# Journal Pre-proof


Deep Learning for predicting neutralities in Offensive Language Identification Dataset

Mayukh Sharma, Ilanthenral Kandasamy, Vasantha Kandasamy

PII: S0957-4174(21)00871-X
DOI: https://doi.org/10.1016/j.eswa.2021.115458
Reference: ESWA 115458

To appear in: *Expert Systems With Applications*

Received date : 11 August 2020
Revised date : 16 June 2021
Accepted date : 19 June 2021

Please cite this article as: M. Sharma, I. Kandasamy and V. Kandasamy, Deep Learning for predicting neutralities in Offensive Language Identification Dataset. *Expert Systems With Applications* (2021), doi: https://doi.org/10.1016/j.eswa.2021.115458.

# Deep Learning for predicting neutralities in Offensive Language Identification Dataset

Mayukh Sharma

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

Ilanthenral Kandasamy*

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

Vasantha Kandasamy

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



*Corresponding author

*Email addresses:* mayukh.sharma2016@vitalum.ac.in (Mayukh Sharma), ilanthenral.k@vit.ac.in (Ilanthenral Kandasamy), vasantha.wb@vit.ac.in (Vasantha Kandasamy)

**Abstract**

Deep learning is advancing rapidly; it has aided in solving problems that were thought impossible. Natural language understanding is one such task that has evolved with the advancement of deep learning systems. There have been several sentiment analysis attempts, but they aim to classify it as a single emotion. Human emotion in natural language is generally a complex combination of emotions, which may be indeterminate or neutral at times. Neutrosophy is a branch of philosophy that identifies neutralities and uses membership functions (positive, negative, neutral) to quantify a sample into Single Valued Neutrosophic Set (SVNS) values. Our work aims to combine the power of deep learning with SVNS to represent a sample's sentiment into membership functions of SVNS. We have worked on the Offensive Language Identification Dataset (OLID). Combining the power of state-of-the-art neural network techniques with neutrosophy allowed us to quantify the sentiments and identify the transition phase between positive and negative ones. We used the transition phase to capture neutral samples, which is beneficial if we want to obtain purely positive/negative samples. We performed experiments using Bi-directional Long Short Term Memory (BiLSTM) with attention, Bidirectional Encoder Representations from Transformers (BERT), A Lite BERT (ALBERT), A Robustly Optimized BERT Approach (RoBERTa), and MPNet. Our SVNS model performed equivalent to state-of-the-art neural network models on the OLID dataset. Here, we propose a novel framework that can integrate with any neural network model and quantify sentiments using SVNS.

*Keywords:* Neutrosophy, SVNS, Sentiment Analysis, BiLSTM, BERT, ALBERT, RoBERTa, MPNet, OLID

## 1. Introduction

Offensive language and hate speech identification are sub-domains of Natural Language Processing (NLP) that explore the automatic identification of offensive language and hate speech from textual data. Understanding these sub-domains help in identifying and reducing offensive language and hate speech on user-generated content, particularly on social media. One of the most popular social media platforms of interest for research is Twitter, where people tweet short posts. Each tweet may comprise URLs or mentions of other entities or users on Twitter. These tweets help people share their opinions regarding political events, public figures, or products. Hence Twitter has become an essential resource of data both for the industry and academia. Its unique insights are relevant for business intelligence, marketing, and e-governance. This data also benefits NLP tasks such as sentiment analysis, offensive language detection, topic extraction, etc.

In the last decade, due to the extensive development of social media platforms and the interactive web, there has been an upsurge in user-generated content dominating the web. All information available online can reach billions of people within moments. It has resulted in a concrete interchange of ideas. However, at the same time, if not used wisely, it also leads to the spread of offensive and probably harmful content over the web. Many dangerous incidents like bullying or hate have predated the internet. However, the extent and reach of the internet have given these incidents the power and influence to harm millions of people's lives.

Individuals make use of the perceived anonymity of computer-based communication, using this to involve in behaviour that many of them would not consider doing in real life. Social media platforms, online communities, and technology companies have been heavily investing in coping with offensive language to prevent abusive behaviour in social media. Filtering by humans is time-demanding, and it can affect individual annotators with post-traumatic stress disorder-like symptoms.

Tweets are micro-texts, limited to 280 characters, over which users interact with each other or post statements. The input is up to the user, giving them the power to include misspellings, emoticons, hashtags, slang, and abusive words, making those messages a valuable source for further analysis. Identifying offensive content from social media platforms is an extremely tough research problem due to variations in the way people can express themselves in a linguistically diverse setting of the web. One significant challenge in monitoring the content produced on social media platforms is the enormous volumes of data created at a breakneck pace from continually varying demographics, linguistic, cultural, and religious communities. Aside from the humongous amount of incoming data, social media platforms also pose a massive challenge to automated information mining tools due to their noisiness, brevity, unusual structure, idiosyncratic language, and ambiguous discourse representation. Information extraction tasks that use state-of-the-art NLP techniques often give unsatisfactory results when applied in such environments.

Most research on identifying hate speech deals with the positive and the negative sentiments, altogether ignoring the neutralities. It leads to a tremendous amount of bias in the way such tweets are identified. Detecting neutralities can play a pivotal role in better categorizing offensive and non-offensive tweets. Certain tweets represent a person's general opinion on a topic, which may not belong to either the offensive or the non-offensive categories. Simultaneously, specific tweets may be a combination of both offensive and non-offensive sentiment, with one of them slightly overpowering the

other sentiment. Thus, instead of simply labeling the sample with a single sentiment, it is essential to quantify the sample in terms of a combination of offensive, non-offensive, and neutral components. It will help better understand the social media sentiments and separate them as per our needs.

In detecting offensive language, the concept of neutrosophy (Smarandache, 1999) can be a formidable tool. Offensive language in social media is dependent on a user's understanding of the text. He/she can choose a combination of attitudes on a tweet that can vary from very offensive to not at all offensive. Moreover, the user may not have a strong feeling about the tweet's offensive content, and the sentiment may fall in the indeterminate(neutral) category. This category is what neutrosophy tries to capture and bases its estimations on the indeterminacies of sentiments present in a sentence.

We use the concept of neutrosophy to identify neutral sentiment present in a dataset of tweets containing only offensive and non-offensive tweets and quantify each sample as a combination of offensive, non-offensive, and neutral sentiment. We use the idea of neutrosophic sets for this purpose. Neutrosophic sets assign membership functions to each sentiment(in terms of offensive, neutral and non-offensive), and the final sentiment corresponds to its independent membership function. We use the SVNS values (Wang et al., 2010) to identify each component $C_i$ of our tweets where $i \in$ {offensive, neutral, not offensive}. This will also help us separate neutral samples from a dataset of offensive and non-offensive samples. We will recall the definition of the neutrosophic sets and SVNS values in the upcoming sections.

Dense neural networks are capable of learning complex mathematical functions. They learn the given data's features layer by layer in a hierarchical manner starting from small features and eventually learning complex features from the combination of small features. Each neuron learns to recognise a particular specific feature and activates. It maps input features into an n-dimensional vector space that corresponds to the output classes. Similar work in the neural style transfer domain is done in (Gatys et al., 2016), where they use the intermediate layers for computation of style loss for generating artistic images. The work showed promising results in generating art images based on the style images provided. Pre-trained models like VGG-NET can also be used as feature extractors as demonstrated in (Alghamdi et al., 2020a) where they use one layer of the VGG-NET model as a feature descriptor of electrocardiogram(ECG) images for detection of myocardial infarction.

Similarly, our proposed model also uses feature vectors generated from intermediate layers for each sample in our dataset. We then use clustering techniques on these feature vectors to get three clusters (neutral, offensive, non-offensive) used to calculate SVNS values for each sample. We used two clustering techniques, i.e. Gaussian Mixture Model (GMM) and k-means. GMM allows us to model each sentiment (neutral, offensive, non-offensive) as an independent normal distribution, allowing us to model SVNS membership functions as the probability of sample belonging to each distribution. In k-means, the centres of clusters represent the point of maximum probability for each sentiment. We can then use normalized cosine distance with each cluster centre to identify how close a point is to the classification set $C_i$. It provides us with a new technique of quantification for each of the given class. An important aspect is when we cluster the features, we can identify the values between the two extreme opposites, i.e., the offensive and non-offensive class; thus, we identify the neutralities. Hence, we can propose a robust model for identifying neutralities from datasets and quantify them. These values can be represented

together in the form of SVNS values, which represent the neutrosophic sets. More recent advancement in the NLP field came in the form of Pre-trained Language Models (PLMs) (Qiu et al., 2020). PLMs are complex neural network models comprising millions of parameters, and they are trained over large amounts of unlabelled text-corpus. It helps them learn to understand natural language and can be easily fine-tuned on small datasets in a supervised learning environment.

**Objectives and contribution of the paper:**

1. Using neural networks to encode the tweets into n-dimensional feature vectors. The neural network is trained against a dataset of offensive and non-offensive tweets. Intermediate layer outputs are used as feature vectors.
2. Using BiLSTM and PLMs (BERT, ALBERT, MPNet, RoBERTa) based architectures for encoding the tweets. Our proposed model can be used with any deep learning model for feature generation.
3. Using GMM and k-means clustering algorithms to separate the above features into three clusters and use them to find the SVNS values.
4. Quantifying tweets as a combination of sentiments (neutral, offensive, non-offensive) using SVNS instead of simply labelling it as offensive/non-offensive.
5. Using neutral membership function of SVNS to extract neutral tweets from a dataset of offensive and non-offensive samples.

Our work aims to combine deep learning and neutrosophy to create a better sentiment analysis model. We propose quantifying the tweet over a group of sentiments using the SVNS values instead of just assigning a single label. It helps us to extract the neutral component from the OLID dataset containing only positive and negative samples.

The paper is structured as follows: Section 1 is introductory in nature. Section 2 justifies using neutrosophy for sentiment analysis, and Section 3 presents a comprehensive description of the existing articles related to deep learning, sentiment analysis, and neutrosophy. Section 4 describes the basic concepts and the proposed system. Then, Section 5 introduces the framework of the proposed model. Section 6 explains the model architecture and experimental setup. Section 7 presents our model results and discussions, and Section 8 compares our proposed models with other models. Lastly, Section 9 concludes the research study and provides direction for future research.

## 2. Why use neutrosophy for sentiment analysis problems?

1. Deep learning models are highly efficient in classification tasks. These systems are trained on supervised learning problems and learn to assign labels to the data as outputs. They use sigmoid activation as the final layer for predicting the probability of sample belonging to a class as $P$. The probability of the sample not belonging to a class is $1 - P$. It may be optimal for dog/cat like classification problems where the sample can be only one of the two labels. However, in the field of NLP, this might not be the case. A sample might be a combination of both positive/negative labels. Neural networks use binary cross-entropy loss for classification problems which is defined as:

$$-Tlog(P) - (1 - T)log(1 - P) \tag{1}$$

where $T$ is ground truth and $P$ is the probability of prediction. We can see that the loss function tries to maximise the probability to 1 for positive samples and minimise the probability for negative samples. It does not train the network to understand language as a combination of all sentiments. On the other hand, neutrosophy studies each sample as a combination of neutrosophic sets, allowing us to understand natural language better.

2. Natural language is a complicated combination of emotions. Understanding human sentiment is challenging, particularly on social media platforms where people use various sentiments and figurative language to express their opinions. Modern neural networks have shown success in language modelling problems like machine translation, sentiment analysis, natural language generation. However, they are inefficient in studying natural language due to positive, negative and neutral sentiments. They assign/classify samples as positive/negative using a threshold on probability values. However, samples may be slightly positive/negative or sometimes even neutral. A classifier with a 50% threshold on probability will assign a positive label to a sample with 51% probability and a negative label to a sample with 49% probability. At the same time, samples with 95% probability will also be assigned a positive label. Neutrosophy tries to study this indeterminacy in predictions and assign samples that are not highly positive/negative into neutral class. Neutrosophy deals with neutralities and uses membership functions to quantify each sample as neutrosophic sets with independent membership functions allowing us to understand natural language in terms of well-formulated sets.
3. Neutrosophy allows us to capture the transition phase between positive and negative samples in binary classification problems. The transition phase consists of samples that lie between positive and negative samples and are predicted with a very low probability. Neutrosophy treats them as neutral samples and quantifies them in terms of neutral membership function, allowing us to separate them easily.
4. In our work, we combine neutrosophy with deep neural networks. We use features from intermediate layers of the neural networks for quantifying data in terms of different membership functions. Our proposed work allows us to use features of different dimensions from various neural network architectures and quantify them in terms of neutrosophic sets. Different models learn a different kind of features. Neutrosophy provides a common framework for quantifying these features into a standard format that can be used to compare their performance.

## 3. Literature survey

Most of the earlier work done in the hate speech domain deals with analysis and identification of the usage of hate speech (Davidson et al., 2017), and the use of abusive languages in online social media platforms (Nobata et al., 2016). Hate speech differs from abusive speech in that the latter's motive is to hurt via means of general slurs made up of derogatory words. A typology scheme of abusive language was proposed in (Waseem et al., 2017). Both hate speech, as well as abusive content, are sub-categories of offensive language. Detailed surveys on all works related to hate speech can be found in (Fortuna and Nunes, 2018; Schmidt and Wiegand, 2017).

The earliest work in hate speech detection done by (Spertus, 1997), used a decision tree-based text classifier for internet pages with an accuracy of 88.2%. Contemporary

work was done by (Sood et al., 2012) on Yahoo news pages and was later followed by (Yin et al., 2016). Offensive tweets were detected using logistic regression over a dataset of tweets with the help of a dictionary of over 339 offensive words by (Xiang et al., 2012). Some work has also been done to identify offensive content in other languages such as German, Spanish etc., and in confronting situations of code switched languages like Hinglish (Mathur et al., 2018). Nevertheless, despite numerous efforts by online moderators and experts, users have continuously come up with creative ways to continue to disguise their abuse using creative alterations contributing to multidimensional linguistic variations (Clarke and Grieve, 2017). Certain people use sarcasm and irony, making it difficult for the sentiment analysis systems to identify hateful content. Work done in (Reforgiato Recupero et al., 2019) proposed a frame-based model for detecting figurative language. Internet memes have become the most recent trend on social media networks. (Sharma et al., 2020a) provided a dataset covering various sentiments and their extent of occurrence on internet memes. Long short term memory(LSTM) based models using transfer learning proposed by (Sharma et al., 2020b) performed best for offensive meme identification and quantifying them. Convolutional Neural Network (CNN) based classifiers can also be used to classify hateful tweets as sexist and racist, as shown in the work of (Badjatiya et al., 2017).

A combination of WordCNN and CharCNN architectures for abusive text classification was introduced by (Park and Fung, 2017). In (Gambäck and Sikdar, 2017), the authors investigated four CNN models trained on word vectors, character n-grams, using the semantic information developed using word2vec, word vectors coupled with character n-grams and randomly generated word vectors to develop a classification system for hate-speech text. (Pitsilis et al., 2018) used an ensemble model of Recurrent Neutral Networks (RNNs) to identify hateful content in social networks. Most recent works in this domain have been on identifying profanity vs hate speech (Kumar et al., 2018), which identifies the challenges of distinguishing between profanity, and intimidating language, which may not at all contain profane language.

In a similar direction, there has been significant work on learning the main intentions and motives behind obscene expressions in social media platforms (Holgate et al., 2018). Various approaches have been used to deal with both textual and multimodal data from social media in general, to develop deep learning classifiers for similar tasks.

