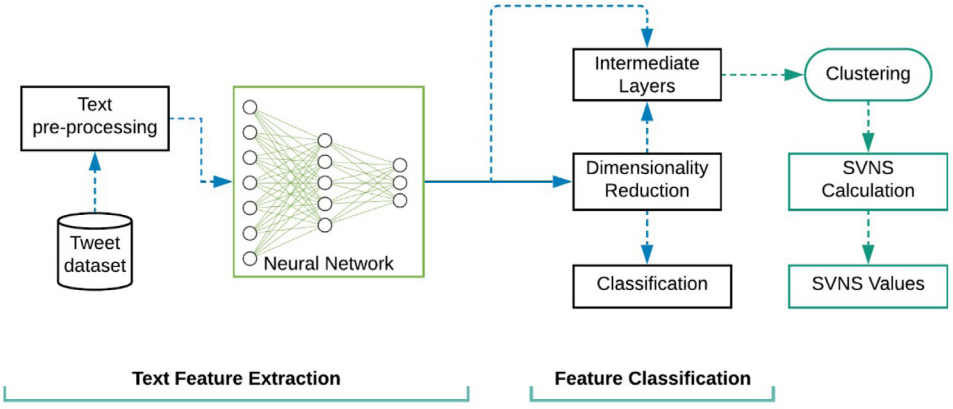


Fig. 1. High-level architecture of our system.

are optimized for speeding performance on GPUs which is not the case with traditional machine learning systems. Another important aspect of using deep learning systems is transfer learning. They allow us to use features learned on massive datasets containing millions of samples over weeks of training. We can use these learned features in our problem helping in saving time as well as computational effort.

3. Basic concepts/overview

3.1. Neutrosophy

The concept of neutrosophy and related definitions are rewritten here in relation to opinion mining. Neutrosophy analyzes an opinion "O" and associates membership values to it in terms of positive, neutral, and negative memberships.

Definition 1 ([10]). Let X be a space of opinions, concepts or points with elements in X denoted by x. A SVNS R in X is characterized by three membership functions namely negative or false $N_R(x)$ membership function, neutral or indeterminacy $I_R(x)$ membership function, and positive or truth $P_R(x)$ membership function. $\forall x \in X$, where $N_R(x), I_R(x), P_R(x) \in [0, 1]$, and $0 \leq N_R(x) + I_R(x) + P_R(x) \leq 3$. Therefore, an SVNS of an opinion O can be represented by

$$R = \{\langle x, N_R(x), I_R(x), P_R(x)\rangle \mid x \in X\}. \tag{2}$$

The distance measures over two opinions represented by R and Q in a universe of discourse, $X = x_1, x_2, \ldots, x_n$, is given in the following:

R and Q are denoted by

$$R = \{\langle x_i, N_R(x_i), I_R(x_i), P_R(x_i)\rangle \mid x_i \in X\}, \text{ and}$$
$$Q = \{\langle x_i, N_Q(x_i), I_Q(x_i), P_Q(x_i)\rangle \mid x_i \in X\},$$

where $N_R(x_i), I_R(x_i), P_R(x_i), N_Q(x_i), I_Q(x_i), P_Q(x_i) \in [0, 1]; \forall x_i \in X$. Let w_i be the weight of an element x_i with $w_i \geq 0 (i = 1, 2, \ldots, n)$ and $\sum_{i=1}^{n} w_i = 1$. Then, the neutrosophic weighted distance of R and Q is defined as follows:

$$d_\lambda(R, Q) = \{\frac{1}{3}\sum_{i=1}^{n} w_i[\mid N_R(x_i) - N_Q(x_i) \mid^\lambda + \mid I_R(x_i) - I_Q(x_i) \mid^\lambda + \\ + \mid P_R(x_i) - P_Q(x_i) \mid^\lambda]\}^{1/\lambda} \tag{3}$$

where $\lambda > 0$.

The distance measures over two SVNSs are discussed in detail in [19,20]. They also define several similarity measures between them and investigate their properties [19,21,22]. To make a more refined and accurate representation of the indeterminacy present in the real-world data, refined neutrosophic sets [23] were defined. Double Valued Neutrosophic Sets (DVNS) was defined [24,25] with two indeterminate memberships. Triple Refined Indeterminate Neutrosophic Sets (TRINS) have been used for personality classification and Likert Scaling [26]. Clustering of neutrosophic sets have been carried out in [13,19,24,27], using k-means clustering algorithm or MSTs.

Refined neutrosophic sets [13,14,24,26–29] have been used in sentiment analysis. Recently Multi Refined Neutrosophic Sets (MRNS) have been used for sentiment analysis [14], they do not represent each tweet as a neutrosophic element. Instead, they study the indeterminacy present in the overall opinion on a seven-point scale. Individual analysis of each tweet is not carried out and importance is not given to the indeterminacy present in each tweet.

A remarkable application of SVNS to sentiment analysis is where each tweet is analyzed separately and represented as SVNS with negative, positive, and indeterminate memberships [13], and they did not use deep learning while analyzing the #MeToo movement tweets. The imaginative play in children was studied using single-valued refined neutrosophic sets in [27]. In [30], a new sentiment similarity measure was proposed between pairs of words, which are considered as SVNS. In [31], "Senti-NSetPSO", a hybrid framework is proposed to analyze large-sized text; it is comprised of two classifiers: binary and ternary based on hybridization of Particle Swarm Optimization (PSO) with a neutrosophic set.

3.2. Pre-trained embeddings (transfer learning)

Embeddings refer to the representation of words in an n-dimensional vector space. These are capable of carrying complex syntactic and semantic information and encode many linguistic regularities and patterns [32]. There are many open-source word vector representations available like GloVe [33], word2Vec [34] and fastText [35]. In our work, we will be using GloVe to perform experiments on the test set [36]. It is an unsupervised learning algorithm developed by Stanford, for generating word embeddings by aggregating global word–word co-occurrence matrix from a corpus. The embeddings map words from vocabulary into dense vectors of fixed size (embedding vectors). These embeddings can be further trained, to better fit the neural network model. The advantage of using embeddings is their linear sub-structure that is similar words will have similar euclidean distances in their

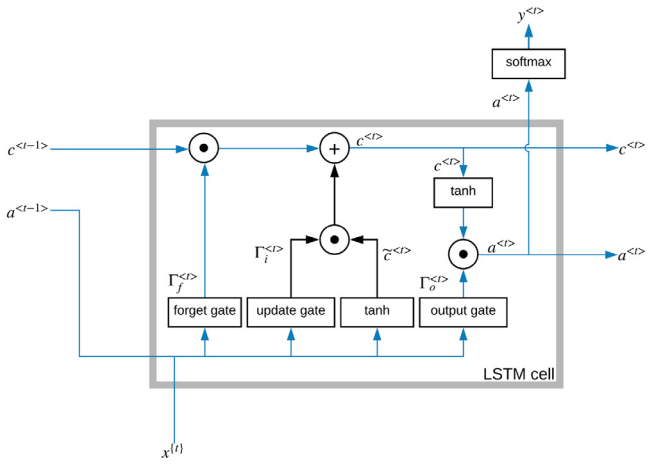


Fig. 2. LSTM Cell.

respective vector spaces. The other benefit is they save a lot of computational cost and training time by providing pre-trained features for several NLP tasks.

3.3. LSTM and GRU

LSTM [37] and GRU [38] are an improved form of RNN that overcome the problem of vanishing gradients that traditional RNN's suffer. The gradients shrink as the RNN back propagates through time. Sometimes the gradient values may become small, and they do not contribute to learning. Thus, the layers getting small gradients eventually stop learning. Those are usually the earlier layers. Due to this, it becomes difficult to carry information from earlier timestamps to later ones [39]. However, LSTM's overcome this problem. They have cell states and various gates. The cell state helps transport relative information down the sequences. It can be considered as the memory of the network. As the cell state moves down the sequence, information gets added or removed as decided by the gates of the cell. The combined effect of the memory cells aided by the gates helps the LSTM and GRU to learn long-term dependencies.

Each of the gates described in Fig. 2 is governed by the equations given below.

$$\Gamma_f^{<t>} = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \tag{4}$$

$$\Gamma_u^{<t>} = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \tag{5}$$

$$\tilde{c}^{\{t\}} = tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c) \tag{6}$$

$$c^{<t>} = \Gamma_f^{<t>} \circ c^{<t-1>} + \Gamma_u^{<t>} \circ \tilde{c}^{<t>} \tag{7}$$

$$\Gamma_o^{<t>} = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \tag{8}$$

$$a^{<t>} = \Gamma_o^{<t>} \circ tanh(c^t) \tag{9}$$

Each gate in the LSTM is nothing but a single-layered small neural network with a bias term. However, when combined, they become an immensely powerful unit. The cell state $c^{<t>}$ acts as a highway and helps to carry long-term dependencies to later timestamps. An LSTM is nothing but a combination of tiny neural networks using the same weights for each timestamp and overcoming the problem of vanishing gradients using various gates as mentioned in Fig. 2.

