Neural Computing and Applications (2023) 35:19465–19479
https://doi.org/10.1007/s00521-023-08767-8

ORIGINAL ARTICLE

Real-time traffic, accident, and potholes detection by deep learning techniques: a modern approach for traffic management

Sarthak Babbar[1] · Jatin Bedi[1]

Received: 17 September 2022 / Accepted: 12 June 2023 / Published online: 28 June 2023

Abstract
The practical applications of social media have raised the bar for real-time event detection all over the globe. It has been deemed useful for extracting important data that enables people across the world to access information within mere seconds of its occurrence. Among the many use cases of the aforementioned applications lies detection of issues regarding the smooth mobility of traffic on the road. Machine learning models that were developed earlier used support vector machines with Bag of Words. However, BoW (Bag of Words) suffers from problems such as the inability to manage semantic relationships between words, limited representation, and high dimensional representation. Our model developed for a similar cause fetches the required data from Twitter to make the target authorities aware of the issues like potholes, accidents, and high traffic density (congestion) of the Chandigarh tri-city area. The data collected then undergoes multiple stages of pre-processing. After that, multiple word embedding models come into play to build semantic relationships between the words and intra software jargon. The resultant is then processed through the several recently introduced deep learning based natural language processing. The current study aims to perform a comparative evaluation of the several advanced state-of-the-art classification models at the target traffic event detection activity. The developed models make a fact-based prediction to make multi-class classification into the aforementioned categories. From the comparative evaluation, it has been observed that the proposed language processing model (RoBERTa based) pipeline outperforms the existing approaches and is 97% accurate at classifying the real-time tweets. Moreover, the proposed pipeline achieves 96% recall at segregating the traffic events efficiently.

Keywords Traffic events · Deep learning · Intelligent transportation · RoBERTa · Accidents

1 Introduction

Detecting traffic activities and places is crucial for building a robust and sustainable system to manage transportation and higher urban policy-making activities. It is related to several critical aspects of the society such as sustainability, delivery, development, economic activities, and many more. Traffic events can be associated with various different activities such as traffic accidents, parking problems, traffic jams, potholes, and more. Currently, traffic events are being detected by hardware components like sensors (e.g., cameras) [1–3]. Previously, enormous amount of efforts have been performed by researchers to accurately identify traffic-related issues by formulating several algorithms using physical sensors to spot the event in real-time. However, these existing studies are characterized by restricted spatial coverage and high upkeep fee, particularly in developing regions [4, 5]. Alternatively, with web 2.0 and other popular mobile platforms, human beings can act as social sensors who share special traffic events from within their respective places. In the past decade, social media networks have obtained an awful amount of attention among ordinary people, corporations, and research pupils [6–8]. Many systems and platforms have been conceived to collect big datasets to represent the behavior and characteristics of different phenomena in varying contexts. In this study, we focus on exploring the reliability of micro-blogging sites (Facebook or Twitter) toward

✉ Jatin Bedi
jatin.bedi@thapar.edu; jbedi@cs.iitr.ac.in

1 Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala, Punjab 147001, India

developing a robust and sustainable system to help the society.

Twitter is one of the fastest-growing social media platforms that enable users to disseminate and study brief messages, referred to as tweets. India is ranked among the first three countries with the highest number of Twitter users 2022 [9]. Using Twitter applications on smartphones, customers can report events happening around them in real-time. Daily, the records disseminated through thousands of active users generate a new version of the dynamic database containing facts about various topics. In recent years, the dataset generated through Twitter API has been widely adopted to resolve problems pertaining to different application domains [7, 8, 10, 11]. The current study uses Twitter to collect data about several transportation-related events in this context. Twitter has built up a reputation as a valuable source of information for identifying and detecting traffic events [12, 13]. Thus, we have crawled tweets from the Chandigarh Tri-city region as a case study.