We analysed the works done in previous challenges involving hate speech and realised deep learning techniques were used most of the time. We, therefore, focus on using deep learning techniques for our research. LSTM (Hochreiter and Schmidhuber, 1997) and attention mechanism (Bahdanau et al., 2015) allows the machine translators to attain all the information the original sentence contains and then generate the proper output according to the current word. It can even allow the translator to focus on local or global features as mentioned in (Luong et al., 2015). Conv-LSTM can also be used as an alternative to attention (Hassan and Mahmood, 2017). It is a neural network architecture that applies CNN and LSTM; it combined a convolutional and recurrent layer on top of word2vec, an unsupervised pre-training of word vectors. It employed a recurrent layer as an alternative for pooling layers to efficiently capture long-term dependencies for the text classification tasks. Most recent work involving Conv-LSTM (Sedik et al., 2020) demonstrated that data augmentation techniques help improve the performance of Conv-LSTM models and outperformed traditional deep learning models for detection of Covid-19 infections.

Sentiment analysis based on LSTM (Li and Qian, 2016) is the same as a standard RNN, except that memory blocks replace the hidden layer's summation units. The results show that RNN with LSTM can lead to a better accuracy and recall rate than the traditional RNN. In addition, it identifies long term dependencies better than conventional RNN. Dzmitry Bahdanau showed a new mechanism called attention (Bahdanau et al., 2015) in machine translation. It helped in focussing on keywords while looking at the input at timestamp $t$. This mechanism helped in significantly increasing the accuracy of neural networks. Work done in (Wu et al., 2020) uses the affine transformation-based approach to find dominant datasets that can represent original time series data with minimal information loss. We also used various methods that are useful for improving the performance of neural networks and deep learning. We used GloVe embeddings (Pennington et al., 2014). It is a transfer learning method that has shown considerably good results in solving many machine learning tasks. Transfer learning using pre-trained CNN models like VGG-NET and fine-tuning it for predictions(Alghamdi et al., 2020a) obtains good results and reduces overfitting. Sometimes it is not easy to train deep learning models, and transfer learning is not feasible. Work done on features generated from oscillometric wave representation (Alghamdi et al., 2020b) showed good results for human blood pressure estimation using lazy learning algorithms like k-NN and weighted k-NN, providing an alternative when it is difficult to train deep learning systems and use transfer learning.

We used LSTM (Hochreiter and Schmidhuber, 1997) based on deep learning methods for our model. LSTM is known to solve gradient vanishing problem as described in the work (Bengio et al., 1994). Work done in (Wozniak et al., 2020) shows that LSTMs perform extremely well for network malware detection based on data received from IoT devices. LSTMs were also used in body pose prediction in (Woźniak et al., 2021) attaining a high efficiency of 99.89%. We used recently developed techniques such as dropout (Srivastava et al., 2014) and batch normalisation (Ioffe and Szegedy, 2015). These techniques show improvement in the performance of deep learning models and also reduce training time. They allow us to use larger learning rates and also solve the problem of the internal co-variance shift. Dropout acts out as the regulariser adding an effect of Bayesian inference using an average of several models created using dropout.

The concept of neutrosophy, which deals with indeterminacy, was proposed by Smarandache (Smarandache, 1999), which was later transformed into Single Valued Neutrosophic Sets (SVNS) by (Wang et al., 2010). SVNS is a generalisation of fuzzy sets (Broumi et al., 2016). Refined neutrosophic sets introduced by (Smarandache, 2000) and developed in (Kandasamy et al., 2020a; Kandasamy and Smarandache, 2016; Vasantha et al., 2020). They have utilised in sentiment analysis in (Kandasamy et al., 2020c,b; Mishra et al., 2020). The existing conventional or fuzzy sentiment analysis does not capture indeterminacy in the tweets/opinion. We use the concept of SVNS defined by (Wang et al., 2010). We extend the concept of SVNS values to deep learning models and propose a novel architecture for generating SVNS values for text content. At the same time, our method identifies the neutralities in the tweets and generate SVNS values. Bipolar neutrosophic sets (Ali et al., 2017) use truth, indeterminacy, and false membership functions for both positive and negative labels.

## 4. Overview of proposed system

Language modelling is one of the most critical and challenging tasks in NLP. Understanding text and its sentiment have been a topic of extensive research in the recent past. Twitter has evolved to be one of the most critical media for providing a massive amount of data for sentiment analysis. Recent advances in deep learning have shown a considerable amount of progress in sentiment analysis tasks. The advancements in deep learning have led to the development of various architectures for solving NLP tasks. Deep learning techniques make use of neural networks. Specialized neural networks called RNNs play an essential role in language modelling and dealing with sequential data such as text. RNNs have existed for a long time; their popularity increased in recent years due to high computation power availability, making it feasible to train them. They have played a key role in almost every NLP task, from image captioning to machine translation. Most recent advances in NLP include the advent of PLMs. These models are developed using the transformer architecture(Vaswani et al., 2017) which uses an attention mechanism. They use a technique called pre-training(Qiu et al., 2020) to learn semantic features of natural language using unlabelled text-corpus. These models can then be fine-tuned on downstream tasks using a supervised training objective.

Two essential aspects that deep learning networks require are massive amounts of data for training and tremendous computational power. Computational power has been made available due to advances in computing, modern high-end graphic cards and distributed computing, while the other aspect, which is enormous data, is easy to gather from Twitter. These two combined allow us to use neural network architectures for solving complex problems. It is because of the neural networks that AI has progressed at such a rapid scale.

Another important aspect that often remains uncovered in neural networks is the surety of predictions. Neural networks often predict the output correctly but are unsure of the confidence level of their predictions. We use neutrosophy to quantify this indeterminacy using SVNS. We use a neural network to generate features and cluster them for finding the SVNS values. We use GMM and k-means for the clustering process. We use various optimization techniques for our neural network model, which we shall be discussing in the subsequent sections. We focus on transfer learning as an essential component of training our neural networks. We use GloVe embeddings (Pennington et al., 2014) for transfer learning in our BiLSTM system. We also experiment with recent state-of-the-art PLMs like BERT(Devlin et al., 2019), RoBERTa(Liu et al., 2019b), ALBERT(Lan et al., 2020), and MPNet(Song et al., 2020).

Figure 1 presents the high-level architecture of our system. We focus on transfer learning for getting the vectorized representation of our tweets. The sentences go through a cleaning pipeline, which we describe in the upcoming section. The neural network consists of RNN/PLM models for learning sequential information. These features are then passed through a dense neural network that learns features of different dimensions. We use the features from the intermediate layers of dense neural network for computing SVNS values.

### *4.1. Neural networks*

Neural networks represent a set of algorithms that intend to mimic the human brain's working for identifying the hidden relationships within a set of data. Thus, neural net-

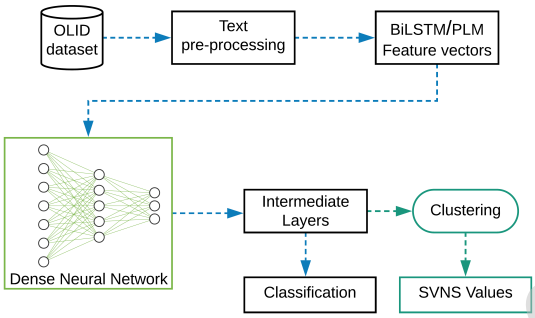



Figure 1: High level overview of our architecture

works represent a system of small units functioning together, trying to mimic our brain. Neural networks are highly robust and quickly adapt to changes in data; thus, they help generate the best possible results without altering the system's overall architecture. A neural network is essentially a system of interconnected nodes called neurons. Each node is a perceptron, and together they behave like a combination of multiple linear regression models. The neurons feed the signals produced by these combinations of linear regression models to a non-linear activation function. In multi-layered neural networks, the neurons are stacked into interconnected layers called hidden layers. The input layers collect the signals, passes them through multiple layers down to the final output layer, which maps the inputs to the classification categories.

Hidden layers learn to identify features from the input signals using the weight vectors to minimize the network's margin of error. The weights are crucial to identifying the salient features representative of the outputs. It is equivalent to features extraction, which is synonymous with statistical techniques like PCA.

### *4.2. Transfer learning*

Deep learning techniques are data intensive. If the dataset size is not large enough, deep learning is not considered to be the right approach, similar is the case of NLP. Everyone wants to solve the problem, but everyone does not have enough data. Big data is a smaller issue than small data. Transfer learning (Tan et al., 2018) is the use of information gained in one task to another semantically similar task. Applying the information from one problem to another similar problem reduces the training time and helps overcome the "small data" problem. Transfer learning is helpful to deep learning in the following ways:

1. More straightforward training requirements using pre-trained data.
2. Much smaller memory requirements.
3. Considerably shortened training time.

Transfer learning has become one of the most critical factors for increasing research in artificial intelligence fields. It allows small research groups and students who lack

computational resources to carry out extensive research in their domains using big organisations technology and features. Transfer learning has been successfully used in several NLP tasks (Alyafeai et al., 2020) and performs better than conventional machine learning algorithms.

### *4.3. Embedding vectors*

A mapping of a discrete variable to a vectorized notation in an n-dimensional space is an embedding. Embeddings in the case of neural networks are continuous vector representations of discrete variables that are low-dimensional representing that variable's characteristics in that vector space. Embeddings are helpful as they meaningfully represent the categories in a transformed space and reduce those variables dimensionality. They are powerful enough to solve word analogies (Ethayarajh, 2019) owing to their linear substructure.

Neural embeddings serve the following primary purposes:

1. Finding closest neighbours in the embedding space can be used to make recommendations based on cluster categories or user interests.
2. For concept visualization and visualization of relations between categories.
3. As input to a machine learning model for a supervised learning task.

One such embedding technique is GloVe(Pennington et al., 2014). It stands for global vectors for word representation. Stanford's unsupervised learning algorithm is open-sourced for generating word embeddings by aggregating a global word-word co-occurrence matrix from a corpus. The resulting embeddings show interesting linear substructures of the word in vector space. In our problem, we will be using GloVE Twitter[1] embeddings for our training purpose. These embeddings have been trained and made available freely by Stanford.nlp group.

### *4.4. Recurrent Neural Networks (RNNs)*

Traditional neural networks assume that all input features are independent. It turns out to be a flawed assumption while dealing with text data. Predicting the next word in a sentence requires prior knowledge of the preceding words. RNNs utilise sequential information; they are recurrent because they perform the same operation for every timestamp of the sequence, where the present timestamp's output depends on the previous computations. Theoretically, RNNs can memorise arbitrarily long sequences, but their range is limited to a few timestamps in practice.

These RNN cells are connected and share the same weights. The activations and the inputs change, but the weights used to compute them remain the same. During backpropagation, the weights are altered for every timestamp; that is why it is called backpropagation through time. RNN suffer from short-term memory. For long sequences, they become incapable of carrying information from earlier timestamps to later ones (Bengio et al., 1994). Exceptionally long timestamps are detrimental to the performance of RNNs. During backpropagation, RNNs suffers from the vanishing gradient problem. Gradients are the lifelines of neural networks that help them learn features. They are

[1] `https://nlp.stanford.edu/projects/glove`

used for updating the weights of the neural networks, which help in identifying important features. The vanishing gradient problem occurs when the gradient shrinks as it back propagates through time. It is because backpropagation uses the chain rule for calculating the gradients. The chain rule is multiplicative. If gradient values become extremely small, they do not contribute much to learning. This will lead to gradients becoming smaller and smaller down the network. So in RNN, layers that get small gradient updates stop learning. These are usually the earlier layers. Because these layers do not learn, RNNs can forget what it has seen in longer sequences, and the information is not carried forward, thus having a short-term memory. There are various approaches to solve the gradient shrinking problem, like gradient clipping, but they do not work very well for RNNs. This poses a pre-eminent challenge while dealing with RNNs. The sentences we deal with in NLP may be as large as paragraphs to whole pages in case of machine translation. Vanishing gradients may also sometimes lead the network weights to become almost constant. That is a ridiculously massive problem as it freezes the gradients, and the loss becomes a constant. The solution to this problem comes from modified recurrent units known as a Gated Recurrent Unit (GRU) (Cho et al., 2014), and LSTM (Hochreiter and Schmidhuber, 1997). These cells contain special memory units that help in carrying information forward in long sequences.

### *4.5. Long Short Term Memory (LSTM)*

The driving idea behind LSTM (Hochreiter and Schmidhuber, 1997) is its states and the gates that control those values. These cell states act as a pathway for the passage of information along the sequence chain. The states are synonymous with the "memory" of the network. Theoretically, the cell states are capable of carrying necessary information down the whole sequence. It is substantial in reducing short-term memory consequences and allows for the passage of information to later timestamps. The cell states are altered along the way using various gates that add or remove the necessary information. The gates are small neural networks that decide what information is to be altered during the training.

### *4.6. Gated Recurrent Unit (GRU)*

GRU is a modification of vanilla RNN that overcomes the problem of vanishing gradients (Bengio et al., 1994). The problem is addressed using specialized "memory" cells, which store information from previous timestamps and improve future predictions. The distinguishing factor between GRUs and RNNs is the gating of the hidden state. GRUs have a dedicated mechanism for updating and resetting the hidden state. These mechanisms, in the form of small neural networks, are learned during training the GRUs. For example, GRU will skip the unimportant information while the important symbols will be updated and passed further to next units in the sequence. The GRUs also learn when to alter the state. GRUs and LSTMs can be differentiated on the frequency of gates and maintenance of cell states. LSTMs have three gates (forget, output, input) and an internal memory state, which makes them computationally expensive but more flexible. However, both can learn long term dependencies.

### *4.7. Attention*

Let us take an example of translating a book's paragraph from English to French. The general approach would not involve reading the whole paragraph and then trying

to translate it at once. Instead, we would re-read and focus on English sentences corresponding to French sentences, which we are writing down. The attention mechanism works similarly and guides the neural network to identify the critical parts.

Attention (Bahdanau et al., 2015) is simply a context vector generated by a combination of dense layers and a softmax function. Without attention, translation requires considering the whole sentence and compressing its features. When the sentences are extremely long, it will undoubtedly lead to data loss and inept translation. Attention solves this problem to some extent. It enables a machine translator to look over all the original sentence's information, then generates the precise word according to the current word and the context. Attention is just a combination of small neural networks which learn complex relationships between the given vector and the sequence. The softmax function, in the end, ensures that the sum of probabilities of attention weights is equal to one. The context vector is just the weighted sum of attention weights and the sequence vectors.

Figure 2 shows the attention mechanism we used as part of our BiLSTM architecture. Attention at timestamp $t$ is calculated for the output of the previous timestamp of post-attention LSTM defined by $S^{<t-1>}$. $X^{<n>}$ represent each timestamp's features vectors for the sequence of length $N$. We concatenate each timestamp's features with the output of post-attention LSTM $S^{<t-1>}$. The joint features are passed through dense layers to learn unit dimensional values $e^{<t>}$. We apply softmax function to $e^{<t>}$ to obtain attention weights $a^{<n>}$. Finally weighted sum of attention weights and feature vectors for the sequence is calculated to find the context vector $c^{<t>}$. This process is carried out for each timestamp for the whole sequence.

### *4.8. Scalar dot product attention*

Scalar dot product attention was introduced in (Vaswani et al., 2017) and is one of the fundamental building block of Transfromers(Vaswani et al., 2017). It is based on the idea of self-attention and is a function of queries, keys, and values. Vectors are used for queries, keys, and values. They are calculated from input embeddings using independent weight matrices for values ($W_v$), queries ($W_q$), and keys ($W_k$). Queries and keys have the same dimension $d_k$. Dimension of values is $d_v$. Attention is a mapping of query and key-value pairs to an output function: It is a weighted sum of values where each value weight computed is a compatibility function of query and its corresponding key. Compatibility function is defined as the dot product of keys and queries, divided by $\sqrt{d_k}$. These values are further passed through a softmax function to compute the attention weights. Finally, the weighted sum of attention weights and values is calculated, giving us the context vector for timestamp $t$ of the sequence $N$. The context vectors for the entire sequence are computed simultaneously using matrix operations. Queries, keys and values are stored in matrices $Q, K, V$ where the dimension of $K$ and $Q$ is $(N, d_k)$, and the dimension of matrix $V$ is $(N, d_v)$. Attention function is defined as:

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

The output of the attention function are context vectors of dimension $(N, d_v)$.