3.4. Attention

Given the task of translating a book from English to Hindi, we will not read the entire book in English and then translate it to Hindi. The general approach would be to translate at the sentence level and focus on the parts of the English sentence corresponding to the parts of the Hindi sentence under translation. Similarly, the attention mechanism helps the neural network identify where it needs to focus for a given timestamp. Attention [4] is essentially a vector representation for a given word/token with respect to all words/tokens in a sentence. It is calculated for each timestamp over a sequence and captures the context of the current timestamp with respect to the sequence. The output of attention is called context vectors.

Let us take a sequence of length N where t represents a timestamp of the sequence. Context vector c_t for timestamp t is calculated as:

$$h_{t,n} = \tanh(x_t, W_a + x_n W_b) \tag{10}$$

$$e_{t,n} = \sigma\left(h_{t,n} W_c\right) \tag{11}$$

$$a_t = softmax\left(e_{t,1}, e_{t,2}, \ldots, e_{t,N}\right) \tag{12}$$

$$c_t = \sum_n a_{t,n} x_n \tag{13}$$

for $n = 1$ to N. W represents the weight vectors. x_t represents the feature vector from timestamp t. σ represents the sigmoid function. Fig. 3 helps us visualize the attention mechanism. We used additive attention. The first step creates a joint representation by combining features at timestamp t with respect to timestamps n of our sequence. The joint features are then passed through a dense layer with weights W_c and sigmoid activation. Next, we convert joint features into a probability distribution over the sequence N by passing through a softmax layer. These are the attention weights for timestamp t represented by a_t. Finally, we find the context vectors as a sum of the linear combination of attention weights over features vectors of the sequence N.

3.5. Scaled dot product attention

It was proposed first for use in transformers [17] and is based on self-attention. They defined the attention function as a mapping of query and key–value pair to an output function. Queries, keys, and values are all vectors. The output function is a weighted sum of values where the weight assigned to each value is computed using a compatibility function of query and its corresponding key. Queries, keys, and values are computed from embeddings of inputs using independent weight matrices for queries (W_q), keys (W_k), and values (W_v). Keys and queries have a dimension d_k. Dimension of values is d_v. The dot product of queries and keys is taken, divided by $\sqrt{d_k}$ and finally passed through softmax to calculate attention weights. The weighted sum of values and attention weights finally gives the context vector for a particular timestamp t of the sequence N. In practice attention function for all queries is calculated simultaneously using matrix operations. Queries, Keys, and values are packed into matrices Q, K, V. Context vector for entire sequence N is computed as :

$$Attention(Q, K, V) = softmax(QK^T/\sqrt{d_k})V$$

where dimension of Q and K is (N, d_k), dimension of V is (N, d_v). The output of attention step has dimension of (N, d_v) which represent the context vectors.

3.6. Multi-head attention

Instead of computing a single attention function, it was found, using multiple attention functions yields better results. Queries, keys, and values are linearly projected h times with different learned projection functions to d_k, d_k, and d_v dimensional features. Attention function is then performed parallelly for each set

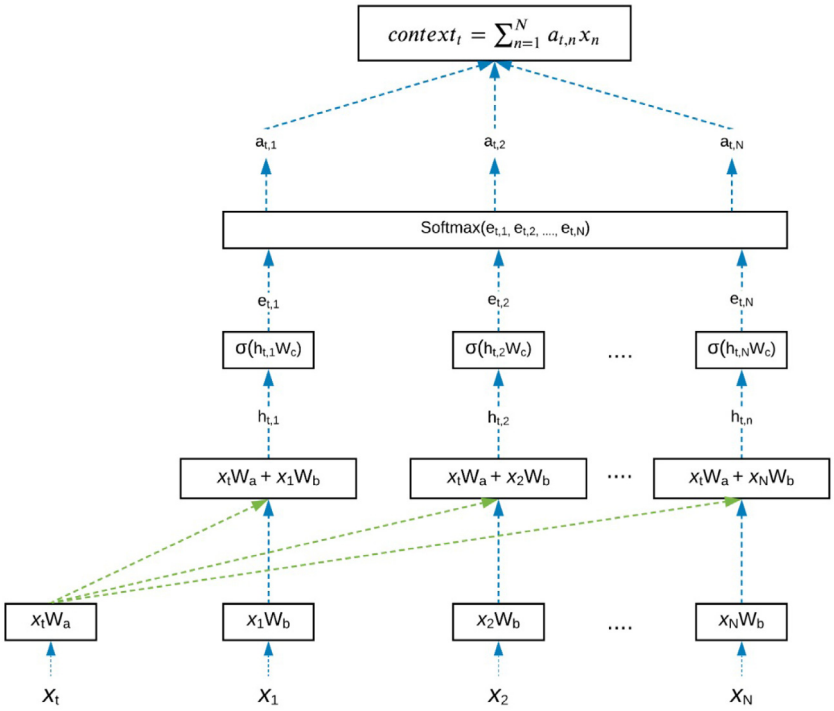


Fig. 3. Additive self-attention.

of projected features to yield d_v dimensional features. These features are then concatenated and once again projected, resulting in final values. This allows the model to jointly attend to different subspaces of information at different positions.

$$MultiHead(Q, K, V) = Concat(head_1, head_2, \ldots, head_h)W^o$$

where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$, where the projections are parameter matrices W_i^Q, W_i^K having dimension (d_{model}, d_k) and W_i^V having dimension (d_{model}, d_v) where d_{model} is dimension of query, key and values.

3.7. Transformers

Transformers were proposed in [4] and have become the most popular choice for solving NLP tasks. LSTM's, due to their sequential nature are dependent on the output of the previous timestamp for future computations. This helps them learn long-range dependencies. However, it also introduces difficulty to perform parallel computations on them using modern GPUs. Transformers solved this problem by removing sequential nature and relying entirely on attention to understand dependencies in a sequence. They use dot product and multi-head attention which forms the core of transformer models. They make use of standard encoder–decoder architecture using multi-head attention to learn dependencies instead of using RNN. This allows both the encoder and decoder to compute features independently and in parallel. This makes them highly efficient for running on distributed systems. Positional embeddings are used to retain position-based information. Detailed architecture can be found at [17].

3.8. Pre-trained language models (PLM)

NLP contains a variety of tasks ranging from sentiment analysis, named-entity recognition, machine translation, language modeling. Deep learning models have become a popular choice for solving these problems and have led to good results in several tasks. Training these models requires a large amount of data for supervised learning problems. Most datasets for NLP tasks contain only a few hundred or a thousand human-labeled samples. The solution to this problem came in form of a method known as pre-training which aims to learn general-purpose language representation by training on large amounts of an unlabeled text corpus. The majority of PLM's make use of transformer architecture. The PLM can then be fine-tuned on several downstream tasks and have shown excellent results. BERT [40], RoBERTa [41], ALBERT [42], MPNet [43] are examples of pre-trained language models.

3.9. Dropout

Ensemble models created by combining different models help to increase the efficiency of the network. In the real world, the idea of averaging outputs of different neural networks becomes computationally expensive. For producing good results, we need to train models with different architectures. At the same time, we need large datasets for training these models, which are not always available. Even if we are successful in training models with different architectures over different subsets of data, using them as a combination to predict output is infeasible at run-time where an immediate response may be required. Dropout solves both these problems. It enables us to combine exponentially many architectures and prevent over-fitting.

"Dropout" [44] means randomly dropping neurons during the training phase of the neural network. The neurons are dropped temporarily and include dropping all outgoing and incoming connections. The dropping is random and is guided by a probability parameter. Neurons are not dropped while predicting and the whole network is used during the prediction phase.

3.10. Batch normalization

The process of training neural networks involves changing the weights of our network to minimize the loss function. This causes a change in the distribution of the weight of each layer as input

parameters from the previous layer change. This slows down the training process as we need lower learning rates, use careful parameter initialization, and makes it difficult to train models with non-linearities.

This problem is called internal covariate shift and is addressed by normalizing the inputs for that particular layer. Batch Normalization [44] draws its strength from the fact that it makes normalization part of the model architecture and performs normalization for each mini-batch. Batch Normalization enables us to use much higher learning rates and be less careful about initialization. It tries to reduce the internal covariate shift and helps in accelerating the training speed of neural networks. It uses the normalization step to fix the variance and mean of layer inputs. It improves the gradient flow through the network as it reduces the dependence of gradients on the initial value of parameters or their scale. This allows us to use higher learning rates. Batch Normalization also enables us to use saturating non-linearities as it prevents the network from getting stuck at saturating nodes.

3.11. Feature extraction from intermediate layers

Dense Neural Networks (DNN) can learn powerful functions. They hierarchically learn features of given data, layer by layer. As we go deep into the neural network, they learn more complex features. Each neuron learns to identify a specific feature and is activated by it. This characteristic can be used to map input features into an n-dimensional vector space corresponding to the output classes from the deeper layers. A similar approach with Convolutional Neural Networks (CNN's) has been used in neural image transfer [45] for generating artistic images. They used the intermediate layers for defining the similarity cost function, and particularly good results were obtained. SenticNet 6 proposed in [46] makes use of contextual features learned by LSTM's and BERT and use them for automatic discovery of concept clusters that are semantically related. The encoder–decoder architecture-based systems also use a similar approach. Another example can be machine translation [47] where sentence features are condensed into a small vector space using an encoder which is then used by a decoder to output the sentence in another language. Another advantage of using such an approach is that we can get feature vectors of different dimensions from different depths of neural networks. We propose a similar approach for finding out the SVNS values using pre-final layers that learn a similar set of features for each class. Another reason for using pre-final layers instead of the final layer is the use of softmax activation. As discussed before, it maximizes the probability of one class and loses information about the remaining classes. Thus SVNS from this layer will not be effectively able to understand sentiment corresponding to each membership function. The results from intermediate layers showed that predictions using SVNS from intermediate layers performed equally well or better in most cases. Hence, the features are correctly learned by the intermediate layers.

3.12. SVNS calculation

SVNS values, as defined in [48] are a set of membership functions for an entity. A neutrosophic set $A \in X$ is characterized by a truth-membership function T_A, an indeterminacy membership function I_A and a falsity-membership function F_A. The range of each of these membership functions is (0,1). It represents the likeliness of element X belonging to each of the given classes and must be independent. For SVNS calculation we use the feature vectors extracted from the intermediate layers of our trained neural networks. These vectors provide a feature-rich numerical representation of tweet samples. Since these vectors are extracted from models trained over supervised data, samples belonging to the same classes will have similar features and vice-versa. We use this property to separate the features into clusters corresponding to their respective class. We used k-means clustering for this step. Once we obtain the clusters, we find the cluster centers by simply averaging the feature vectors for the given cluster. We define SVNS using these cluster centers and feature vectors. Let us define the SVNS values for each sample as ($SVNS_{positive}$, $SVNS_{negative}$, $SVNS_{neutral}$), cluster centers as ($C_{positive}$, $C_{negative}$, $C_{neutral}$) and feature vectors F of N samples then:

$$SVNS_{positive} = Cosine\_distance(C_{positive}, F_n)/2 \tag{14}$$

$$SVNS_{negative} = Cosine\_distance(C_{negative}, F_n)/2 \tag{15}$$

$$SVNS_{neutral} = Cosine\_distance(C_{neutral}, F_n)/2 \tag{16}$$

$$Cosine\_distance = 1 - Cosine\_similarity(C_x, F_n) \tag{17}$$

for $x \in$ (positive, negative, neutral) and $n \in N$, the number of samples.

4. Dataset description

Our work makes use of the SemEval 2017 Task 4(Subtask A) dataset. It involved classifying tweets into positive, neutral, and negative classes. It is one of the largest available datasets which contains the neutral class in annotations instead of just positive and negative classes. Since neutrosophy deals with neutralities, we found it suitable for carrying out experimental analysis for our proposed work. The statistics of the training and testing data are given in Table 1.

For the comparison with our model, the best five systems from the task are taken into consideration; their brief description is as follows.

Datastories: [36] This team used the LSTM model coupled with an attention mechanism for predicting the sentiment of tweets. GloVe embeddings were used for representing the words of tweets in vectorized form. They developed their tool ekphrasis for cleaning and pre-processing the tweets before training their model.

BBtwtr: [49] This team used an ensemble model of CNN and LSTM based model for the task. The CNN model [50] on the word vector representations of the tweets along with BiLSTM was used. They experimented on embeddings trained on Stanford's GloVe [33], Google's word2Vec [34], and Facebook's FastText [51].

LIA: [52] This team used a fusion technique for learning sentiments using different classifiers trained on different kinds of embeddings. The classifiers consisted of a combination of CNN-LSTM based neural networks. They used four different kinds of embeddings and fused the output of each classifier which is used for final prediction using another neural network.

Senti17: [53] This team used a unique approach based on voting. They used ten CNN models. Each model was initialized with the same embeddings for tweets but different initializations for weights were used. The predictions are then counted for each label. The final output was the label with maximum votes.

NNEMBs: [54] This team made use of the RCNN [55] based model for sentence polarity detection. They used an ensemble of these RCNN based models on different word embeddings. Different embeddings trained using different methods and they can carry different semantics. They used this fact to build an ensemble model on them and use it for final predictions.

5. Model description

5.1. Text preprocessing and cleaning

Data preprocessing plays a pivotal role in creating an efficient machine learning model. Data from social media platforms contain a lot of noise, making it difficult to understand its underlying

Table 1
Dataset statistics.

Dataset	Labels			Total
	Neutral	*Positive*	*Negative*	
Train	22 524	19 799	7809	50 132
Test	5937	3972	2375	12 284

Table 2
Common chat-words and their meanings.

Chat-word	Meaning
LOL	Laugh out loud
BTW	By the way
FYI	For your information
U	You

Table 3
Normalization of common words.

Un-normalized word	Normalized word
www.linkedin.com	< URL >
@stark	< USER >
4	< NUMBER >
Jun 29	< DATE >

meaning. The dataset we used was extracted from Twitter and contained hashtags, emojis, emoticons, web-URLs. We process the data so that useful information can be extracted from them. We describe the data cleaning process as follows:

Removing website names: Tweets sometimes contain a small description which is followed by a URL to the actual source. These URLs do not carry any semantic information about the tweet. Hence, one approach can be removing the URLs from the tweet. Another option is to replace all URLs with a common word similar to "URL". We use the latter in our proposed data-cleaning pipeline.

Chat word conversion: (e.g. LOL — laugh out loud). The use of slang words is common on Twitter. These words are not part of standard vocabulary, but when expanded, provide useful insights about the sentiment present in the tweet. We, therefore, use a dictionary of common slang words for converting them into their actual meaning. Some common slang words and their meanings are given in Table 2.

Replacing emoticons and emojis with their meaning: Using emojis and emoticons to represent sentiment is common on social media platforms. They can be of significant importance while trying to identify the sentiment of the tweet. To make full use of information provided by emojis and emoticons we used the python package emoji[3] for converting emojis into their textual meaning. For emoticons, we used a dictionary mapping the emoticons to their respective meaning.

Ekphrasis library: For the next step, we use the python Ekphrasis[4] [36] library. It normalizes the URLs, money, date, and numbers so that a single representation of them can be used across the dataset. Some examples of normalization are given in Table 3. It annotates 'hashtag', 'allcaps', 'elongated', 'repeated', 'emphasis', 'censored'. The pre-trained embeddings usually contain vector representations for these annotated words, and thus, they can help in better understanding the sentence within these annotations. e.g., STRONGLY DISAGREE will be converted to < allcaps > strongly disagree < allcaps >.

Hashtag segmentation and spelling correction Hashtags are used to represent the topic of the tweet by users. It may contain useful information about events and the associated sentiment. Sometimes different words are grouped to represent the topic. We also perform spelling correction. Tweets sometimes contain intentional spelling mistakes which may be understood by users but pose a challenge for language modeling tasks. We used the implementation of the Viterbi algorithm [56] and Peter Norvig's algorithm[5] based on Twitter corpus by Ekphrasis library for hashtag segmentation and spelling correction.

Out-of-vocabulary words: We skip the out-of-vocabulary words after the above pre-processing steps. Most of them were numbers, spelling mistakes, unsegmented hashtags having a low frequency of occurrence. Another option is to replace all out-of-vocabulary words with a special (< UNKNOWN >) token. But this makes every unknown word to be represented equally which may not be the case.

Tokenization for PLM's: Different PLM's have their own set of vocabulary which they use during the pre-training process. They use tokenizers to break the words into sub-words which makes up the vocabulary for PLM's. Some popular tokenization algorithms PLM's use are Byte-Pair Encoding (BPE) [57], WordPiece [58], and SentencePiece [59]. For fine-tuning these models on our dataset, we need to tokenize the tweets into PLM's vocabulary. For this step, we used Hugging Faces's implementation of FastTokenizers[6] for the respective models.