### *4.9. Transformers*

Transformer architecture was first proposed in (Vaswani et al., 2017) and has become one of the most popular choices for solving NLP tasks. Traditional RNN models achieve

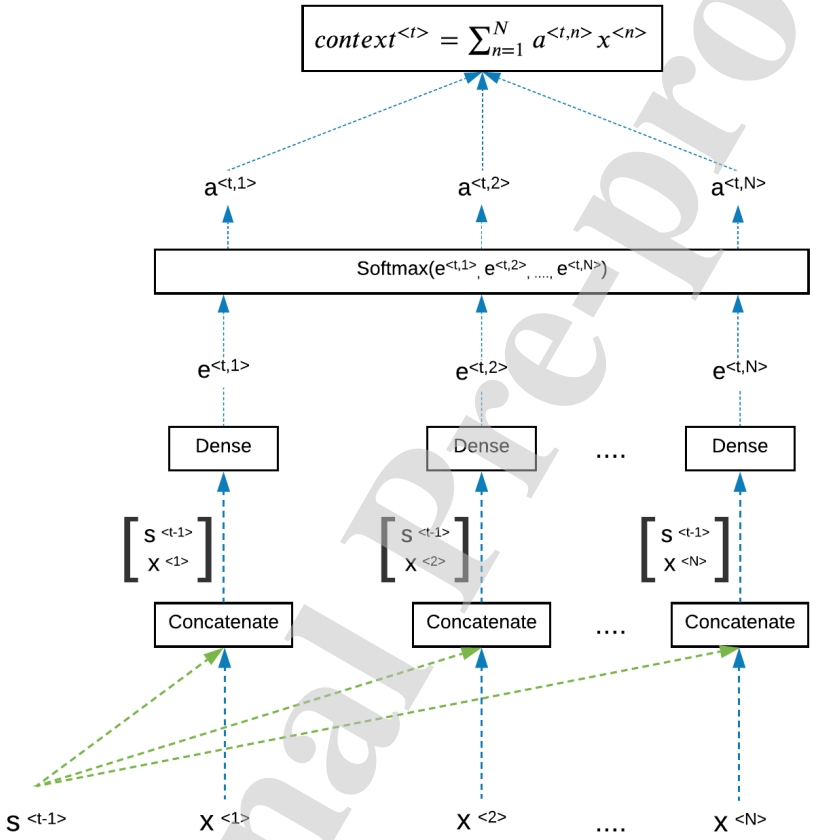



Figure 2: Attention mechanism used as part of our BiLSTM architecture.

good results in NLP tasks, but their sequential nature depends on previous timestamps outputs. This makes it difficult to implement parallel computations on them using recent GPUs.

Transformers overcame this difficulty by removing the sequential nature and focusing entirely on understanding dependencies within sequential data. Dot product attention and multi-head attention (Vaswani et al., 2017) were used for learning these dependencies. Transformers use standard encoder-decoder architecture with attention instead of RNNs for learning dependencies, allowing encoder and decoder blocks to function parallelly and independently during training. For retaining the position based information, they make use of positional embeddings. Detailed architecture can be found at (Vaswani et al., 2017).

### *4.10. Pre-trained Language Models (PLMs)*

NLP contains a wide array of tasks like machine translation, sentiment analysis, language modelling, name-entity recognition, to name a few. Over the past few years, deep learning has become popular in the field of NLP. Deep learning models have shown promising results in different NLP tasks. One problem with deep learning models is that they are data intensive. For training, these models require massive amounts of data for supervised learning problems. Datasets for most NLP tasks contain only a few thousand human labelled samples. This makes training deep learning models on these datasets a challenging task. This problem's solution came with a method known as pre-training(Qiu et al., 2020). The idea behind pre-training is to learn the general representation of text and understand its syntactic and semantic features by training huge amounts of unlabelled text corpus. Unlabelled corpora are easily available, and thus, we can train on more extensive data over greater periods. PLMs can then be fine-tuned on downstream tasks for which labelled datasets are available. Most PLMs use Transformer architecture(Vaswani et al., 2017) with different pre-training objectives. BERT(Devlin et al., 2019), RoBERTa(Liu et al., 2019b), MPNet(Song et al., 2020), ALBERT(Lan et al., 2020) are examples of some PLMs.

### *4.11. Regularization*

Regularization is a process that constrains, regularizes, or contracts the coefficient estimates towards zero. Simply put, this procedure inhibits learning a more complex or flexible model to avoid overfitting. A significant aspect of training any model is evading overfitting because it causes the model will have low accuracy due to the model attempting too hard to capture the noise in the training dataset. The noise here refers to the data points that do not represent their actual properties but random occurrence. The model is flexible at the risk of overfitting because of learning such data points. It leads to a high amount of bias in the model, and the model does not fit well with training points. Training and validation graphs for loss and accuracy are often used to find the bias and estimate the model's goodness. Some forms of regularisation, like L2 norm, prevent the weights from obtaining large values that prevent significant variances in weights, improving the model. Recently a new type of regularisation called dropout (Srivastava et al., 2014) has obtained considerably good results.

#### *4.11.1. Dropout*

Combining different models is always known to increase network efficiency. In reality, we often come across large neural networks, and the idea of averaging the outputs of those individually trained networks is computationally high. The idea of combining several models produces substantial results only when individual models are different from each other. It is possible only when they train on different data or have different architectures. Also, it is challenging to train models with different architectures as we need to tune hyperparameters for each architecture, which is tedious and computationally expensive. Also, training large networks requires a large amount of data, and it is not possible to train models on different subsets of data due to insufficient data. Even if we manage to train various architectures with distinct data sets, using their combination to predict the output at runtime is infeasible where an instant response is required. Dropout is a method that solves both these issues. It provides a method of combining exponentially many architectures and preventing overfitting at the same time.

"Dropout" means dropping out neurons of a neural network randomly. The dropping of units is temporary and includes dropping all incoming and outgoing connections. The choice to drop units is random and guided by a likelihood parameter.

### *4.12. Batch normalisation*

Training neural networks involve adjusting the weights for each layer. The distribution of each layer's input is changed as the previous layer's parameters change. The training process slows down as it requires lower learning rates, careful parameter initialisation, and challenging training models with nonlinearities.

This internal covariate shift problem is addressed by normalising that layer's inputs. Batch normalisation (Ioffe and Szegedy, 2015) induces strength by making normalisation part of the model architecture and implements normalisation for every mini-batch. This provides us with an option to use much higher learning rates and be less concerned about initialisation. Batch normalisation aims to decrease the internal covariate shift and dramatically accelerate the neural network's training speed. It makes use of a normalisation step that fixes the variance and mean of the layer's inputs. It increases the gradient flow through the network as it decreases the dependency of gradients on the parameter's initial value or scale. It enables us to use higher learning rates without the risk of divergence. Batch normalisation empowers us to use saturating nonlinearities by preventing the network from getting held or struck at the saturated nodes.

### *4.13. Activations*

Activation functions are a crucial part of a neural network and help to learn and process non-linear complex mappings between the inputs and the outputs. A neural network without activations is just a linear regression model. Activations introduce non-linearities to our model. They convert an input signal to a non-linear output signal, and the output is then used as the following layer's input in the stack. In a neural network, we use a linear combination of our weights and features and apply an activation to get the current layer's output, which is the input to the next layer. Without the activation functions, our output signal is just a simple linear function (Nwankpa et al., 2018). A linear function represents a polynomial of degree one. This makes our model very simple for learning complex functions. Linear equations though easy to solve, have

limited complexity and less power to learn complicated mappings from data. We want our neural network model to learn more complicated functions. Activations allow the neural networks to learn these complex functions on complex data such as images, videos, or text.

### *4.14. k-means*

The k-means clustering (Sammut and Webb, 2011) is one of the essential and most straightforward unsupervised machine learning algorithms. A cluster relates to a group of data points aggregated together because of specific similarities. When dealing with machine learning problems, we may sometimes come across unlabelled datasets. Unsupervised learning is used most of the time when we need to find certain similarities between the data. The idea behind k-means clustering algorithm is that it makes points with similar features fall into a single cluster. It uses the concept of Euclidean distance for identifying similar features. The k-means is an iterative algorithm that is straightforward yet effective. For the first iteration, we initialize centres randomly. Then, points are assigned to clusters as per their Euclidean distance with the centres. At the end of the iteration, we take the average points for each cluster to assign new centres. Clusters depend on the initialization. The initialization technique proposed in (Arthur and Vassilvitskii, 2007) is known to improve the performance of k-means. Another drawback of k-means is the boundary points. It just assigns clusters, but it fails to tell the confidence with which it predicts the points. This drawback can be covered using GMM, which we discuss next.

### *4.15. Gaussian Mixture Model (GMM)*

Gaussians, each identified by $a \in \{1, \ldots, C\}$, where $C$ is the number of clusters. Every Gaussian $a$ in the mix is made up of the following parameters:

- Its centre defined by a mean $\mu$.
- A covariance matrix $\Sigma$ that defines its correlation between the Gaussians. It is equivalent to an ellipsoid's dimensions in a multivariate scenario.
- A mixing probability $\pi$ defines how small or how big the Gaussian function will be.

GMM (Bishop, 2006) follow a probabilistic approach to solving the clustering problem. The main idea behind GMM is the Bayes theorem. It is an essential part of the GMM. Another important concept used in this case is the latent variable model. We consider each latent variable that causes the data $X$. Using the expectation-maximization algorithm (Bishop, 2006) latent variable models can be trained. The advantage of using Gaussian models for clustering is that we can predict the cluster with our predictions confidence, which is absent in k-means clustering.

### *4.16. Dataset details*

We used the Offensive Language Identification Dataset (OLID) (Zampieri et al., 2019a) for training our model. OLID was the official dataset for SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval)(Zampieri et al., 2019b). We performed our experiments on SubTask A dataset for OffensEval. The

| Keyword | Offensive % |
|---|---|
| medical marijuana | 0.0 |
| they are | 5.9 |
| to:NewYorker | 8.3 |
| you are | 21.0 |
| she is | 26.6 |
| to:BreitBartNews | 31.6 |
| he is | 32.4 |
| gun control | 34.7 |
| -filter:safe | 58.9 |
| conservatives | 23.2 |
| antifa | 26.7 |
| MAGA | 27.7 |
| liberals | 38.0 |

Table 1: Keywords and their offensive content from OLID dataset. First three rows are from initial trial annotations.

dataset samples are annotated as offensive and non-offensive. The dataset was collected using the Twitter API by searching for tweets that contained specifically selected keyword patterns popular in offensive posts. The authors of OLID proposed nine keywords initially for trial basis represented by the first nine rows of Table 1. Experts annotated 300 samples using these keywords. Twitter 'safe' filter, which is used to flag unsafe tweets by Twitter ('-' symbol indicates 'not safe') resulted in the maximum percentage of offensive tweets. Keywords like 'they are', 'to:NewYorker' (to is a Twitter API keyword indicating that the tweet refers to a specific account) that did not result in a high number of offensive content were excluded after the trial. However, Political keywords tend to contain offensive content. So they included 'MAGA', 'antifa', 'conservatives', and 'liberals' in the final annotation process. The whole dataset was then sampled so that 50% of tweets were from political keywords and the rest was non-political. Table 1 shows the keywords used and their offensive content in the entire dataset. The collected data was annotated using Figure-Eight[2](now known as Appen), a popular crowdsourcing platform. Further details could be found in the original OLID paper (Zampieri et al., 2019a). Figures 3 show the distribution of the tweets over prediction classes for offence detection.

We can see from the distribution the imbalance in the class labels. A closer examination of the dataset also shows some discrepancies between the classes definition and their actual annotations. The mislabelling was prominent for the offensive class being mislabelled as non-offensive. Table 2 highlights some samples which we found to be mislabelled. We used our trained model to find wrongly labelled samples in our training set. After training the model, we filtered the wrongly predicted samples with prediction confidence greater than 80%. Let $P$ be the prediction probability for non-offensive, then

[2]http://www.figure-eight.com

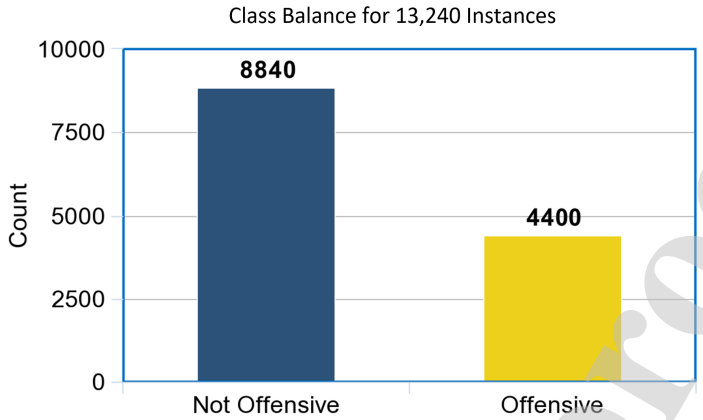



Figure 3: Distribution of class labels for offensive tweets.

| Tweet | Original Label |
|---|---|
| @USER @USER is a DISGRACE to the United States of America!! | Not Offensive |
| @USER @USER When your bio is the most relevant piece of shit ever🤔👀 | Not Offensive |
| @USER Lmao what a joke he is. | Not Offensive |
| @USER @USER found out the name but I forgot. | Offensive |
| @USER I got this | Offensive |

Table 2: Mislabelled samples

$1-P$ will be the probability for the offensive samples. Samples with $P$ or $1-P$ greater than 0.8, which were mispredicted, were considered. These filtered samples were then manually analysed, and we found that some samples were clearly labelled wrong. We needed to make sure these mislabelled examples do not add to bias, so we did proper regularisation to overcome these outliers. This adds robustness to the model.

## 5. Framework of our model

### *5.1. Text pre-processing*

The first step in feeding data to any machine learning algorithm involves cleaning the text. This process plays a vital role in the performance of the model. Random internet texts contain much noise, making it difficult for the classifier to understand their semantics. The tweet dataset represented in OLID consisted of totally raw data directly extracted from Twitter. It contained emojis, hashtags, emoticons, and web URLs, which needed much cleaning to extract useful information from them. The following process of data cleaning was carried out. Table 5 demonstrates some examples after being processed using our proposed pipeline.

*Removing website names*: Tweets usually contain many links to websites and other articles. These are of very little importance for identifying the semantic of that tweet.

| Chat-word | Meaning |
|---|---|
| LOL | laugh out loud |
| FYI | for your information |
| BTW | by the way |
| U | you |

Table 3: Common chat-words with their meanings

| Un-normalised word | Normalised word |
|---|---|
| www.google.com | <URL> |
| @user | <USER> |
| 10 | <NUMBER> |
| Jun 29 | <DATE> |

Table 4: Normalization of common words.

| *Before Pre-processing* | *Post Pre-processing* |
|---|---|
| **@USER** Clearly as long as she is safe no one else matters **lol** 😂😂😂 | **<user>** clearly as long as she is safe no one else matters **laughing out loud** *face with tears of joy face with tears of joy face with tears of joy* |
| **#BiggBossTamil** janani won the task. She is going to first final list👏👏👏👏 | **<hashtag> bigg boss tamil </hashtag>** janani won the task . she is going to first final list *clapping hands clapping hands clapping hands clapping hands* |
| **@USER @USER** Go home you're drunk**!!! @USER #MAGA #Trump2020** 👊**US**👊 **URL** | **<user> <user>** go home you ' re drunk ! **<repeated> <user> <hashtag>** maga **<hashtag> <hashtag>** trump **<number> <hashtag>** *fist* **united states** *fist* url |

Table 5: Pre-processing results using our pipeline.

We usually remove those websites or may standardize them into a common word for all tweets like "URL"

*Chat word conversion*: The millennial slang has spread widely on social media. It has become an integral part of the life of people. We can very often see people using these slangs in their day to day life. However, computers do not understand these chat words. Thus, we need to convert them into their respective meaning. These play a vital role in identifying the sentiment and mood of a user and the tweet. We used a dictionary containing the mapping of popular chat words to their respective meanings to convert them into their respective meanings. Table 3 contains some common chat words. The complete list can be found in our GitHub repository [3].

*Replacing emoticons and emojis with their meaning*: The advent of messenger-based applications has led to the use of emojis and emoticons in conversations. These play a significant role in expressing emotions. The Twitter dataset is full of emojis, and they can be instrumental in identifying the tweet's sentiment. Python package emoji[4] was used for converting emojis to their respective meaning. For emoticons, a dictionary mapping to their respective meaning was done; the dictionary is on our Github[3] page.

The next steps in our pre-processing pipeline made use of ekphrasis[5] (Baziotis et al., 2017) library. We normalize the URLs, @user, numbers, date, etc., so that they are uniform for all the dataset. Table 4 contains examples of normalisation.

We annotate 'hashtag', 'all-caps', 'elongated', 'repeated', 'emphasis', 'censored'. These are used to represent a strong sense of emotion or sarcasm. Identifying that is important

[3] https://github.com/04mayukh/Neutral-tweet-identification-on-OLID-using-Neutrosophy
[4] https://github.com/carpedm20/emoji/
[5] https://github.com/cbaziotis/ekphrasis

and may help in a better understanding of sentiment. The advantage of normalising and annotating is that word embedding vectors used in transfer learning contain vector representations. It helps us to reduce out-of-vocabulary words.

*Hashtag segmentation and spelling correction*: Twitter is full of hashtags. For every tweet, people associate it with a hashtag to represent their views on that topic. They may play a key role in identifying the topic or overall sentiment of the tweet. The last preprocessing step is spelling correction. Wrong spellings may lead to words being left unidentified. This increases out-of-vocabulary words and may negatively affect the model. Thus, we use spelling correction. We used the ekphrasis library for both hashtag segmentation as well as spelling correction. It uses Viterbi algorithm (Forney, 1973) for hashtag segmentation and Peter Norvig's algorithm[6] for spelling correction using the Twitter text corpus. More details on it can be found online. [7].

*Out-of-vocabulary words*: We skipped the out-of-vocabulary words after the earlier preprocessing steps. Maximum out-of-vocabulary words found were spelling mistakes, numbers, unsegmented hashtags, with a shallow occurrence frequency. Vector representations can be trained for them, but it will be computationally expensive and will not be reused much due to their low frequency of occurrence. Out-of-vocabulary words can be replaced by a unique word (<UNKNOWN>) in some pre-trained embedding vectors, making every unknown word equally represented, which may not be correct. Moreover, on analysis, we found the total vocabulary count to be 373,809 and 3664 out-of-vocabulary words, which is 0.0098% of the total count. Thus we ignored the out-of-vocabulary words.

### *5.2. Using tokenisation with PLM models*

As mentioned, we experimented with PLM based models in our work. PLMs do not use word-level vocabulary for generating features. They split the original words into sub-words. This helps in considerably reducing the vocabulary size and re-using the embeddings. Tokenisation techniques like WordPiece (Wu et al., 2016), Byte-Pair-Encoding(BPE)(Sennrich et al., 2016), and SentencePiece (Kudo and Richardson, 2018) are used by PLMs. It is necessary to tokenise the input before we feed them into the models. We tokenised our tweet dataset after completing the pre-processing steps. We used Hugging Face's implementation of FastTokenisers[8] for implementing tokenisation in our PLM based architecture. The tokenisation step also adds some extra tokens to the sequence, like [CLS] used in Next Sentence Prediction (NSP) loss in BERT and Sentence Order Prediction (SOP) loss in ALBERT. <s>(start) and </s>(end) tokens are added in MPNet and RoBERTa. PLMs generate features for all tokens. We use the features generated for [CLS] and <s>(start) token in our experiments to represent the entire text sequence.

### *5.3. Fixing sentence/token length*

Tweets can be of various lengths. They can range from a single word to a few sentences. The varying lengths pose a challenge for RNNs; as RNNs work in mini-batches, we need constant length sentences without which we cannot perform parallel

[6] https://norvig.com/spell-correct.html
[7] https://github.com/cbaziotis/ekphrasis
[8] https://huggingface.co/transformers/tokenizer_summary.html

computations. To overcome this problem, we fix a constant length. Sentences smaller than that are zero-padded, and the one's greater are trimmed.

PLMs also work on sequences of fixed length so we fix the token length for them as well. Unlike for RNNs we first tokenise the complete sentences in our PLM architecture. Once tokenisation is done, we pad or trim the tokenised sequences.

### *5.4. Feature retrieval from middle layers*

Dense neural networks learn complex functions. The initial layers learn simple features, while the deeper layers learn to identify complex features using the simple features learnt in the initial layers. Deeper we go into the neural network, their confidence keeps increasing. The deeper layer learns complex features according to the output classes on which the neural network is trained. We can use the features of these intermediate layers as vector representation for our samples. These features are synonymous with the data on which we train our network. They learn features corresponding to the output classes. We use the features of pre-final layers of our model for calculating the SVNS values. We use cosine distance on clusters of features generated using k-means. The cosine distance between each cluster centre for all data points is calculated. This represents the $\text{SVNS}_\text{x}$ where $x \in \{\text{offensive, non-offensive, neutral}\}$ for each data point.

We also use another method called the GMM for calculating the SVNS values. The GMM is better known for predicting the confidence of each point belonging to a cluster. These confidence values may be directly used as SVNS values, with each cluster representing one component of the SVNS membership functions. This approach works entirely on the probabilistic model, which provides us with a more mathematical way of predicting the SVNS values using Gaussians. The expectation-maximization algorithm is also known to be very robust and efficient for such kind of problems.

## 6. Model architecture

In the previous sections, we defined all the essential components needed to build our system. Next, we combine these tools to define our architecture. Our system architecture consists of two essential parts:

### *6.1. Feature extraction*

The first part deals with the learning features synonymous to offensive and non-offensive tweets. We experimented with two different architectures for feature extraction, i.e. BiLSTM-attention using GloVe embeddings and PLMs. The final features learnt by both architectures are then further used by the next component of our architecture.

**BiLSTM architecture**

1. *Embeddings*: For our model, we used GloVe embeddings trained on the Twitter corpus. We used 100-dimensional embeddings for representing the tweets. The tweets were first cleaned using a pre-processing pipeline defined in the previous sections. The embeddings were passed through a dropout layer with a probability of 0.4. This adds regularisation to our model and helps in better learning.

2. *LSTM network*: The next step in our model involved passing the embeddings vectors through an LSTM layer. We passed the embeddings through a BiLSTM. The use of BiLSTM is more advantageous to understand the context, which may be used later in the tweet concerning a previous word. It helps us obtain a condensed summary of the full tweet. It represents the features extracted from the tweet, which are used by further layers of our network.
3. *Attention*: This is one of the essential components that help improve most of the NLP tasks. In our model, we used attention to find the importance of information up to the given timestamp with respect to the whole tweet. We use attention with a post attention LSTM network. We calculate the context vectors using the output of the post attention LSTM, which contains compiled information up to that timestamp and features corresponding to the tweet obtained from the previous layer. Then, the context vectors are fed to the post attention LSTM for the next timestamp.
4. *GRU layer*: We pass the information of the post attention LSTM onto a GRU layer. The previous layers have learned most of the long-range dependencies, and we use the GRU layer to compile the information further. We use the final timestamp value as the compiled information.

**PLM Architecture**

1. *BERT*: BERT( Bidirectional Encoder Representations from Transformers) was the first PLM to learn bidirectional language representations from unsupervised data. It uses Masked Language Modelling (MLM) as a pre-training objective for learning the bidirectional representations. In MLM, we mask certain words, and the model predicts them using the neighbouring contextual information. It also uses NSP as a pre-training objective to learn if the next sentence is the continuation of the previous sentence. Further information on BERT can be found at (Devlin et al., 2019).
2. *RoBERTa*: Facebook proposed the RoBERTa (Robustly Optimised BERT approach) model, an improved pre-training approach to BERT, the same architecture as BERT. They replicated BERT to analyse the impact of hyperparameters on its performance and found that it was under-trained. They trained the model with more extensive data and greater batch size. Static masking, which was used in BERT, was replaced by dynamic masking. Upon experimentation, NSP loss did not contribute much to the performance and was hence removed. More details can be found at (Liu et al., 2019b).
3. *MPNet*: Microsoft proposed the Masked and Permuted Pre-training for Language Understanding. It is based on the best of both Auto-Regressive (AR) and Auto-Encoding (AE) modelling techniques. The advantages of both MLM and permuted language modelling are used. Permuted language modelling cannot include the positional information of complete sentence, which is visible during downstream tasks. On the other hand, MLM cannot model the dependency between masked tokens. MPNet overcomes both these problems. It makes use of two-stream self-attention for modelling dependencies between predicted tokens. Position compensation was introduced to make input information consistent with downstream tasks. More details can be found at (Song et al., 2020).

4. *ALBERT*: ALBERT (A Lite BERT) is a modification of BERT for self-supervised learning of language representations. It tries to increase the model's efficiency by designating the models capacity more efficiently. This helps in decreasing parameter size and thus reducing training time and memory consumption. ALBERT learns context-independent input embeddings of lower dimension, which are then transformed to a higher dimension hidden layer embeddings within the network to learn context-dependent representations. Earlier PLMs used the same embedding size for input as well as hidden layer embeddings. ALBERT used smaller embeddings for input features, which helped in an 80% decrease in the number of parameters. Parameter sharing was also introduced across all layers. They also changed the NSP loss with SOP loss. Detailed architecture of ALBERT can be found at (Lan et al., 2020).

### *6.2. Feature inference*

We use the features learnt from feature extraction component and use them to obtain feature vectors of different dimensions. We use these features of different dimensions for calculating the SVNS values. Next we define the various components of feature inference.

**Dense network**: By the time we reach the dense layers, our model has learned to compile the features to a certain extent. We use the dense network further to increase the confidence of our model in the predictions. We use a combination of dense layers on the features learnt by the feature extraction component. We use dense layers with different dimensions allowing us to learn multi-dimensional feature vectors. We use them for feature extraction, which we define next.

**Intermediate layer features**: Apart from predictions, we need to calculate SVNS values. We extracted features from the dense network defined above. Since those features are from the pre-final layers, they have learned features related to the training data. We use these features for calculating SVNS values and using the neutral membership function of SVNS to separate neutral samples. We use the outputs of the dense layers of our network for extracting the features.

**k-means**: Once we get the features from our final dense network, we need to separate them into characteristic classes they represent. Our dataset contained just two classes, i.e., offensive and non-offensive. We aim to detect the neutral component from these features. We use k-means clustering to seperate the offensive/non-offensive features into three clusters. k-means has been successful in solving such problems. The features act as a dataset for an unsupervised learning problem. We model the whole neural network to learn those features for the unsupervised learning problem.

**Cosine distance**: Once we find the clusters, we need a method to quantify the features based on these clusters. We choose cosine distance as the tool for this quantification process. Once we have clustered the values into **three** different sets, we extract their centres. These centres represent the point of maximum probability for that class. We use the cosine distance of those features with the centres to quantify the features with respect to that particular class. If we compute the cosine distance between the neutral cluster with all the features of the tweets generated using the dense layer, that will represent how close the features are to being neutral. Cosine distance has a range (0, 2), we normalise it to the range (0, 1). These cosine distances represent the probability of a point being as close to the cluster centres. They can be used as a quantification measure

to predict the nature of a particular tweet. Thus, instead of simply assigning a label to a sample, we have a better representation for each sentiment of the sample.

**GMM**: Another alternative to the k-means algorithm is the GMM. It is another unsupervised learning algorithm capable of separating features based on their characteristic feature. It models the data based on Bayes theorem into a multivariate normal distribution. This can be useful for detecting outliers. At the same time, we get the probability of each sample belonging to a particular cluster instead of simply assigning a label.

**SVNS values**: The SVNS values represent membership probability for each class. Each feature is represented as a value in a range (0, 1) for each membership function in the SVNS set. These memberships are neutral, offensive, non-offensive. These provide a framework to measure a particular tweet based on membership to each of the classes. This helps in better analysing the overall sentiment of the tweet rather than just predicting the output class.

### *6.3. Detailed architecture:*

The detailed architecture for our proposed BiLSTM based model is represented in Figure 4. Next we define the layer-wise architecture for our BiLSTM and PLM based models.

#### *6.3.1. BiLSTM with Attention*

The technical specifications of the layer wise architecture is as follows:

1. Embedding layer – 100-dimensional
2. Dropout – 0.1
3. BiLSTM – 128 units
4. Dropout – 0.4
5. Post-attention LSTM – 256 units
6. GRU – 128 units
7. Batch normalisation – (momentum = 0.99, epsilon = 0.001)
8. Dense – 64 units
9. Dense – 3 units
10. Dense – 128 units
11. Dense – 64 units
12. Dense – 32 units
13. Batch normalisation – (momentum = 0.99, epsilon = 0.001)
14. Dense – 1 unit
15. Activation – Sigmoid

*Hyperparameters*: 100-dimension GloVe vectors were used for representing the text. The length of the timestamp for LSTM and GRU was set to 100. Any sentence below this length was padded with zeros, and sentences greater than this length were truncated. A dropout of 0.2 was used on the embedding layer. A dropout of 0.4 was used after the first LSTM layer, a dropout of 0.1 was used after the GRU layer, and dropout of 0.2 was used for recurrent connections of GRU and LSTM. L2 regularization of 0.001 is used in the final dense layers. All dense layers used 'elu' activations. The outputs of the final layers used a sigmoid activation. Both clustering algorithms, i.e., k-means and GMM were trained using 3 clusters. We used intermediate layers of size 3, 32, and 128 for calculating the SVNS values.