5.2. Model architecture

Our architecture is made up of two components i.e., text feature extraction and feature classification. The feature classification component is built on the features learned by the text feature extraction component. We use three different architectures to define the text feature extraction component. We used BiLSTM-GRU with GloVe embeddings, PLM's, and a stacked ensemble approach for our feature extraction component. The second part of our model, feature classification, is used for the prediction and dimensionality reduction of features. The outputs of the feature extraction component make the input for it. We used the output of the feature extraction component and intermediate layer features from the feature classification component for SVNS calculation. The advantage of separating the model into two components is that it provides us with the flexibility to use any model inside the feature extraction component. This is important as it makes our architecture compatible with different models. Once we choose an architecture the whole model is trained end to end just like any other machine learning model. It also provides scope for including future models making our architecture very robust. The system architecture of our model is given in Fig. 4.

Next, we define the architectures of the two components of our model in detail. Text feature extraction used a BiLSTM, a PLM model, and a stacked ensemble architecture. Text feature classification consists of a simple dense layer for dimensionality reduction and feature normalization.

5.2.1. Text feature extraction

1. **BiLSTM GRU:** The concept of Dropout is from [44], Batch Normalization is from [60] and L2 Regularization is from [61]. The model consists of a 100-dimensional embedding layer that converts the words into respective embedding vectors. Gaussian noise with $\sigma = 0.2$ and a Dropout of 0.3 were used for regularization in the embedding layer. Two BiLSTM layers of size 150 (300 for bi-directional) stacked above each other form the next layer. We use the bi-directional layer to capture the overall context of the

[3] https://github.com/carpedm20/emoji/.
[4] https://github.com/cbaziotis/ekphrasis.
[5] https://norvig.com/spell-correct.html.
[6] https://huggingface.co/transformers/tokenizer_summary.html.

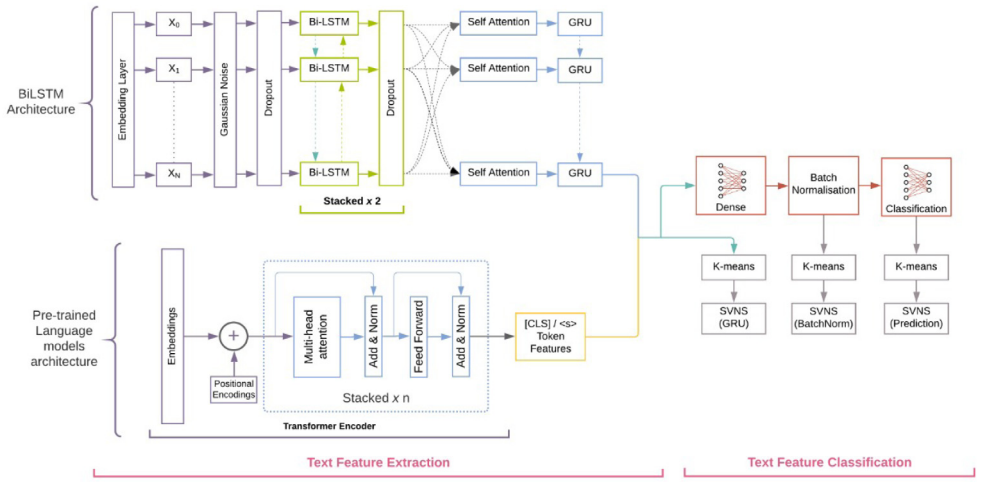


Fig. 4. Detailed overview of our system architecture. In text feature extraction we show BiLSTM and PLM-based architectures. Both were used independently and are shown together only for illustration. The PLM architecture shows the structure of a transformer which is the basic building block of PLM's.

sentence before we feed it to the attention layer. A Dropout of 0.2 is used between intermediate layers and 0.25 between recurrent connections of LSTM's. The output of the second BiLSTM is fed to the attention layer. The context vectors from the attention layer are used as GRU's input. The size of GRU is 64 units, with a Dropout of 0.2 and a recurrent Dropout of 0.25. It encodes the information from the attention layers. Only the output of the last timestamp of GRU is used which is the output of our BiLSTM-GRU text feature extraction component.

2. **Pre-trained language models (PLM):** We used final layer features from PLM for extracting the overall feature representation of textual data. We experimented with BERT, RoBERTa, ALBERT, and MPNet. We used the features of [CLS] token for BERT, RoBERTa, and $< s >$ token for MPNet, RoBERTa as final outputs for PLM architecture. We briefly describe the architecture of each model as:
BERT: BERT (Bidirectional Encoder Representations from Transformers), is a language model proposed by Google in the paper [40]. It was the first model to learn bidirectional, unsupervised language representation from unsupervised data. BERT learns these bi-directional representations using a Masked Language Modeling (MLM) objective during the pre-training step. It essentially masks certain words and tries to predict them from neighboring textual data. BERT also uses the Next Sentence Prediction (NSP) objective while pre-training where it tries to learn if the next sentence is a continuation of the first. Once the general representations are learned BERT can be fine-tuned for several downstream tasks.
RoBERTa: A Robustly Optimized BERT pre-training approach (RoBERTa), was proposed by Facebook [41]. It uses the same architecture as BERT. They replicated BERT and measured the impact of hyper-parameters and training size on performance. It was found that BERT was under-trained and, its performance can further be improved. BERT uses static masking which was replaced by dynamic masking. They also trained the model with greater data and with a bigger batch size. Their experiments also found that NSP loss was not useful for the performance of the model and pre-training without it led to equal or better performance hence, they removed it.
ALBERT: A Lite BERT for self-supervised learning of language representations [42] is a modification of BERT aiming to improve the efficiency by allocating the model's capacity more efficiently. This helped reducing memory consumption and shortens training times. The authors focused that input level embeddings need to learn context-independent embeddings while hidden layer embeddings need to learn context-dependent representations. They achieved this using factorization of the embedding parametrization. The embedding matrix was split into input-level embeddings with a relatively lower dimension while the hidden layer embeddings used higher dimensionalities. This step helped in an 80% reduction of parameter size. They also observed that different layers learned to perform similar functions on the output and hence introduced parameter sharing across all layers. These two steps helped reduce the parameter size by almost 89%. They also replaced NSP loss with Sentence Order Prediction (SOP) loss based on coherence.
MPNet: Masked and Permuted Pre-training for Language Understanding [43], was proposed by Microsoft. It aimed to use the best of both Auto-Encoding (AE) and Auto-Regressive (AR) modeling techniques. MPNet builds from the advantages of both MLM and permuted language modeling. MLM cannot learn the dependencies between the masked tokens. permuted language modeling on the other hand cannot see the complete positional information which is visible in downstream tasks. To overcome both these problems MPNet introduced a unified view of permuted language modeling and MLM. To model dependency among predicted tokens, they used two-Stream self-attention. To make input information consistent with downstream tasks they introduced position compensation.
3. **Stacked ensemble of PLM's:** Ensemble models aim to improve the overall performance of a system by combining the outputs of several candidate systems. The main goal is to use the best of each candidate for enhancing the performance. Traditional ensemble techniques include Bagging [62], Boosting [63], and Voting [64]. We use a Multi-Layer Perceptron (MLP) based ensemble model similar to one proposed in [65]. Our approach combines features of all individually trained PLM's. We extract the features generated from the text feature extraction component for BERT, ALBERT, RoBERTa, and MPNet. The features are extracted from already fine-tuned PLM's. We concatenate the features forming the output of the feature extraction component which are fed to the feature classification layer. Only the feature classification layer is trained in this approach as we use features from already fine-tuned models.

5.2.2. Text feature classification

We pass the features from the text feature extraction component through a dense layer of 32 hidden units with elu activation. We normalize the output of the previous layer using Batch Normalization to reduce the activation mean to zero and standard deviation to one. The next and final layer is a dense layer with three units and tanh activation. The final dense layer uses the L2 Regularization of 0.0001 to reduce overfitting. The final layer's output is fed into softmax activation, which predicts the probability of each class. The maximum of the three values is taken as the predicted label of our neural network system.

Layers used for SVNS calculation: Two layers from the above system were used for calculating the SVNS values. The details of intermediate layers are:

- Output of feature extraction component: The output of this layer marks the end of our feature extraction component used for learning the sequential text information. The features in this layer are 64 dimensional in the case of BiLSTM-GRU, 768 dimensional in the case of BERT, RoBERTa, MPNet, and 1024 dimensional in the case of ALBERT.
- Batch Normalization layer: This is the pre-final layer of our feature classification network which contains the 32-dimensional feature vectors for each sample generated from the GRU layer. The values have been normalized using the Batch Normalization layer. This reduces the variance in the features used for SVNS calculation.