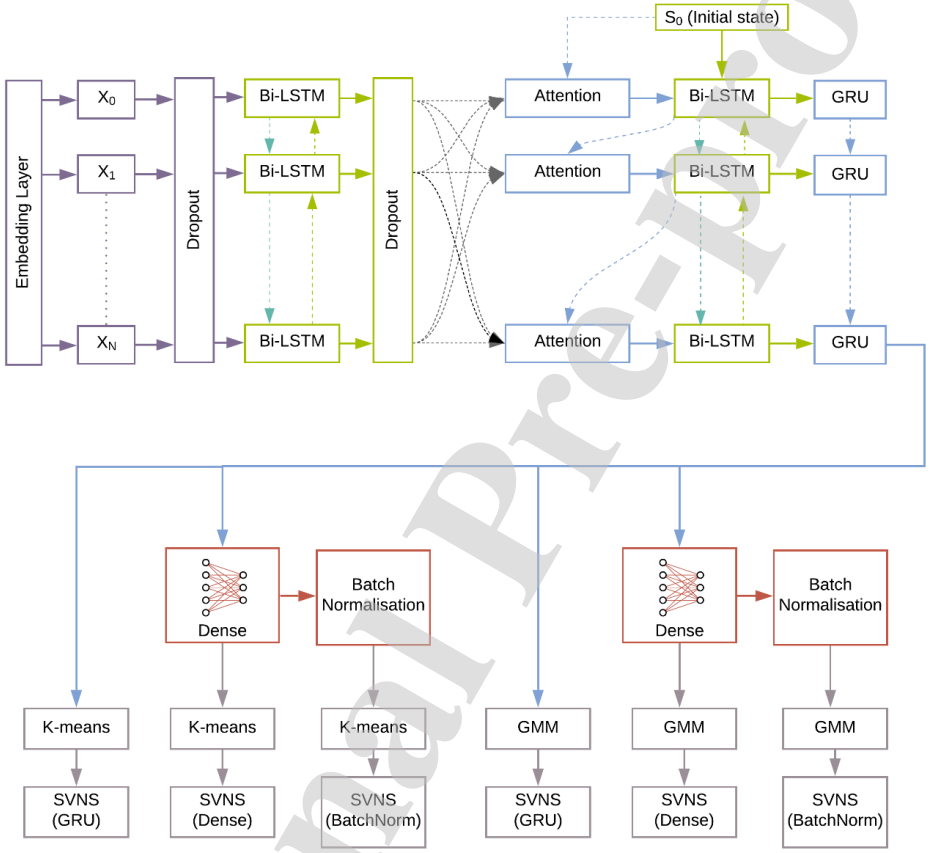



Figure 4: Detailed architecture of our BiLSTM model

| 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<br>1024 – Hidden size | 16 | 17M |
| MPNet | 12 | 768 | 12 | 110M |

Table 6: Hyperparameters for PLMs used in our feature extraction component.

*6.3.2. PLM models*

Our PLM based architecture made use of four different types of PLM models namely: BERT, ALBERT, RoBERTa, and MPNet. Outputs of features learned by these models were fed as input to the feature inference component. Detailed architecture is defined as follows:

1. PLM layer ([CLS]/<s>) – 768
2. Batch normalisation – (momentum = 0.99, epsilon = 0.001)
3. Dropout – 0.1
4. Dense – 128 units
5. Dense – 32 units
6. Dense – 3 units
7. Dense – 32 units
8. Batch normalisation – (momentum = 0.99, epsilon = 0.001)
9. Dense – 128 units
10. Dense – 1 units
11. Activation – Sigmoid

*Hyperparameters*: The length of tokens for PLMs was fixed to 100. L2 regularization of 0.001 is used in the final dense layers. All dense layers used 'elu' activations. The outputs of the final layers used a sigmoid activation. Both k-means and GMM were trained using 3 clusters. We used intermediate layers of size 3, 128, and 768 for calculating the SVNS values. Hyperparameters for PLMs are mentioned in Table 6.

*6.4. Training:*

**Validation and test sets:** We used the OLID dataset for training our model. The downloaded dataset was already split into train and test sets. This split ensures the same samples in a test set for comparing the model's performance with other models, which may not be the case if we perform the train-test split ourselves. While training deep learning models, validation sets can help fine-tune the model. So we performed an 80-20 split on the training set. One set was used for training, while the other was used as a validation set for fine-tuning the model. The statistics of the dataset can be found in Table 7.

**Optimiser used**: Adam(Kingma and Ba, 2015) is an optimisation algorithm with an adaptive learning rate explicitly designed for training deep neural networks. Adam was used for training all our models. We used different learning rates for different models. We used a learning rate of 1e-3 for training the BiLSTM model. For training PLM models we

| | Training set | Validation set | Test set |
|---|---|---|---|
| Number of tweets | 10592 | 2648 | 860 |
| Offensive tweets | 3512 | 888 | 240 |
| Non-Offensive tweets | 7080 | 1760 | 620 |

Table 7: Dataset statistics for training our models.

experimented with learning rates of 1e-5, 2e-5, and 5e-5 for different PLM architectures and used the weights with best results on the validation set. Similar range of learning rates were used in the original papers of these PLMs(Devlin et al., 2019; Liu et al., 2019b; Song et al., 2020; Lan et al., 2020) for fine-tuning tasks. Learning rate of 2e-5 had best performance for MPNet and RoBERTa, 5e-5 for BERT, and 1e-5 for ALBERT.

**Loss function**: Binary cross-entropy, also called sigmoid cross-entropy loss was used. It is a sigmoid activation plus a cross-entropy loss, used for training binary classification problems. It predicts the probability of a sample belonging to one of the classes, it is defined as:

$$-Tlog(f(s)) - (1-T)log(1-f(s))$$

where $s$ is the input, $f(s)$ is output of training function, and $T$ is the true label for $s$.

**Dataset statistics**: The dataset statistics are provided in Table 7, it is seen clearly that there is a class imbalance in the dataset. Class weights were used to overcome this problem which we define next.

**Class weights**: The goal is to learn equally good features for both offensive and non-offensive samples, but we do not have enough positive samples to work with, so the classifier must weigh the few available examples heavily. It is done by passing weights for every class through a parameter. These will cause the model to "pay more attention" to examples from an under-represented class. Let $X$ be the vector containing counts of each class $X_i$ where $i \in X$. Then the weights for each class is given as:

$$weight_i = \frac{max(X)}{X_i + max(X)}$$

**System setup**: The model was developed on python using keras[9] library which is an open-source tool for perofrming machine learning tasks. Tensorflow[10] (Abadi et al., 2015) backend was used along with libraries like numpy[11]. Pandas[12] was used for importing the dataset and pre-processing the tweets. For developing PLM based models we used the transformers[13] package by Hugging Face(Wolf et al., 2020).

**Evaluation metrics**: The F score, also known as the F measure or the F1 score, is used to measure the test set's efficiency. It is defined as the weighted harmonic mean of the test set's precision and recall. The score is calculated as:

[9] https://keras.io
[10] https://tensorflow.org
[11] https://numpy.org/
[12] https://pandas.pydata.org
[13] https://huggingface.co/transformers/

$$F_1 = 2.\frac{precision \times recall}{precision + recall}$$

using the precision and recall scores of the test set. Precision, also known as the positive predictive value, is the proportion of positive results that truly are positive. Recall, known as sensitivity, is a model's ability to identify positive results to get the true positive rate correctly. The F score reaches the best value, meaning perfect precision and recall, at a value of 1. The worst F score, which means the lowest precision and lowest recall, would be 0. We used macro-averaged precision, recall, and F1-scores while reporting our results. Other performance metrics used include accuracy, specificity, False Discovery Rate (FDR), False-Positive Rate (FPR), False-Negative Rate (FNR), False Omission Rate (FOR), Negative Predictive Value (NPV). We used F1(macro) as our primary evaluation metric as used in OffensEval(Zampieri et al., 2019b). We will be using these evaluation metrics for comparison.

## 7. Results

### *7.1. Training curves*

The training curves are presented in Figure 5 and Figure 6. We trained our BiLSTM model for 24 epochs and PLM based models for 6 epochs. For all models the training set accuracy (Figure 5) constantly increases accompanied by a smooth decrease in loss (Figure 6). The validation accuracy reaches a maximum for BiLSTM (Figure 5a) and then gradually decreases. For PLM based models there was fluctuation in validation accuracy (Figure 5b,5c,5d,5e) with no general trend. The loss curves for validation set (Figure 6) show a decrease for all models followed by an increase which indicates the model starts overfitting. We used the weights with minimum validation loss for evaluating our models on the test set. We also observed that weights with minimum validation loss had best validation accuracy across all models.

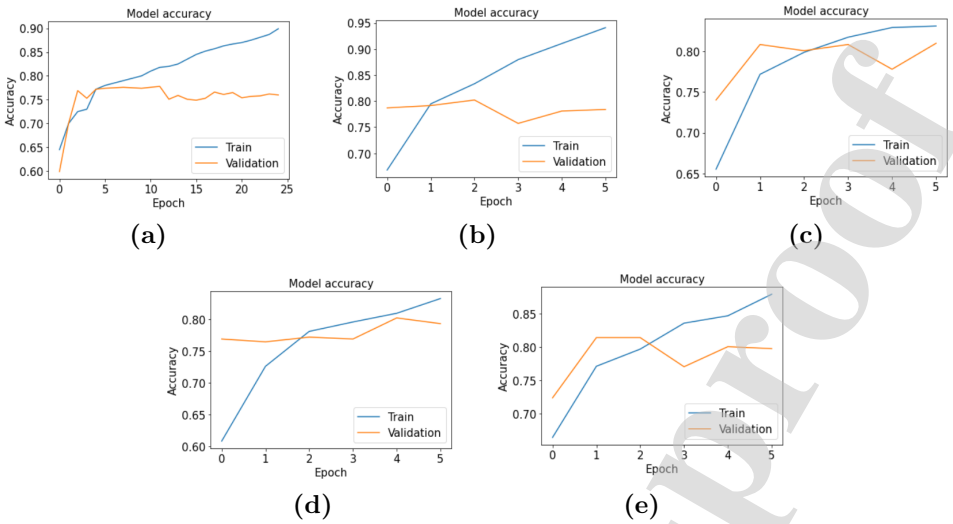



Figure 5: Train-Validation set accuracy curves for **(a)** BiLSTM **(b)** BERT **(c)** MPNet **(d)** ALBERT **(e)** RoBERTa

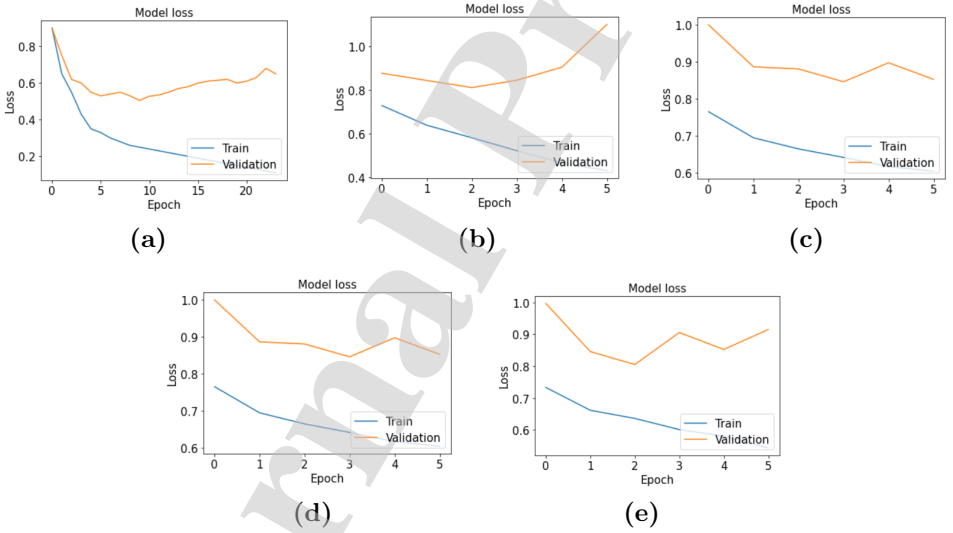



Figure 6: Train-Validation set loss curves for **(a)** BiLSTM **(b)** BERT **(c)** MPNet **(d)** ALBERT **(e)** RoBERTa

### *7.2. Performance metrics and confusion matrices*

We evaluated our models performance on the test set using the weights with minimum validation loss during our training procedure. The results are shown in Table 8. Precision,

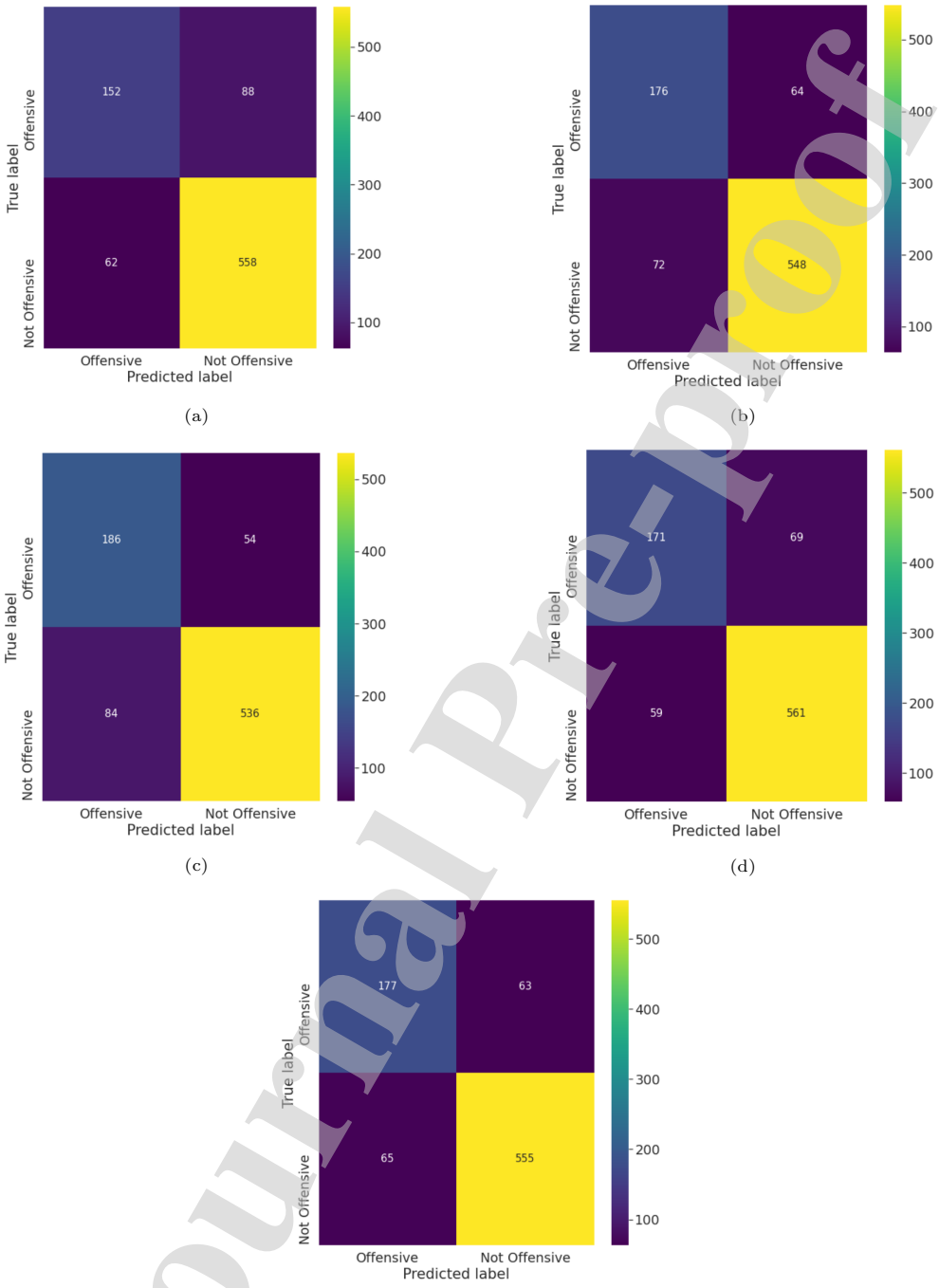



Figure 7: Test set confusion matrices for (a) BiLSTM (b) BERT (c) MPNet (d) ALBERT (e) RoBERTa

| Model | Precision | Recall | F1 | Accuracy | Specificity | FDR | FPR | FNR | FOR | NPV |
|---|---|---|---|---|---|---|---|---|---|---|
| BiLSTM | .787 | .767 | .776 | .826 | .864 | .367 | .136 | .290 | .100 | .900 |
| BERT | .803 | .809 | .805 | .842 | .895 | .267 | .105 | .290 | .116 | .884 |
| MPNet | .799 | **.820** | .808 | .840 | **.908** | **.225** | **.092** | .311 | .135 | .865 |
| ALBERT | **.817** | .809 | .813 | **.851** | .890 | .287 | .110 | **.257** | **.095** | **.905** |
| RoBERTa | .815 | .816 | **.816** | **.851** | .898 | .263 | .102 | .269 | .105 | .895 |

Table 8: Evaluation results on test set using neural network architecture.

recall, and F1 scores in Table 8 are macro-averaged due to the class imbalance present in the dataset allowing us to better evaluate our models. We observe that PLM models performed better than BiLSTM model in most of the metrics. We can also observe from Table 8 that BiLSTM model had a significantly high FDR in comparison to PLM models, which negatively affected its performance. Among PLM models, RoBERTa and ALBERT performed slightly better than BERT and MPNet. Both RoBERTa and ALBERT had lowest FNR, leading to a slight improvement in performance with respect to other models. The confusion matrices for each model are presented in Figure 7.

### *7.3. Analysis of dense layer outputs*

We use the feature vectors from the dense layer with three neurons for visualising the features learnt by our neural network system. The three neuron layer is the last layer of compressing features from where we again obtain high dimensional features. Therefore we use it for analysis of our learnt features. We use the final layer probabilities for labelling the offensive and non-offensive tweets, which were predicted with more than 70% confidence by our model. We used a final layer with a single neuron and sigmoid activation, which was trained using binary cross-entropy loss. The sigmoid function in the final layer of our neural network, outputs values ranging from 0 to 1. The labelling scheme used 0 for offensive and 1 for non-offensive. Let the neural network output be $P$, which is in range (0, 1). So, the probability of a tweet being non-offensive is $P$ and being offensive is $1 - P$. Thus, to get offensive and non-offensive tweets with confidence greater than 70% we used the range (0, 0.3) for offensive and (0.7, 1) for non-offensive. The rest of the samples in the range (0.3, 0.7) were treated as neutral.

Thus, for getting tweets with more than 70% prediction accuracy we used the following threshold values on our final layer output:
*(0, 0.3) Offensive*
*(0.7, 1.0) Non-offensive*
*(0.3, 0.7) Neutral*

Figure 8 shows 3-dimensional features learnt by our PLM models. The features were labelled using the threshold values we defined above. It is evident from the figure that there are many samples for which the prediction probability is less than 70%. These samples represent the transition from offensive to non-offensive class. They may be completely neutral or may contain a similar proportion of both positive and negative sentiments, which may confuse our model. We treat these samples as neutral/indeterminate.

Plots in Figure 8 clearly show the presence of neutral/indeterminate samples in the transition between offensive and non-offensive features. Each PLM model in Figure 8 learnt features over different range of values and distributions but had indeterminate/neutral features in the transition between offensive/non-offensive features.

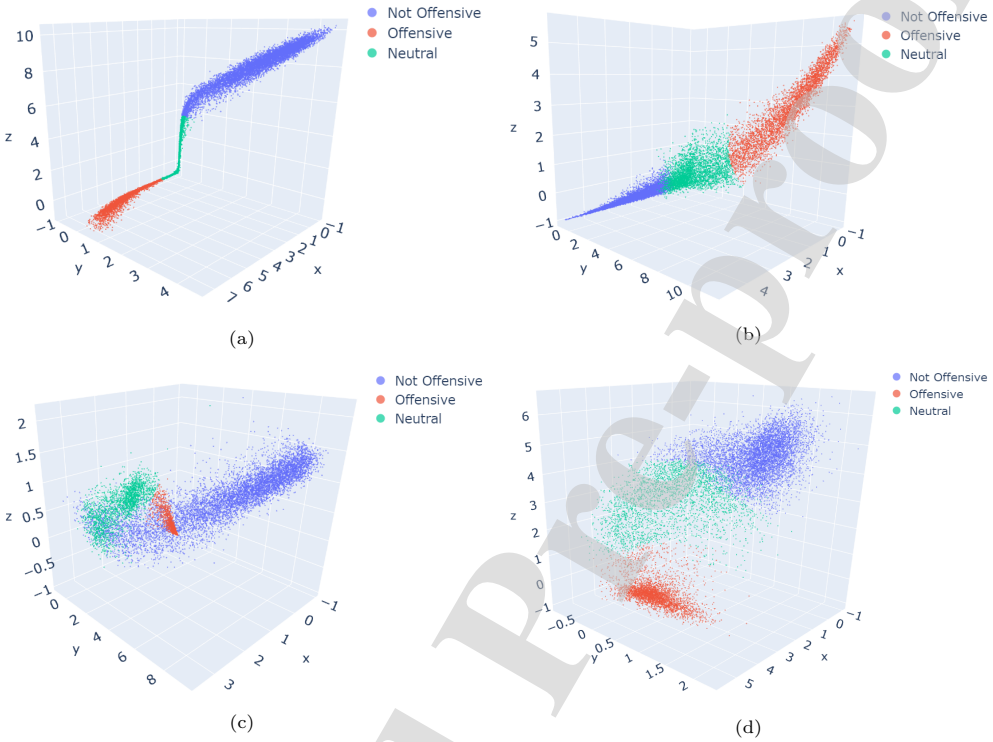
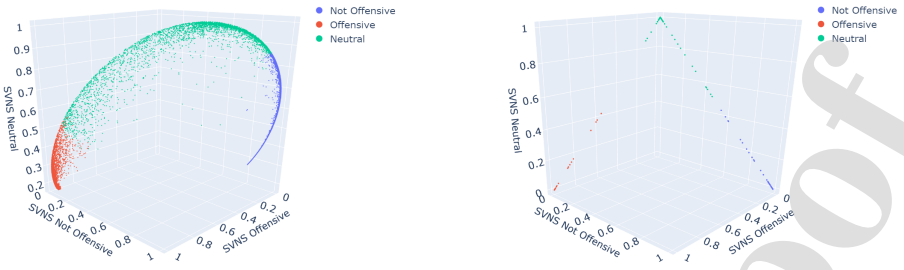



Figure 8: Features separated using the threshold values as offensive, neutral/indeterminate, and non-offensive for (a)RoBERTa (b)MPNet (c)BERT (d)ALBERT.

### *7.4. SVNS values and clustering*

As per our analysis of the dense layer outputs, we found uncertainty in predicting some samples. The neural network usually assigns a label to these sample but with less confidence. Human sentiments are difficult to understand. Humans use a combination of sentiments to express themselves. People on social media platforms like Twitter use sarcasm and irony to express their sentiments (Reforgiato Recupero et al., 2019). Assigning a single label (offensive/non-offensive) does not consider the combination of sentiments (offensive, non-offensive and neutral) that may be used on social media. Here we use the idea of neutrosophy which help us better understand human sentiments. We use SVNS as described before for quantifying a tweet into membership functions. These membership functions are defined as $\text{SVNS}_x$ where $x \in$ {offensive, non-offensive, neutral} for each sample.

The analysis of feature vectors from the dense layer with three neurons showed that we could separate those features into three different sets $C_x$ were $x \in$ {offensive, non-




(a) k-means (b) GMM

Figure 9: SVNS values calculated using Batch Normalisation layer features from MPNet.

offensive, neutral}. We used two clustering techniques, namely k-means and GMM, for separating these features into three different clusters $C$. We experimented with features from three different layers for each of our BiLSTM and PLM architecture.

Once the features are separated in clusters $C$, we use them for calculating the SVNS membership functions for each sample. For prediction using SVNS, we take the maximum of the three membership functions and assign the sentiment corresponding to that membership function to the sample. We used outputs of our neural network as true labels. Samples with less than 70% confidence were labelled as neutral/indeterminate and the rest as offensive/non-offensive. Table 9 show prediction results using SVNS membership functions calculated on feature vectors generated on the training set. It is evident from the results that SVNS membership functions can successfully separate the given features into three different sets $C$ efficiently. Dense-3 layer features clustered using k-means for BERT, and BatchNorm(Batch Normalisation) layer features clustered using GMM for MPNet had the best performance attaining more than 95% score on all evaluation metrics.

The SVNS plots generated from features of different layers along with layer-wise description are as follows:

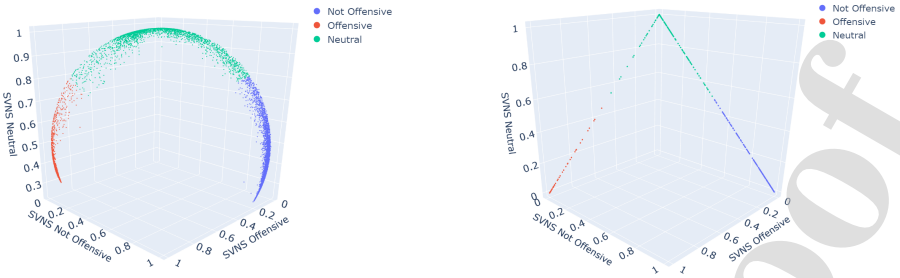
***Batch normalisation layer***:The pre-final layer of our architecture for both BiLSTM and PLM models is the Batch Normalisation Layer. The size of the features is 32 for BiLSTM and 128 for PLM BatchNorm layer. MPNet had the best performance for this layer using both k-means and GMM. SVNS values calculated using k-means and GMM are given in Figure 9a and Figure 9b respectively.

***Dense-3 layer***: This layer contains highly compressed features. The size of the features is three. We normalised the features of this layer. BERT features had the best performance for this layer using k-means, performing over 95% for all evaluation metrics. RoBERTa features clustered using GMM had best results in comparison to other models for this layer. SVNS values calculated using k-means and GMM are given in Figure 10a and Figure 10b respectively.

***GRU/PLM layer***: The GRU layer represents the final layer of our recurrent network. The PLM layer represents the output of the transformer network. We used [CLS] token features for BERT, ALBERT and $< s >$ token features for MPNet, RoBERTa. The size of the features is 128 for GRU and 768 for the PLM layer. RoBERTa and BERT had a similar performance for features of this layer using k-means. RoBERTa had the

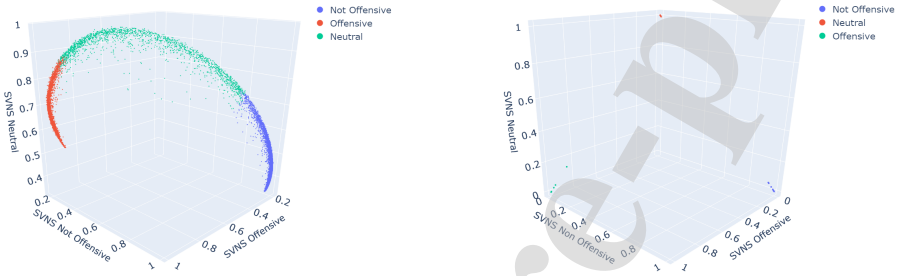
| Model | Clustering | Layer | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|---|---|
| BiLSTM | k-means | GRU | .878 | .897 | .877 | .884 |
| | | Dense-3 | .766 | .801 | .741 | .781 |
| | | BatchNorm | .844 | .880 | .852 | .891 |
| | GMM | GRU | .890 | .911 | .894 | .903 |
| | | Dense-3 | .771 | .808 | .766 | .794 |
| | | BatchNorm | .899 | .934 | .911 | .923 |
| BERT | k-means | PLM([CLS]) | .918 | **.964** | **.936** | .949 |
| | | Dense-3 | .956 | .982 | .968 | .975 |
| | | BatchNorm | .878 | .932 | .899 | .920 |
| | GMM | PLM([CLS]) | .884 | .942 | .899 | .915 |
| | | Dense-3 | .818 | .857 | .795 | .809 |
| | | BatchNorm | .848 | .902 | .868 | .898 |
| MPNet | k-means | PLM(<s>) | .897 | .881 | .886 | .896 |
| | | Dense-3 | .871 | .826 | .823 | .857 |
| | | BatchNorm | .934 | .945 | .935 | .936 |
| | GMM | PLM(<s>) | .860 | .840 | .843 | .855 |
| | | Dense-3 | .899 | .904 | .890 | .890 |
| | | BatchNorm | **.969** | **.969** | **.969** | **.970** |
| ALBERT | k-means | PLM([CLS]) | .829 | .791 | .802 | .835 |
| | | Dense-3 | .892 | .900 | .896 | .905 |
| | | BatchNorm | .849 | .864 | .855 | .866 |
| | GMM | PLM([CLS]) | .837 | .805 | .812 | .844 |
| | | Dense-3 | .787 | .824 | .792 | .807 |
| | | BatchNorm | .841 | .869 | .842 | .840 |
| RoBERTa | k-means | PLM(<s>) | **.954** | .918 | .930 | **.954** |
| | | Dense-3 | .972 | .906 | .931 | .947 |
| | | BatchNorm | .926 | .916 | .915 | .944 |
| | GMM | PLM(<s>) | .910 | .945 | .921 | .933 |
| | | Dense-3 | .880 | .911 | .879 | .894 |
| | | BatchNorm | .918 | .964 | .935 | .940 |

Table 9: Evaluation scores(macro-averaged) for separating the features into offensive, neutral/indeterminate, and non-offensive categories using SVNS values.




(a) k-means (b) GMM

Figure 10: SVNS values calculated using Dense-3 layer features from BERT.




(a) k-means (b) GMM

Figure 11: SVNS values calculated using PLM layer features for RoBERTa.

best performance using GMM over this layer's features. SVNS values calculated using k-means and GMM are given in Figure 11a and Figure 11b respectively.

**Analysing SVNS plots generated using k-means and GMM**: From the SVNS-value plots from different layers, it is evident that the k-means clustering algorithm provides us with a better representation of SVNS values. It can smoothly capture the transition between neutral, offensive, and non-offensive features in terms of SVNS membership functions. In the case of SVNS using GMM, we can see a triangular plot for every layer. It also has a sharp point of inflexion in the case of neutral SVNS values. Moreover, the SVNS calculated using GMM are either along the xz-axis or yz-axis, thus capturing only two sentiments, i.e. (neutral, offensive) or (neutral, non-offensive). Hence, it will be unable to capture neutral/indeterminate sentiments existing as combination of both offensive and non-offensive sentiments. Thus we conclude k-means performs better for SVNS calculation. A sample of SVNS values generated using k-means and GMM from our BiLSTM architecture is given in Table 10.