6. Experimental setup

6.1. Hyper-parameters:

Text Feature Extraction

1. BiLSTM Architecture: We used 100-dimensional GloVe vectors for representing the words of the tweets. Gaussian noise of 0.2 and a Dropout of 0.3 was added to the embedding layer. Two stacked BiLSTM of size 150 were used on the embedding layer. A Dropout of 0.2 was used after each LSTM and a recurrent Dropout of 0.25 was used. Post attention GRU of size 64 with a recurrent Dropout of 0.25 was used.
2. PLM Architecture: We experimented with BERT, RoBERTa, ALBERT, and MPNet for our pre-trained language model architecture. We used the [CLS] token (trained on SOP/NSP loss) for BERT and ALBERT, while $< s >$ token (start token) was used in the case of RoBERTa and MPNet. The dimension of these features were 768 for BERT, RoBERTa, MPNet, and 1024 for ALBERT. Parameters of each model are given in Table 4.

Text Feature Classification The final dense network for classification consisted of 2 dense layers of size 32 and 3 neurons. The 32-neuron dense layer was followed by Batch Normalization. L2 regularization of 0.0001 was used in dense layers. Dense layers use an elu and tanh activation followed by a final softmax activation.

6.2. Training

We trained/fine-tuned our model to minimize the cross-entropy loss. We used ADAM [66] optimizer for back-propagation. We used a learning rate of 1e-3 for BiLSTM, 2e-5 for RoBERTa, and 1e-5 for BERT, ALBERT, MPNet, and stacked ensemble. We fine-tuned BERT, RoBERTA, ALBERT, MPNet, stacked ensemble using a batch size of 32 for 6 epochs. BiLSTM-GRU was trained for 50 epochs with a batch size of 128. There was an imbalance in training samples for given classes. This may affect the model by adding bias and reduce efficiency. We used class weights to overcome this problem. We used class weights described as:

$$weight_i = max(X)/(X_i + max(X)) \tag{18}$$

where X is the vector containing counts of each given class.

6.3. System setup

1. BiLSTM: We developed the model on python using keras[7] library which is an open-source tool for developing, prototyping, and testing machine learning tasks. Tensorflow backend was used along with libraries like numpy.[8] Pandas[9] and Ekphrasis were used for importing the dataset and applying the pre-processing steps on tweets.
2. PLM: For using PLM's we used the transformers[10] package by Hugging Face. We extended their models using Keras to build our custom models. Tokenization is required for making input consistent with PLM vocabulary. We used Hugging Face implementation of FastTokenizers for each pre-trained model.

All models were developed and trained on Google Colab using GPU/TPU. For SVNS calculation we used scikit-learn's[11] implementation of k-means and cosine distance.

7. Results and analysis

The motive behind the proposed work was to introduce an idea of quantification of sentiment into SVNS values instead of directly predicting the class labels. We proposed a framework to predict the output as a combination of all sentiments so that none of the components i.e positive, negative, neutral is ignored. We proposed to use intermediate layer activations from our neural networks as feature vectors for SVNS calculation. It is evident from the results given in Table 5 that intermediate layers successfully capture the context. SVNS generated using intermediate layers performed equally well and better in most metrics in comparison to the final softmax layer of the neural network. Our stacked ensemble outperformed all other models in all the evaluation metrics. We have underlined the best scores among different layers for each model.

Table 6 shows class-wise evaluation metrics for our best performing model using stacked ensemble. Fig. 5 shows the confusion matrix for predictions made using SVNS generated from the BatchNorm layer of our stacked ensemble.

Fig. 6 depicts sentiment quantification of our proposed SVNS values against the Softmax layer of the classification component. We generated the plots using MPNet. Fig. 6(a) represents the output of softmax from the feature classification layer. We can see from the plot that the outputs are tightly packed. The emotion corresponding to the correct label learns an extremely high probability of prediction completely ignoring the other sentiments. We can even see a very sharp point of inflection between changes in output labels. We also see a sharp triangle in the neutral sentiment. It is because each side captures the transition from neutral to either positive or negative sentiment and thus completely ignoring the third sentiment. Fig. 6(b) and (c) represent the SVNS plots corresponding to BatchNorm and feature extraction output. It is evident from the plots that they capture the

[7] https://keras.io.
[8] https://numpy.org/.
[9] https://pandas.pydata.org.
[10] https://huggingface.co/transformers/.
[11] https://scikit-learn.org/.

Table 4
Hyper-parameters for PLM.

Model	Layers	Hidden state dimension	Number of attention heads	Parameter size
BERT-base-uncased	12	768	12	110M
RoBERTa-base	12	768	12	125M
ALBERT-large-v2	24	128 — Embedding size 1024 — Hidden size	16	17M
MPNet	12	768	12	110M

Table 5
Results on the test set using intermediate layers and final layer (softmax) of feature classification component. In stacked ensemble, PLM ensemble represents the concatenated features of all PLM's we used in our experiments.

Model	Layer	F1	Recall (weighted)	Recall (macro)	Accuracy
BiLSTM (GloVe)	Softmax	0.674	0.675	0.682	0.675
	SVNS (BatchNorm)	0.672	0.676	0.682	0.676
	SVNS (GRU)	0.673	0.677	0.681	0.677
BERT	Softmax	0.703	0.678	0.709	0.678
	SVNS (BatchNorm)	0.707	0.700	0.713	0.700
	SVNS ([CLS])	0.700	0.697	0.710	0.697
RoBERTa	Softmax	0.706	0.680	0.715	0.680
	SVNS (BatchNorm)	0.705	0.684	0.716	0.684
	SVNS (<s>)	0.709	0.703	0.715	0.703
ALBERT	Softmax	0.701	0.680	0.717	0.680
	SVNS (BatchNorm)	0.700	0.692	0.713	0.692
	SVNS ([CLS])	0.661	0.679	0.681	0.678
MPNet	Softmax	0.717	0.700	0.723	0.700
	SVNS (BatchNorm)	0.716	0.706	0.718	0.706
	SVNS (<s>)	0.717	0.702	0.720	0.702
Stacked Ensemble	Softmax	0.720	0.698	0.729	0.698
	SVNS (BatchNorm)	**0.724**	0.714	**0.733**	0.714
	SVNS (PLM Ensemble)	0.721	**0.716**	0.726	**0.716**

Table 6
Metrics for each class using different layers of stacked ensemble.

Layer	Label	Precision	Recall	F1-Score
Softmax	Neutral	0.784	0.569	0.659
	Positive	0.633	0.771	0.696
	Negative	0.662	0.847	0.743
SVNS(BatchNorm)	Neutral	0.758	0.641	0.695
	Positive	0.645	0.770	0.702
	Negative	0.707	0.788	0.745
SVNS(PLM Ensemble)	Neutral	0.745	0.670	0.706
	Positive	0.668	0.736	0.701
	Negative	0.709	0.773	0.740

transition between sentiments smoothly and better quantify the combination of sentiments in the given space. SVNS BatchNorm values though comparatively better than neural networks but are still slightly packed. It is the first layer to receive back-propagated gradients from the final softmax layer leading to such behavior. The output of the MPNet feature extraction component which is further behind in our network experiences a shallow effect of gradients from softmax and thus better learns the combination of sentiments.

Table 7 contains some examples and their corresponding sentiment values calculated using our model. In Table 7, we can see that the neural network has learned to predict the correct output however, it loses information about the other sentiments. e.g., the positive predictions of our neural network assign similar probabilities to the neutral and negative sentiment which, is in complete contrast to real-world sentiments. On the other hand, the SVNS values are better able to quantify the sentiments as observed from the table. We can observe similar patterns for other sentiment classes also.

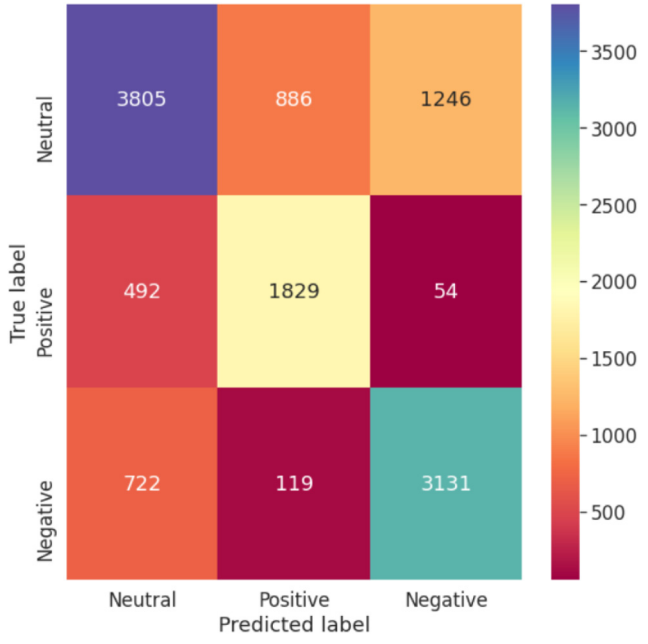
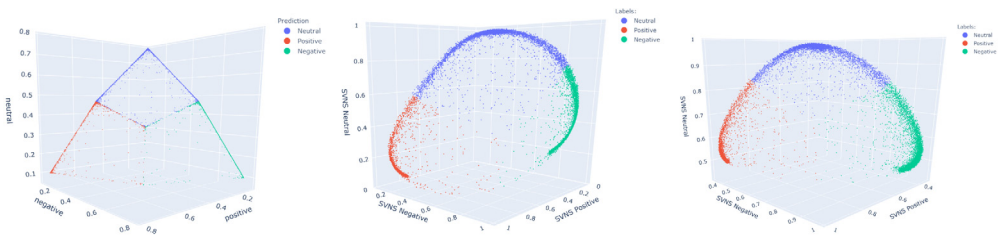


Fig. 5. Confusion matrix for predictions using SVNS (BatchNorm) from Stacked Ensemble.

7.1. Error analysis

The following deals with analyzing the incorrect predictions made by our model. We also try to analyze the possible reasons for the wrong predictions and come up with the following broad reasons:



(a) Softmax (b) Batchnorm (c) PLM

Fig. 6. Probability of prediction for each sentiment using MPNet model (a) Final output of feature classification using softmax (b) SVNS Batch Normalization (c) SVNS PLM ensemble (Concatenated features using BERT, ALBERT, RoBERTa, MPNet).

Table 7
Comparison of probabilities generated using softmax vs. SVNS (BatchNorm) generated from MPNet based model.

Tweet	Layer	Neutral	Positive	Negative	Original Label
@Okupuna funny guy you. But I know you can get a fake id; however, it will cost you a lot more than that to fake a Voter Registration card	*Softmax*	0.439	0.119	0.440	***Negative***
	SVNS BatchNorm	0.607	0.124	0.836	
@cools8n like I saw someone on FB share an article saying that instead of vaccines people should just be exposed to the disease to fight it	*Softmax*	0.445	0.066	0.488	***Negative***
	SVNS BatchNorm	0.512	0.144	0.970	
Um. For many of us, life-threatening. Save #Medicare #Medicaid thou shalt not kill https://t.co/tZoyQIXMSs	*Softmax*	0.463	0.072	0.463	***Negative***
	SVNS BatchNorm	0.599	0.117	0.863	
When a legend speaks about another Legend ♡ #Gerrard #Messi #FCBLive https://t.co/lDsWMBlNuc	*Softmax*	0.144	0.750	0.106	***Positive***
	BatchNorm	0.354	0.953	0.091	
tremendous advances in artificial intelligence are powering everything from self-driving cars to apps that help diagnose cancer	*Softmax*	0.106	0.785	0.108	***Positive***
	SVNS BatchNorm	0.175	0.984	0.229	
If I could be one male celebrity I would be Zac Efron. The dude has his life together.	*Softmax*	0.177	0.721	0.100	***Positive***
	SVNS BatchNorm	0.260	0.951	0.142	
New #Tesla video shows how robot #cars will react when they're launched in 2017. #BlackFriday https://t.co/o9v6GtPE6D	*Softmax*	0.507	0.418	0.073	***Neutral***
	SVNS BatchNorm	0.873	0.385	0.202	
NY's largest Medicaid provider to pull out of LI https://t.co/8zoePx1Dso	*Softmax*	0.750	0.101	0.147	***Neutral***
	SVNS BatchNorm	0.912	0.079	0.625	
I need to make a t-shirt that says 'Let's discuss Westworld!' and wear it practically daily.	*Softmax*	0.468	0.463	0.068	***Neutral***
	SVNS BatchNorm	0.794	0.450	0.167	

URL's and links: Upon observation of the validation set, we found out that certain tweets had URL's pointing to external links containing information about the tweets. Our system had removed URL's. Moreover, information extraction from URL's is not easy and can make the model extremely complex.
Tweet: Undersecretary of Cement on the Mexican Border..BUILD THE WALL.
https://t.co/wUuutXHPvf
Original Label: negative
Predicted Label: neutral

Common sense: Our model does not contain any means to learn common sense reasoning about the general trends present in the world. We as humans have access to an ever-updating knowledge base from the news which we use to make decisions. The model only has the training set which may sometimes not be enough to learn such information.
Tweet: 67.4 kg gold missing from Delhi airport in 7 months - https://t.co/0GU8Iq51BB via https://t.co/8lC6isG73M
Original Label: negative
Predicted Label: neutral

Sarcasm Social media especially Twitter is full of sarcasm based on current events. And sometimes it may be even difficult for humans to get sarcasm. Some examples where our model went wrong are:
Tweet: First Thanksgiving, Christmas, New Years, Valentines Day and Birthday without my Dad
Original label: negative
Predicted label: positive

8. Discussions and comparisons

We further compare the results of our proposed SVNS model to the first five teams from SemEval 2017 Task 4 (Subtask A). With our results given in Table 8, we have highlighted the best scores obtained by our proposed models along with the best score of the competing teams.

Our model has the best accuracy (0.716), F1 (0.724), and recall (0.733) scores over all the other five models, as tabulated in Table 2. Our proposed model using PLM architecture outperformed the top-performing teams. SVNS predictions for most metrics performed equally or better than individually trained models. BiLSTM architecture also performed equivalent to top-performing

Table 8
Comparison of our best performing model (PLM Ensemble) with other models.

Rank no	System	Recall	F1	Accuracy
1	DataStories	**0.681**	0.677	0.651
2	BB twtr	0.681	**0.685**	0.658
3	LIA	0.676	0.674	0.661
4	Senti17	0.674	0.665	0.652
5	NNEMBs	0.669	0.658	**0.664**
	Final (softmax)	0.729	0.720	0.698
Proposed	SVNS (BatchNorm)	**0.733**	**0.724**	0.714
	SVNS (PLM Ensemble)	0.726	0.721	**0.716**

Table 9
Comparison with recent state-of-the-art models.

Model		Recall
TweetEval	RoBERTa-Bs	0.720
	RoBERTa-RT	0.729
	RoBERTa-TW	0.693
BerTweet		0.732
Stacked Ensemble (SVNS BatchNorm)		**0.733**

teams in some evaluation metrics. Although the goal of our system was to develop a better quantification method for the combination of emotions in sentiment analysis, our model must perform at par with the state-of-the-art systems. This would ensure good prediction and a better and improved way for studying sentiment analysis problems.

We also compare our model performance to the most recent benchmarks on SemEval 17 Task 4 (Subtask A) dataset. Comparison is shown in Table 9.

TweetEval [67] is a unified benchmark created for the comparative evaluation of various datasets available for Twitter. It consists of SemEval 17 Task 4 (Subtask A) dataset as the benchmark for the task of sentiment analysis on Twitter. They used three variants of RoBERTa: RoBERTa-Bs, the standard pre-trained RoBERTa model, RoBERTa-RT, standard RoBERTa model retrained on Twitter data, and RoBERTa-TW, trained from scratch on the Twitter dataset. BerTweet proposed in [68] is the first large-scale public dataset trained on 80 GB of data containing 850M tweets. They used the BERT architecture and RoBERTa pre-training procedure to achieve the state-of-the-art result on sentiment analysis on Twitter. It is evident from Table 9 that our stacked ensemble using neutrosophy performs better than all the recent state-of-the-art systems. Our approach using SVNS provided a significant performance gain without using any pre-training procedures and helped achieve state-of-the-art performance.

The idea behind the proposed work was to make a modern sentiment analysis system not only capable of predicting the sentiments but also understanding each component of it. The proposed method can be easily used for querying tweets with not only one specific sentiment but with a varying combination of sentiments varying from slightly positive, positively neutral, etc. This can be very useful for social media campaigns and product feedback analysis from social media. It can be considered as a sentence-to-vector encoding process. The main idea is to combine neutrosophy and deep learning for better quantification of natural sentiment and be equally effective in prediction tasks.