**Significance of SVNS values**: SVNS values allow us to model the sentiments in terms of membership functions. The range of membership functions is from 0 to 1. The advantage of using SVNS is that it allows us to quantify the features using a well-defined framework. If we carefully observe the Dense-3 features in Figure 8, we can see that features are spread over different values for different models. Each model learns a different representation of features. Similar will be the case for other high dimensional features from different layers. Using SVNS, we try to map the features into membership

| Tweet | Clustering | SVNS Offensive | SVNS Not Offensive | SVNS Neutral | Original Label |
|---|---|---|---|---|---|
| @USER Legit got the saying written down. But you are right. | K-means | 0.323 | 0.508 | 0.916 | Not Offensive |
| | GMM | 0.0 | 0.0 | 1.0 | |
| @USER This is NOT lisa. She is in the front row and skinnier. | K-means | 0.333 | 0.481 | 0.943 | Not Offensive |
| | GMM | 0 | 0.0 | 1 | |
| @USER Can we talk about how he is having the worst year of any player in yankee history? | K-means | 0.445 | 0.407 | 0.946 | Offensive |
| | GMM | 0 | 0 | 1 | |
| @USER @USER You are so funny. Lol rolling about the floor laughing uncontrollably! | K-means | 0.306 | 0.521 | 0.910 | Not Offensive |
| | GMM | 0 | 0 | 1 | |
| @USER Likely shift? He is a political operative for the GOP. This will be a life-long dream for many who have been working towards this outcome for decades. It's sad. | K-means | 0.318 | 0.494 | 0.941 | Offensive |
| | GMM | 0 | 0 | 1 | |

Table 10: SVNS values for some neutral samples generated using BiLSTM architecture.

| Model | Layer | Precision | Recall | F1 | Accuracy | Specificity | FDR | FPR | FNR | FOR | NPV |
|---|---|---|---|---|---|---|---|---|---|---|---|
| BiLSTM | GRU | .741 | .777 | .752 | .784 | .896 | .237 | .104 | .412 | .208 | .792 |
| | Dense-3 | .739 | .771 | .749 | .784 | .889 | .465 | .110 | .574 | .161 | .839 |
| | BatchNorm | .785 | .774 | .779 | .826 | .871 | .342 | .129 | .301 | .110 | .890 |
| BERT | PLM | .796 | .809 | .802 | .837 | .899 | .254 | .101 | .306 | .127 | .873 |
| | Dense-3 | .799 | .808 | .804 | .840 | .896 | .263 | .104 | .298 | .121 | .879 |
| | BatchNorm | .766 | .813 | .777 | .802 | .926 | .163 | .074 | .395 | .211 | .789 |
| MPNet | PLM | .789 | .818 | .800 | .830 | .913 | .208 | .087 | .336 | .155 | .845 |
| | Dense-3 | .787 | **.824** | .800 | .828 | .921 | .183 | .079 | .347 | .168 | .832 |
| | BatchNorm | .806 | **.824** | **.814** | .845 | .909 | .225 | .091 | .298 | .127 | .873 |
| ALBERT | PLM | .783 | .805 | .792 | .826 | .901 | .242 | .099 | .336 | .148 | .852 |
| | Dense-3 | .813 | .816 | .814 | .850 | .898 | .263 | .102 | .272 | .106 | .894 |
| | BatchNorm | .724 | .777 | .721 | .740 | **.929** | **.138** | **.071** | .480 | .308 | .692 |
| RoBERTa | PLM | .821 | .808 | **.814** | **.853** | .888 | .296 | .112 | .246 | .089 | .911 |
| | Dense-3 | **.823** | .804 | .813 | **.853** | .885 | .308 | .115 | **.239** | **.084** | **.916** |
| | BatchNorm | .775 | .813 | .788 | .816 | .915 | .196 | .085 | .365 | .179 | .821 |

Table 11: Performance of SVNS based predictions on OLID Test set.

functions where each membership function ranges from 0 to 1. This allows us to map features of any dimension and range to a standard format. Figure 9, Figure 10 and Figure 11 show the SVNS calculated from different layers over different dimensions quantified in terms of SVNS membership functions. We can see from the plots that all values are distributed over a similar range and have a similar distribution. Once we obtain the SVNS values, it becomes effortless to compare features from different models, dimensions, and distribution since they are represented in terms of standard neutrosophic sets.

**Clustering results on OLID test set**: For classifying samples as offensive/non-offensive using SVNS, we take the maximum of the offensive/non-offensive membership function and assign the sentiment corresponding to that membership function to the sample. We used SVNS values generated using k-means for evaluation on the test set. The cluster centres learnt on the train set in Section 6.4 are used along with feature vectors on the test set for SVNS calculation. The results are shown in Table 11. We observe that different layers from each model had similar performance on the test set. Comparing the results of our classification model and SVNS model in Table 8 and Table 11 we can see that SVNS models perform as good as the original classification models. Thus, we can conclude that SVNS perform equivalent to deep learning models and at the same time allow us to define a sentiment in terms of membership functions instead of simply labelling the output as a single sentiment. It also helps us capture the indeterminacy in prediction in terms of $\text{SVNS}_{\text{neutral}}$ membership function.

## 8. Comparison with other models

OLID was the official dataset for SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval) (Zampieri et al., 2019b). We compare the results of our SVNS model to the top five teams from OffensEval. NULI (Liu et al., 2019a) experimented with linear, LSTM, and BERT based models. Their best performing model was fine-tuned using BERT over three epochs. Vradivchev Anikolov (Nikolov and Radivchev, 2019) experimented using different models like SVM, CNN, RNN. Their best performing model was fine-tuned using BERT-large. They also experimented with different thresholds to obtain the best results. UM-IU@LING (Zhu et al., 2019) also experimented using BERT to obtain their results. They also experimented with cased and uncased version of BERT. The uncased model outperformed the cased model. Embeddia (Pelicon et al., 2019) also used BERT-large and fine-tuned it on the OLID dataset. MIDAS (Mahata et al., 2019) experimented with CNN, BiLSTM with attention, and GRU and their best performing model were obtained by ensembling their individual models. The results are tabulated in Table 12.

We also compare the performance of our model with other benchmarks on the dataset. OffensEval (Zampieri et al., 2019a) contains the benchmarks for data using different models. We compared with those benchmarks to evaluate our models performance. We also compared the performance of our model with the offensive category benchmark in TweetEval (Barbieri et al., 2020). It is a unified benchmark for the comparative evaluation of different datasets available for Twitter. It consists of the OffensEval dataset as the benchmark of offensive language detection on Twitter. The TweetEval used three variants of RoBERTa: RoBERTa-Bs, the RoBERTa-base pre-trained model, RoBERTa-RT, RoBERTa-base model retrained on Twitter data, and RoBERTa-TW, trained from scratch on the Twitter dataset. Comparison is shown in Table 13. Our classification,

| Rank No. | System | F1 (macro) |
|---|---|---|
| 1 | NULI | **.829** |
| 2 | vradivchev anikolov | .815 |
| 3 | UM-IU@LING | .814 |
| 4 | Embeddia | .808 |
| 5 | MIDAS | .807 |
| Proposed (Classification) | BiLSTM | .776 |
| | BERT | .805 |
| | MPNet | .808 |
| | ALBERT | .813 |
| | RoBERTa | **.816** |
| Proposed (SVNS) | BiLSTM (BatchNorm) | .779 |
| | BERT (Dense-3) | .804 |
| | MPNet (BatchNorm) | **.814** |
| | ALBERT (Dense-3) | **.814** |
| | RoBERTa (PLM) | **.814** |

Table 12: Comparison of results against top five systems in OffensEval.

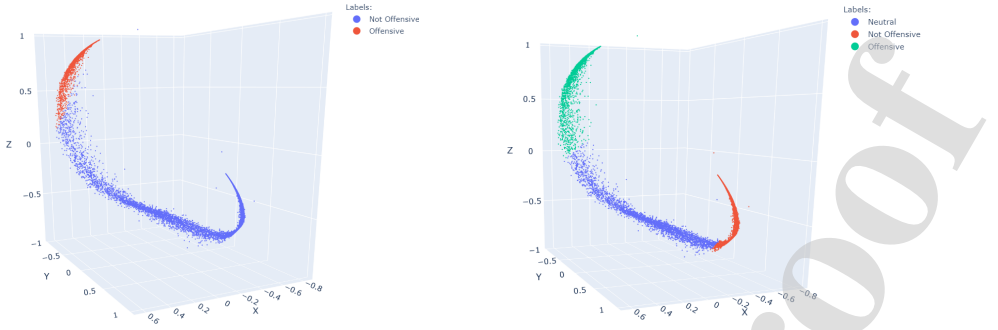
as well as SVNS based models, perform equivalent to these benchmarks. Our standard RoBERTa-Base model outperforms the RoBERTa-Bs and RoBERTa-RT models by a significant margin and perform equivalent to RoBERTa-TW model, which was first pre-trained and then fine-tuned on the given dataset.

### *8.1. Identifying neutralities given a classification problem*

Neural networks are highly efficient in solving complex language problems like machine translation, classification, sentiment analysis, and feature extraction. One drawback we usually encounter during classification problems is that although neural networks are satisfactory in predicting the correct class, they cannot quantify the output over other

| Model | | F1 (macro) |
|---|---|---|
| OffensEval (Baselines) | SVM | .690 |
| | LSTM | .750 |
| | CNN | **.800** |
| TweetEval | RoBERTa-Bs | .787 |
| | RoBERTa-RT | **.816** |
| | RoBERTa-TW | .786 |
| Proposed (Classification) | RoBERTa | **.816** |
| Proposed (SVNS) | RoBERTa (PLM) | **.814** |

Table 13: Comparison with benchmarks.




(a) Features labelled using neural network.

(b) Clustering features to capture the transition phase.

Figure 12: Transition of features from offensive to non-offensive.

prediction classes. Loss functions used in training train the neural network to maximise the correct class probability, and the probabilities of wrong classes are minimised to minute values. Due to this, we are unable to capture the transition phase of classes from one to another. This transition phase represents the neutralities which contains components from both the positive and negative ones, and it is not easy for the classifier to predict the correct class for these transition samples. Another remarkable aspect of neural networks is that their middle layers are utilised as feature extractors. These features represent the characteristics of different classes and can be used to separate them. Figure 12 shows the features of the three neuron layers learnt by our BiLSTM architecture. Figure 12a is labelled using the output of our neural network, while Figure 12b shows the same features when formed into 3 clusters.

Figure 12a shows the transition phase from non-offensive to offensive tweets, which usually goes unnoticed in standard neural network architectures. This area contains the tweets on the borderline of both the given classes and represents the neutralities. Figure 12b shows the same feature set upon clustering into 3 clusters. From Figure 12b, we can see that clustering shows promising results in separating the transition phase. We use these clusters to calculate the SVNS values, which quantify the tweets into three membership functions (positive, negative, neutral). We can use the neutral membership function of SVNS to separate the neutral samples. Thus, instead of simply predicting the output class, we now have quantified values for each tweet, which help better analyse the data.

*Application based use case scenario*: Data labelling is not an easy task and can be expensive at times, and it makes collecting large datasets a difficult task. Neural networks depend on large datasets for training. Moreover, during labelling, the labels become biased due to different opinions of the annotators. Consider, for example: If we want to get the dataset labelled into positive, negative and neutral classes, the dataset can become highly biased because it is exceedingly difficult to differentiate between the positive and neutral classes. Also, the problem due to different persons annotating the dataset can disturb the distribution of the data. This will add a bias in the classifier, reducing the efficiency. On the other hand, if we only label the dataset into positive and negative classes, the task becomes less complex and easy. However, assume we want to separate the neutral component to obtain purely positive and negative samples. For

example, consider running a social media campaign for our product where users can share their experiences about our product. We want to obtain the drawbacks for improving the product and positive reviews for promoting the product. We want to discard the neutral (transition phase) samples. Training a classifier with a positive, negative, neutral class can be expensive and inefficient due to the reasons mentioned above. Training a positive, negative classifier is easy and feasible, but the problem will be removing the thousands of neutral (transition phase) tweets. The proposed model is efficient in such cases where we identify the neutral component given only positive and negative classes. The model will be cheap and, at the same time, will be capable of quantifying the tweet into a three-value membership function from where we can easily segregate the required tweets.

## 9. Conclusions

We proposed a novel framework that used the essence of neutrosophy as SVNS to detect neutralities present in tweets by using deep learning systems. We experimented with state-of-the-art language models to learn features for determining the SVNS values. We generated SVNS sets using five various models: BiLSTM using GloVe, BERT, ALBERT, RoBERTa and MPNet.

SemEval 2019 Task 6 (Subtask A) dataset / OLID dataset was used for experimental analysis. Our proposed model is a unique and first of its kind framework for quantifying tweets into SVNS values. Also, using the neutral membership function of SVNS, we determine the neutralities in the tweets. Most datasets deal with only positive and negative samples and forget about the neutral part. Our model helps in those cases to understand neutralities and use SVNS values to quantify the data better. We used state-of-the-art language models for identifying the transition phase between positive and negative samples and used clustering to quantify them in terms of SVNS values. Our SVNS model performed equivalent to the state-of-the-art models and successfully captured the sentiment in terms of SVNS membership functions.

In future, we would like the explore the use of Double Valued Neutrosophic Sets (DVNS) for classification problems in natural language. DVNS uses four membership functions to define the uncertainty in samples. It uses a truth membership function $T_A$, indeterminacy leaning towards truth membership function $I_{TA}$, indeterminacy leaning towards falsity membership function $I_{FA}$, and falsity membership function $F_A$. It will help us to understand better the neutralities and their inclination towards either positive/negative samples. Bipolar neutrosophic sets use truth, indeterminacy, and false membership functions for both positive and negative labels. These are used to study positive/negative samples independently in terms of membership functions. We would also like to introduce the concept of neutrosophy in multi-label classification problems in future. It will help us explore the truth, indeterminate, and false membership functions for each label independently. Neutrosophic refined sets split the truth, indeterminate and false membership functions to n components. This can be easily used to formulate multi-label classification problems in terms of neutrosophy. Neutrosophy is a powerful tool and has been used to measure indeterminacies in terms of membership functions. Neural networks are capable of learning complex relations in natural language. Neutrosophy combined with neural networks can be a powerful tool in NLP and understanding natural language.

## References


Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. URL: https://www.tensorflow.org/. software available from tensorflow.org.

Alghamdi, A., Hammad, M., Ugail, H., Abdel-Raheem, A., Muhammad, K., Khalifa, H.S., Abd El-Latif, A.A., 2020a. Detection of myocardial infarction based on novel deep transfer learning methods for urban healthcare in smart cities. Multimedia Tools and Applications URL: https://doi.org/10.1007/s11042-020-08769-x, doi:10.1007/s11042-020-08769-x.

Alghamdi, A.S., Polat, K., Alghoson, A., Alshdadi, A.A., Abd El-Latif, A.A., 2020b. A novel blood pressure estimation method based on the classification of oscillometric waveforms using machine-learning methods. Applied Acoustics 164, 107279. URL: http://www.sciencedirect.com/science/article/pii/S0003682X20300013, doi:https://doi.org/10.1016/j.apacoust.2020.107279.

Ali, M., Son, L.H., Deli, I., Tien, N.D., 2017. Bipolar neutrosophic soft sets and applications in decision making. Journal of Intelligent & Fuzzy Systems 33, 4077–4087. URL: https://doi.org/10.3233/JIFS-17999, doi:10.3233/JIFS-17999. 6.

Alyafeai, Z., AlShaibani, M.S., Ahmad, I., 2020. A survey on transfer learning in natural language processing. arXiv:2007.04239.

Arthur, D., Vassilvitskii, S., 2007. K-means++: The advantages of careful seeding, in: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, USA. p. 1027–1035.

Badjatiya, P., Gupta, S., Gupta, M., Varma, V., 2017. Deep learning for hate speech detection in tweets. Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion URL: http://dx.doi.org/10.1145/3041021.3054223, doi:10.1145/3041021.3054223.

Bahdanau, D., Cho, K., Bengio, Y., 2015. Neural machine translation by jointly learning to align and translate, in: Bengio, Y., LeCun, Y. (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. URL: http://arxiv.org/abs/1409.0473.

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

Baziotis, C., Pelekis, N., Doulkeridis, C., 2017. 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. pp. 747–754. URL: https://www.aclweb.org/anthology/S17-2126, doi:10.18653/v1/S17-2126.

Bengio, Y., Simard, P.Y., Frasconi, P., 1994. Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks 5 2, 157–66.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg.

Broumi, S., Talea, M., Bakali, A., Smarandache, F., 2016. Single valued neutrosophic graphs. Journal of New Theory 10, 86–101.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. 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. pp. 1724–1734. URL: https://www.aclweb.org/anthology/D14-1179, doi:10.3115/v1/D14-1179.

Clarke, I., Grieve, J., 2017. Dimensions of abusive language on twitter, in: Proceedings of the First Workshop on Abusive Language Online, Association for Computational Linguistics, Vancouver, BC, Canada. pp. 1–10. URL: https://www.aclweb.org/anthology/W17-3001, doi:10.18653/v1/W17-3001.