None of the previous work has focused on using deep learning with neutrosophic sets. Neutrosophy covers an important aspect of membership functions using SVNS which can be used for better understanding the task of sentiment analysis. Deep learning has shown significant progress in understanding text. New advances are made every year improving the efficiency of neural networks. Thus, we aim to create a general framework for combining the field of deep learning and neutrosophy which together may serve as a powerful tool for sentiment analysis.

The indeterminacy membership is used to represent the degree of uncertainty in the classification. The relationship among these three memberships can be used to support the confidence level in the classification. Hence, if the classification is involved in the uncertain information then our approach is another technique that can be chosen for the prediction.

8.1. Computational complexity

In [69], they introduced neutrosophic logic SVM (n-SVM), which deals with image segmentation in multi-class problems using the idea of neutrosophic sets. They used the one-against the rest strategy which involved training m-classifiers which is computationally expensive. The model we propose makes use of a single neural network architecture for directly learning the feature space for each positive, negative and neutral class.

In [18], they introduced N-ANN, in which ANN was employed for calculating the positive and negative membership functions. It involved training two separate classifiers one for each truth and false membership function. Training similar neural networks by taking complement can be expensive to train. On the other hand, our model makes use of a single neural network to learn the features used for learning the membership functions

Limitation of other papers for SVNS values: The N-ANN neural network defined in [18] makes use of two ANN for learning the truth and false membership function. But the indeterminacy function is learned as the difference between these two functions. This makes the indeterminacy function directly dependent on truth and false functions. In our proposed model the neural network independently learns features for each membership function which are used to obtain the cluster centers for SVNS calculation. Thus each membership function proposed by us has a range (0,1) and is independent of each other.

9. Conclusions

A neutrosophic approach for deep learning models, which uses state-of-the-art techniques, was proposed in this paper. It is a unique and novel contribution to neutrosophy and deep learning. We have modeled the neutrosophic approach to six models: BiLSTM using GloVe, BERT, ALBERT, RoBERTa, MPNet, and stacked ensemble. The experimental analysis was on SemEval 2017 Task 4 (Subtask A) dataset. We generated SVNS values from two different layers of our DNN models and used the SVNS values to quantify and predict the sentiment. The proposed stacked ensemble model has performed better among all six models that we have proposed. It is also better than the top five teams that took part in the competition and the most recent state-of-the-art models. Natural language is a conglomerate of sentiments, and it is understood differently by each individual. Thus, combining neutrosophy, which captures this uncertainty, with state-of-the-art deep learning techniques can be used to understand and quantify this uncertainty.

CRediT authorship contribution statement

Mayukh Sharma: Conceptualization, Methodology, Software, Visualization, Data curation, Investigation. **Ilanthenral Kandasamy:** Methodology, Writing - original draft, Investigation. **W.B. Vasantha:** Supervision, Conceptualization, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Availability of data and material (data transparency)

The data is from SemEval 2017 Task 4(Subtask A). The data is available at Rosenthal et al. (2017).

Funding

This research received no funding.

Code availability

The code is available on GitHub[12].

References

[1] E. Cambria, Affective computing and sentiment analysis, IEEE Intell. Syst. 31 (2) (2016) 102–107, http://dx.doi.org/10.1109/MIS.2016.31.
[2] S. Rosenthal, N. Farra, P. Nakov, SemEval-2017 task 4: Sentiment analysis in Twitter, in: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEVal-2017), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 502–518, http://dx.doi.org/10.18653/v1/S17-2088, URL https://www.aclweb.org/anthology/S17-2088.
[3] D. Chen, C.D. Manning, A fast and accurate dependency parser using neural networks, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 740–750.
[4] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015, URL http://arxiv.org/abs/1409.0473.
[5] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F.E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing 234 (2017) 11–26, http://dx.doi.org/10.1016/j.neucom.2016.12.038, URL https://www.sciencedirect.com/science/article/pii/S0925231216315533.
[6] M.E. Basiri, S. Nemati, M. Abdar, E. Cambria, U.R. Acharya, ABCDM: An attention-based bidirectional CNN-RNN deep model for sentiment analysis, Future Gener. Comput. Syst. 115 (2021) 279–294, http://dx.doi.org/10.1016/j.future.2020.08.005, URL https://www.sciencedirect.com/science/article/pii/S0167739X20309195.
[7] S. Sun, C. Luo, J. Chen, A review of natural language processing techniques for opinion mining systems, Inf. Fusion 36 (2017) 10–25.
[8] L.M. Rojas-Barahona, Deep learning for sentiment analysis, Lang. Linguist. Compass 10 (12) (2016) 701–719.
[9] F. Smarandache, Neutrosophy, Infinite Study, 2000.
[10] H. Wang, F. Smarandache, Y. Zhang, R. Sunderraman, Single valued neutrosophic sets, Review 16 (1) (2010) 10–14.
[11] S. Broumi, M. Talea, A. Bakali, F. Smarandache, Single valued neutrosophic graphs, J. New Theory 10 (2016) 86–101.
[12] F. Smarandache, A Unifying Field in Logics: Neutrosophic Logic. Neutrosophy, Neutrosophic Set, Probability, and Statistics, American Research Press, Rehoboth, 2000, URL https://arxiv.org/pdf/math/0101228.
[13] I. Kandasamy, W. Vasantha, N. Mathur, M. Bisht, F. Smarandache, Sentiment analysis of the# MeToo movement using neutrosophy: Application of single-valued neutrosophic sets, in: Optimization Theory Based on Neutrosophic and Plithogenic Sets, Academic Press, 2020, pp. 117–135.
[14] I. Kandasamy, W.B. Vasantha, J. Obbineni, F. Smarandache, Sentiment analysis of tweets using refined neutrosophic sets, Comput. Ind. (2019) Accepted.
[15] A. Elhassouny, S. Idbrahim, F. Smarandache, Machine learning in neutrosophic environment: A survey, in: Neutrosophic Sets and Systems, Vol. 28, 2019, pp. 58–68.
[16] P. Kraipeerapun, C.C. Fung, Comparing performance of interval neutrosophic sets and neural networks with support vector machines for binary classification problems, in: 2008 2nd IEEE International Conference on Digital Ecosystems and Technologies, 2008, pp. 34–37.
[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L.u. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc., 2017, URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.
[18] P. Kraipeerapun, C.C. Fung, K.W. Wong, Multiclass classification using neural networks and interval neutrosophic sets, in: Proceedings of the 5th WSEAS International Conference on Computational Intelligence, Man-Machine Systems and Cybernetics, in: CIMMACS'06, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 2006, pp. 123–128.
[19] J. Ye, Single valued neutrosophic cross-entropy for multicriteria decision making problems, Appl. Math. Model. 38 (3) (2014) 1170–1175, http://dx.doi.org/10.1016/j.apm.2013.07.020.
[20] J. Ye, Improved cosine similarity measures of simplified neutrosophic sets for medical diagnoses, Artif. Intell. Med. 63 (3) (2015) 171–179.
[21] J. Ye, Single valued neutrosophic cross-entropy for multicriteria decision making problems, Appl. Math. Model. 38 (3) (2014) 1170–1175.
[22] P. Liu, F. Teng, Multiple attribute decision making method based on normal neutrosophic generalized weighted power averaging operator, Int. J. Mach. Learn. Cybern. 9 (2) 281–293. http://dx.doi.org/10.1007/s13042-015-0385-y.
[23] F. Smarandache, N-valued refined neutrosophic logic and its applications to physics, Prog. Phys. 4 (2013) 143.
[24] I. Kandasamy, Double-valued neutrosophic sets, their minimum spanning trees, and clustering algorithm, J. Intell. Syst. 27 (2) (2018) 163–182, http://dx.doi.org/10.1515/jisys-2016-0088.
[25] I. Kandasamy, Smarandache, Multicriteria decision making using double refined indeterminacy neutrosophic cross entropy and indeterminacy based cross entropy, Appl. Mech. Mater. 859 (2016) 129–143, http://dx.doi.org/10.4028/www.scientific.net/AMM.859.129.
[26] I. Kandasamy, F. Smarandache, Triple refined indeterminate neutrosophic sets for personality classification, in: Computational Intelligence (SSCI), 2016 IEEE Symposium Series on, IEEE, 2016, pp. 1–8, http://dx.doi.org/10.1109/SSCI.2016.7850153.
[27] W. Vasantha, I. Kandasamy, F. Smarandache, V. Devvrat, S. Ghildiyal, Study of imaginative play in children using single-valued refined neutrosophic sets, Symmetry 12 (3) (2020) 402.
[28] I. Kandasamy, W.B.V. Kandasamy, J.M. Obbineni, F. Smarandache, Indeterminate likert scale: feedback based on neutrosophy, its distance measures and clustering algorithm, Soft Comput. 24 (10) (2020) 7459–7468, http://dx.doi.org/10.1007/s00500-019-04372-x.
[29] K. Mishra, I. Kandasamy, V. Kandasamy WB, F. Smarandache, A novel framework using neutrosophy for integrated speech and text sentiment analysis, Symmetry 12 (10) (2020) 1715.
[30] F. Smarandache, M. Colhon, Ş. Vlăduţescu, X. Negrea, Word-level neutrosophic sentiment similarity, Appl. Soft Comput. 80 (2019) 167–176.
[31] A. Jain, B.P. Nandi, C. Gupta, D.K. Tayal, Senti-NSetPSO: large-sized document-level sentiment analysis using Neutrosophic Set and particle swarm optimization, Soft Comput. 24 (1) (2020) 3–15.
[32] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Neural and Information Processing System (NIPS), 2013.
[33] J. Pennington, R. Socher, C. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1532–1543, http://dx.doi.org/10.3115/v1/D14-1162, URL https://www.aclweb.org/anthology/D14-1162.
[34] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, 2013, URL arXiv:1301.3781.
[35] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, A. Joulin, Advances in pre-training distributed word representations, in: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018), 2018.
[36] C. Baziotis, N. Pelekis, C. Doulkeridis, Datastories at SemEval-2017 task 4: Deep LSTM with attention for message-level and topic-based sentiment analysis, in: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEVal-2017), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 747–754, http://dx.doi.org/10.18653/v1/S17-2126, URL https://www.aclweb.org/anthology/S17-2126.
[37] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780, http://dx.doi.org/10.1162/neco.1997.9.8.1735.
[38] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder–decoder for statistical machine translation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1724–1734, http://dx.doi.org/10.3115/v1/D14-1179, URL https://www.aclweb.org/anthology/D14-1179.
[39] Y. Bengio, P.Y. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. 5 2 (1994) 157–166.
[40] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long

[12] https://github.com/04mayukh/Comparison-of-Neutrosophic-Approach-to-various-Deep-Learning-Models-for-Sentiment-Analysis

and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186, http://dx.doi.org/10.18653/v1/N19-1423, URL https://www.aclweb.org/anthology/N19-1423.

[41] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, RoBERTa: A robustly optimized BERT pretraining approach, 2019, URL arXiv:1907.11692.

[42] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, ALBERT: A lite BERT for self-supervised learning of language representations, in: International Conference on Learning Representations, 2020, URL https://openreview.net/forum?id=H1eA7AEtvS.

[43] K. Song, X. Tan, T. Qin, J. Lu, T.-Y. Liu, MPNet: Masked and permuted pre-training for language understanding, in: H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, Vol. 33, Curran Associates, Inc., 2020, pp. 16857–16867, URL https://proceedings.neurips.cc/paper/2020/file/c3a690be93aa602ee2dc0ccab5b7b67e-Paper.pdf.

[44] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: F. Bach, D. Blei (Eds.), Proceedings of the 32nd International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 37, PMLR, Lille, France, 2015, pp. 448–456, URL http://proceedings.mlr.press/v37/ioffe15.html.

[45] L.A. Gatys, A.S. Ecker, M. Bethge, Image style transfer using convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2414–2423.

[46] E. Cambria, Y. Li, F.Z. Xing, S. Poria, K. Kwok, SenticNet 6: Ensemble application of symbolic and subsymbolic AI for sentiment analysis, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, in: CIKM '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 105–114, http://dx.doi.org/10.1145/3340531.3412003.

[47] Y. Wu, M. Schuster, Z. Chen, Q.V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, J. Dean, Google's neural machine translation system: Bridging the gap between human and machine translation, 2016, CoRR abs/1609.08144. arXiv:1609.08144. URL http://arxiv.org/abs/1609.08144.

[48] H. Wang, F. Smarandache, Y. Zhang, R. Sunderraman, Single Valued Neutrosophic Sets, Infinite Study. URL https://books.google.co.in/books?id=RFbVDwAAQBAJ.

[49] M. Cliche, BB_twtr at SemEval-2017 task 4: Twitter sentiment analysis with CNNs and LSTMs, in: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEVal-2017), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 573–580, http://dx.doi.org/10.18653/v1/S17-2094, URL https://www.aclweb.org/anthology/S17-2094.

[50] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1746–1751, http://dx.doi.org/10.3115/v1/D14-1181, URL https://www.aclweb.org/anthology/D14-1181.

[51] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Trans. Assoc. Comput. Linguist. 5 (2017) 135–146.

[52] M. Rouvier, LIA at SemEval-2017 task 4: An ensemble of neural networks for sentiment classification, in: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEVal-2017), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 760–765, http://dx.doi.org/10.18653/v1/S17-2128, URL https://www.aclweb.org/anthology/S17-2128.

[53] H. Hamdan, Senti17 at SemEval-2017 task 4: Ten convolutional neural network voters for tweet polarity classification, in: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEVal-2017), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 700–703, http://dx.doi.org/10.18653/v1/S17-2116, URL https://www.aclweb.org/anthology/S17-2116.

[54] Y. Yin, Y. Song, M. Zhang, NNEMBs at SemEval-2017 task 4: Neural Twitter sentiment classification: a simple ensemble method with different embeddings, in: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEVal-2017), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 621–625, http://dx.doi.org/10.18653/v1/S17-2102, URL https://www.aclweb.org/anthology/S17-2102.

[55] T. Lei, H. Joshi, R. Barzilay, T. Jaakkola, K. Tymoshenko, A. Moschitti, L. Màrquez, Semi-supervised question retrieval with gated convolutions, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, San Diego, California, 2016, pp. 1279–1289, http://dx.doi.org/10.18653/v1/N16-1153, URL https://www.aclweb.org/anthology/N16-1153.

[56] G.D. Forney, The viterbi algorithm, Proc. IEEE 61 (3) (1973) 268–278, http://dx.doi.org/10.1109/PROC.1973.9030.

[57] R. Sennrich, B. Haddow, A. Birch, Neural machine translation of rare words with subword units, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Berlin, Germany, 2016, pp. 1715–1725, http://dx.doi.org/10.18653/v1/P16-1162, URL https://www.aclweb.org/anthology/P16-1162.

[58] M. Schuster, K. Nakajima, Japanese and Korean voice search, in: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012, pp. 5149–5152, http://dx.doi.org/10.1109/ICASSP.2012.6289079.

[59] T. Kudo, J. Richardson, SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 66–71, http://dx.doi.org/10.18653/v1/D18-2012, URL https://www.aclweb.org/anthology/D18-2012.

[60] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015, URL arXiv:1502.03167.

[61] C. Cortes, M. Mohri, A. Rostamizadeh, L2 regularization for learning kernels, in: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, in: UAI '09, AUAI Press, Arlington, Virginia, USA, 2009, pp. 109–116.

[62] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140, http://dx.doi.org/10.1023/A:1018054314350.

[63] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: ICML, 1996, pp. 148–156.

[64] J. Kittler, M. Hatef, R.P.W. Duin, J. Matas, On combining classifiers, IEEE Trans. Pattern Anal. Mach. Intell. 20 (3) (1998) 226–239, http://dx.doi.org/10.1109/34.667881.

[65] M.S. Akhtar, A. Ekbal, E. Cambria, How intense are you? Predicting intensities of emotions and sentiments using stacked ensemble [application notes], IEEE Comput. Intell. Mag. 15 (1) (2020) 64–75, http://dx.doi.org/10.1109/MCI.2019.2954667.

[66] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015, URL http://arxiv.org/abs/1412.6980.

[67] F. Barbieri, J. Camacho-Collados, L. Espinosa Anke, L. Neves, TweetEval: Unified benchmark and comparative evaluation for tweet classification, in: Findings of the Association for Computational Linguistics: EMNLP 2020, Association for Computational Linguistics, 2020, pp. 1644–1650, http://dx.doi.org/10.18653/v1/2020.findings-emnlp.148, Online. URL https://www.aclweb.org/anthology/2020.findings-emnlp.148.

[68] D.Q. Nguyen, T. Vu, A. Tuan Nguyen, BERTweet: A pre-trained language model for english tweets, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, 2020, pp. 9–14, http://dx.doi.org/10.18653/v1/2020.emnlp-demos.2, Online. URL https://www.aclweb.org/anthology/2020.emnlp-demos.2.

[69] W. Ju, H. Cheng, A novel neutrosophic logic svm (n-svm) and its application to image categorization, New Math. Nat. Comput. 9 (01) (2013) 27–42.