Davidson, T., Warmsley, D., Macy, M.W., Weber, I., 2017. Automated hate speech detection and the problem of offensive language, in: Proceedings of the Eleventh International Conference on Web and

Social Media, ICWSM 2017, Montréal, Québec, Canada, May 15-18, 2017, AAAI Press. pp. 512–515. URL: https://aaai.org/ocs/index.php/ICWSM/ICWSM17/paper/view/15665.

Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2019. 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 and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota. pp. 4171–4186. URL: https://www.aclweb.org/anthology/N19-1423, doi:10.18653/v1/N19-1423.

Ethayarajh, K., 2019. Rotate king to get queen: Word relationships as orthogonal transformations in embedding space, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China. pp. 3503–3508. URL: https://www.aclweb.org/anthology/D19-1354, doi:10.18653/v1/D19-1354.

Forney, G.D., 1973. The viterbi algorithm. Proceedings of the IEEE 61, 268–278. doi:10.1109/PROC.1973.9030.

Fortuna, P., Nunes, S., 2018. A survey on automatic detection of hate speech in text. ACM Comput. Surv. 51. URL: https://doi.org/10.1145/3232676, doi:10.1145/3232676.

Gambäck, B., Sikdar, U.K., 2017. Using convolutional neural networks to classify hate-speech, in: Proceedings of the First Workshop on Abusive Language Online, Association for Computational Linguistics, Vancouver, BC, Canada. pp. 85–90. URL: https://www.aclweb.org/anthology/W17-3013, doi:10.18653/v1/W17-3013.

Gatys, L.A., Ecker, A.S., Bethge, M., 2016. Image style transfer using convolutional neural networks, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2414–2423. doi:10.1109/CVPR.2016.265.

Hassan, A., Mahmood, A., 2017. Deep learning approach for sentiment analysis of short texts, in: 2017 3rd International Conference on Control, Automation and Robotics (ICCAR), pp. 705–710.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9, 1735–1780. URL: https://doi.org/10.1162/neco.1997.9.8.1735, doi:10.1162/neco.1997.9.8.1735.

Holgate, E., Cachola, I., Preoţiuc-Pietro, D., Li, J.J., 2018. Why swear? analyzing and inferring the intentions of vulgar expressions, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Brussels, Belgium. pp. 4405–4414. URL: https://www.aclweb.org/anthology/D18-1471, doi:10.18653/v1/D18-1471.

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

Kandasamy, I., Kandasamy, W.V., Obbineni, J.M., Smarandache, F., 2020a. Indeterminate likert scale: feedback based on neutrosophy, its distance measures and clustering algorithm. Soft Computing 24, 7459–7468.

Kandasamy, I., Smarandache, F., 2016. Triple refined indeterminate neutrosophic sets for personality classification, in: Computational Intelligence (SSCI), 2016 IEEE Symposium Series on, IEEE. pp. 1–8. doi:10.1109/SSCI.2016.7850153.

Kandasamy, I., Vasantha, W., Mathur, N., Bisht, M., Smarandache, F., 2020b. Sentiment analysis of the# metoo movement using neutrosophy: Application of single-valued neutrosophic sets, in: Optimization Theory Based on Neutrosophic and Plithogenic Sets. Elsevier, pp. 117–135.

Kandasamy, I., Vasantha, W., Obbineni, J.M., Smarandache, F., 2020c. Sentiment analysis of tweets using refined neutrosophic sets. Computers in Industry 115, 103180.

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

Kudo, T., Richardson, J., 2018. 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. pp. 66–71. URL: https://www.aclweb.org/anthology/D18-2012, doi:10.18653/v1/D18-2012.

Kumar, R., Ojha, A.K., Malmasi, S., Zampieri, M., 2018. Benchmarking aggression identification in social media, in: Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018), Association for Computational Linguistics, Santa Fe, New Mexico, USA. pp. 1–11. URL: https://www.aclweb.org/anthology/W18-4401.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R., 2020. Albert: A lite bert

for self-supervised learning of language representations, in: International Conference on Learning Representations. URL: https://openreview.net/forum?id=H1eA7AEtvS.

Li, D., Qian, J., 2016. Text sentiment analysis based on long short-term memory, in: 2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI), pp. 471–475.

Liu, P., Li, W., Zou, L., 2019a. NULI at SemEval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers, in: Proceedings of the 13th International Workshop on Semantic Evaluation, Association for Computational Linguistics, Minneapolis, Minnesota, USA. pp. 87–91. URL: https://www.aclweb.org/anthology/S19-2011, doi:10.18653/v1/S19-2011.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V., 2019b. Roberta: A robustly optimized bert pretraining approach. arXiv:1907.11692.

Luong, T., Pham, H., Manning, C.D., 2015. Effective approaches to attention-based neural machine translation, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal. pp. 1412–1421. URL: https://www.aclweb.org/anthology/D15-1166, doi:10.18653/v1/D15-1166.

Mahata, D., Zhang, H., Uppal, K., Kumar, Y., Shah, R.R., Shahid, S., Mehnaz, L., Anand, S., 2019. MIDAS at SemEval-2019 task 6: Identifying offensive posts and targeted offense from Twitter, in: Proceedings of the 13th International Workshop on Semantic Evaluation, Association for Computational Linguistics, Minneapolis, Minnesota, USA. pp. 683–690. URL: https://www.aclweb.org/anthology/S19-2122, doi:10.18653/v1/S19-2122.

Mathur, P., Shah, R., Sawhney, R., Mahata, D., 2018. Detecting offensive tweets in Hindi-English code-switched language, in: Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media, Association for Computational Linguistics, Melbourne, Australia. pp. 18–26. URL: https://www.aclweb.org/anthology/W18-3504, doi:10.18653/v1/W18-3504.

Mishra, K., Kandasamy, I., Kandasamy WB, V., Smarandache, F., 2020. A novel framework using neutrosophy for integrated speech and text sentiment analysis. Symmetry 12, 1715.

Nikolov, A., Radivchev, V., 2019. Nikolov-radivchev at SemEval-2019 task 6: Offensive tweet classification with BERT and ensembles, in: Proceedings of the 13th International Workshop on Semantic Evaluation, Association for Computational Linguistics, Minneapolis, Minnesota, USA. pp. 691–695. URL: https://www.aclweb.org/anthology/S19-2123, doi:10.18653/v1/S19-2123.

Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y., Chang, Y., 2016. Abusive language detection in online user content, in: Proceedings of the 25th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE. p. 145–153. URL: https://doi.org/10.1145/2872427.2883062, doi:10.1145/2872427.2883062.

Nwankpa, C., Ijomah, W., Gachagan, A., Marshall, S., 2018. Activation functions: Comparison of trends in practice and research for deep learning. CoRR abs/1811.03378. URL: http://arxiv.org/abs/1811.03378, arXiv:1811.03378.

Park, J.H., Fung, P., 2017. One-step and two-step classification for abusive language detection on Twitter, in: Proceedings of the First Workshop on Abusive Language Online, Association for Computational Linguistics, Vancouver, BC, Canada. pp. 41–45. URL: https://www.aclweb.org/anthology/W17-3006, doi:10.18653/v1/W17-3006.

Pelicon, A., Martinc, M., Kralj Novak, P., 2019. Embeddia at SemEval-2019 task 6: Detecting hate with neural network and transfer learning approaches, in: Proceedings of the 13th International Workshop on Semantic Evaluation, Association for Computational Linguistics, Minneapolis, Minnesota, USA. pp. 604–610. URL: https://www.aclweb.org/anthology/S19-2108, doi:10.18653/v1/S19-2108.

Pennington, J., Socher, R., Manning, C., 2014. 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. pp. 1532–1543. URL: https://www.aclweb.org/anthology/D14-1162, doi:10.3115/v1/D14-1162.

Pitsilis, G.K., Ramampiaro, H., Langseth, H., 2018. Effective hate-speech detection in twitter data using recurrent neural networks. Applied Intelligence 48, 4730–4742. URL: http://dx.doi.org/10.1007/s10489-018-1242-y, doi:10.1007/s10489-018-1242-y.

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., Huang, X., 2020. Pre-trained models for natural language processing: A survey. Science China Technological Sciences 63, 1872–1897. URL: https://doi.org/10.1007/s11431-020-1647-3, doi:10.1007/s11431-020-1647-3.

Reforgiato Recupero, D., Alam, M., Buscaldi, D., Grezka, A., Tavazoee, F., 2019. Frame-based detection of figurative language in tweets [application notes]. IEEE Computational Intelligence Magazine 14, 77–88. doi:10.1109/MCI.2019.2937614.

Sammut, C., Webb, G.I., 2011. Encyclopedia of Machine Learning. 1st ed., Springer Publishing Company, Incorporated.

Schmidt, A., Wiegand, M., 2017. A survey on hate speech detection using natural language processing, in: Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media, Association for Computational Linguistics, Valencia, Spain. pp. 1–10. URL: https://www.aclweb.org/anthology/W17-1101, doi:10.18653/v1/W17-1101.

Sedik, A., Iliyasu, A.M., Abd El-Rahiem, B., Abdel Samea, M.E., Abdel-Raheem, A., Hammad, M., Peng, J., Abd El-Samie, F.E., Abd El-Latif, A.A., 2020. Deploying machine and deep learning models for efficient data-augmented detection of covid-19 infections. Viruses 12, 769. URL: http://dx.doi.org/10.3390/v12070769, doi:10.3390/v12070769.

Sennrich, R., Haddow, B., Birch, A., 2016. 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. pp. 1715–1725. URL: https://www.aclweb.org/anthology/P16-1162, doi:10.18653/v1/P16-1162.

Sharma, C., Bhageria, D., Scott, W., PYKL, S., Das, A., Chakraborty, T., Pulabaigari, V., Gambäck, B., 2020a. SemEval-2020 task 8: Memotion analysis- the visuo-lingual metaphor!, in: Proceedings of the Fourteenth Workshop on Semantic Evaluation, International Committee for Computational Linguistics, Barcelona (online). pp. 759–773. URL: https://www.aclweb.org/anthology/2020.semeval-1.99.

Sharma, M., Kandasamy, I., Vasantha, W., 2020b. Memebusters at SemEval-2020 task 8: Feature fusion model for sentiment analysis on memes using transfer learning, in: Proceedings of the Fourteenth Workshop on Semantic Evaluation, International Committee for Computational Linguistics, Barcelona (online). pp. 1163–1171. URL: https://www.aclweb.org/anthology/2020.semeval-1.154.

Smarandache, F., 1999. A unifying field in logics: Neutrosophic logic. neutrosophy, neutrosophic set, neutrosophic probability: Neutrosophic logic: neutrosophy, neutrosophic set, neutrosophic probability. American Research Press.

Smarandache, F., 2000. A Unifying Field in Logics: Neutrosophic Logic. Neutrosophy, Neutrosophic Set, Probability, and Statistics. American Research Press, Rehoboth. URL: https://arxiv.org/pdf/math/0101228.

Song, K., Tan, X., Qin, T., Lu, J., Liu, T.Y., 2020. Mpnet: Masked and permuted pre-training for language understanding, in: NeurIPS 2020, ACM. URL: https://www.microsoft.com/en-us/research/publication/mpnet-masked-and-permuted-pre-training-for-language-understanding/.

Sood, S.O., Churchill, E.F., Antin, J., 2012. Automatic identification of personal insults on social news sites. J. Am. Soc. Inf. Sci. Technol. 63, 270–285. URL: https://doi.org/10.1002/asi.21690, doi:10.1002/asi.21690.

Spertus, E., 1997. Smokey: Automatic recognition of hostile messages, in: Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence, AAAI Press. p. 1058–1065.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15, 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C., 2018. A survey on deep transfer learning, in: Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., Maglogiannis, I. (Eds.), Artificial Neural Networks and Machine Learning – ICANN 2018, Springer International Publishing, Cham. pp. 270–279.

Vasantha, W., Kandasamy, I., Smarandache, F., Devvrat, V., Ghildiyal, S., 2020. Study of imaginative play in children using single-valued refined neutrosophic sets. Symmetry 12, 402.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I., 2017. Attention is all you need, in: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wang, H., Smarandache, F., Zhang, Y., Sunderraman, R., 2010. Single valued neutrosophic sets. Review 16, 10–14.

Waseem, Z., Davidson, T., Warmsley, D., Weber, I., 2017. Understanding abuse: A typology of abusive language detection subtasks, in: Proceedings of the First Workshop on Abusive Language Online, Association for Computational Linguistics, Vancouver, BC, Canada. pp. 78–84. URL: https://www.aclweb.org/anthology/W17-3012, doi:10.18653/v1/W17-3012.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M., 2020. Transformers: State-of-the-

art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Online. pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Woźniak, M., Wieczorek, M., Siłka, J., Połap, D., 2021. Body pose prediction based on motion sensor data and recurrent neural network. IEEE Transactions on Industrial Informatics 17, 2101–2111. doi:10.1109/TII.2020.3015934.

Wozniak, M., Silka, J., Wieczorek, M., Alrashoud, M., 2020. Recurrent neural network model for iot and networking malware threads detection. IEEE Transactions on Industrial Informatics , 1–1doi:10.1109/TII.2020.3021689.

Wu, Y., Liu, Y., Ahmed, S.H., Peng, J., Abd El-Latif, A.A., 2020. Dominant data set selection algorithms for electricity consumption time-series data analysis based on affine transformation. IEEE Internet of Things Journal 7, 4347–4360. doi:10.1109/JIOT.2019.2946753.

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

Xiang, G., Fan, B., Wang, L., Hong, J., Rose, C., 2012. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus, in: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, Association for Computing Machinery, New York, NY, USA. p. 1980–1984. URL: https://doi.org/10.1145/2396761.2398556, doi:10.1145/2396761.2398556.

Yin, D., Hu, Y., Tang, J., Daly, T., Zhou, M., Ouyang, H., Chen, J., Kang, C., Deng, H., Nobata, C., Langlois, J.M., Chang, Y., 2016. Ranking relevance in yahoo search, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, NY, USA. p. 323–332. URL: https://doi.org/10.1145/2939672.2939677, doi:10.1145/2939672.2939677.

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., Kumar, R., 2019a. Predicting the type and target of offensive posts in social media, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota. pp. 1415–1420. URL: https://www.aclweb.org/anthology/N19-1144, doi:10.18653/v1/N19-1144.

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., Kumar, R., 2019b. SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval), in: Proceedings of the 13th International Workshop on Semantic Evaluation, Association for Computational Linguistics, Minneapolis, Minnesota, USA. pp. 75–86. URL: https://www.aclweb.org/anthology/S19-2010, doi:10.18653/v1/S19-2010.

Zhu, J., Tian, Z., Kübler, S., 2019. UM-IU@LING at SemEval-2019 task 6: Identifying offensive tweets using BERT and SVMs, in: Proceedings of the 13th International Workshop on Semantic Evaluation, Association for Computational Linguistics, Minneapolis, Minnesota, USA. pp. 788–795. URL: https://www.aclweb.org/anthology/S19-2138, doi:10.18653/v1/S19-2138.

# HIGHLIGHTS

1. Sentiment analysis model using deep learning, neutrosophy, and transfer learning.
2. Analysing tweets as a combination of sentiments identifying neutralities.
3. Quantifying tweets into Single Valued Neutrosophic Sets (SVNS) for OLID dataset.
4. Experimental analysis using BiLSTM, BERT, RoBERTa, ALBERT, and MPNet.
5. Gaussian Mixture Model and k-means clustering algorithm for SVNS calculation

**Mayukh Sharma:** Visualization, Investigation, Methodology, Software **Ilanthenral Kandasamy**: Formal analysis, Data curation, Writing- Original draft preparation. **Vasantha Kandasamy***:* Conceptualization, Supervision, Validation, Writing- Reviewing and Editing

**Declaration of interests**

☐ 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.