Henceforth, this report proposes a methodology to extract tweets using Tweepy API, save those tweets as an SQL-lite file, pre-process, and sort those extracted tweets. These tweets are then examined to crawl non-recurrent events facts using deep learning and Natural Language Processing (NLP) techniques. The paper also proposes an approach using NER (Named Entity Recognition) and POS (Part of speech) to identify the information regarding the location from the textual content [14, 15]. The primary research contributions of the research study are listed as follows:

- The result and fidelity of social media in apprehending human behavior on multiple common issues have been well established [12, 13] in recent years. The research utilized Twitter data for the multi-class transportation-related event classification task. Moreover, the location extraction of the identified incidents is performed using Named Entity Recognition (NER) and Part-of-Speech (POS) tagging.
- Identifying transportation-related complaints is critical for the smooth execution of several related activities, such as optimized transportation, reducing traffic accidents, resource usage optimization, and many more. The current work provides a standardized dataset and an end-to-end pipeline (tweets collection, pre-processing, location extraction, and event identification) to identify/classify traffic, accidents, and potholes-related tweets.
- A detailed analysis (from the perspective of implementation and comparison) of several state-of-the-art word embeddings [16] in integration with deep learning models is provided for a better understanding of the problem under study. The comparative evaluation results of the proposed work with these deep learning models show that the proposed approach outperforms existing approaches by generating the least prediction error.
- The applications of recently introduced natural language processing models (BERT: Bidirectional Encoder Representations from Transformers [17] and XLNET [18]) on the target application domain (transportation) are well explored in the current study. Furthermore, a RoBERTa (Robustly Optimized BERT Pre-training Approach) [19]-based approach is proposed to identify/classify the real-time traffic, potholes, and accidents related events with 97% accuracy, which is higher than any other existing benchmark techniques in the transportation domain.

2 Literature review

Traffic event detection has always been a hot research topic for the community as it can directly impact society and related factors such as air pollution, accidents, hotspots, etc. [20–22]. In recent years, the role of Twitter in the detection of traffic events has gained wide popularity. We perform an exhaustive analysis of research studies targeted at using Twitter for intelligent traffic management activities in the current section.

Andrea et al. [6] proposed a machine learning-based approach for detecting traffic events (either related to traffic or not) from real-time tweets. The authors implemented Support Vector Regression for the events classification task (binary classification). The experimental results evaluated on the Italian dataset show that the model achieved 95% classification accuracy in identifying traffic events. Ribeiro et al. [23] developed a methodology to analyze the traffic conditions of a particular place. The approach identifies the traffic conditions and geocodes the situation to visualize traffic scenarios to the users. The approach achieved satisfactory results with precision varying between 50% and 90%.

Yiming Gu et al. [24] introduced an approach to collect, pre-process, and identify traffic incidents from the users' tweets. From the experimental results, the authors concluded that incidents occurrence frequency is higher on weekends than weekdays and is directly related to the crowd density. Dabiri and Heaslip [25] developed a deep learning-based approach to detect or classify traffic events from the real-time streaming dataset. The classification approach involved implementing two neural models (Convolutional Neural Networks and Recurrent Neural Networks) for identifying three classes of tweets, i.e., traffic incident, non-traffic, and traffic information. The experimental results validated the effectiveness of the approach.

Albuquerque et al. [26] developed a tool to detect traffic-related incidents by interpreting users' tweets. The approach utilized two kinds of tweets (a) users posted tweets, (b) government and news agencies posted tweets. The approach includes deploying machine learning techniques for the tweets interpretation tasks. Traffic flow estimation has always been a critical topic for several traffic management activities such as congestion, road closures and accidents. Essien et al. [27] developed an approach to integrate traffic flow data with twitter information for the short-term traffic patterns estimation task. The authors implemented a deep neural architecture, i.e., a Bi-directional Long Short-Term Memory network with auto-encoders for the prediction task. Alomari et al. [28] proposed iktishaf, a Spark-based tool to analyze traffic data collected from Twitter API. The tool utilizes three machine learning algorithms (Naïve Bayes, Support Vector and Logistic Regression) to classify the eight-event types (road, fire, traffic, accident, closure, roadwork, weather and social events). The tool assessment and validation are carried out on several real-time events in Saudi Arabia. Agarwal et al. [29] utilized Twitter data to identify and classify citizens' complaints on several traffic-related issues such as bad road conditions, discomfort and bad road experiences. The proposed approach obtained 67% accuracy at classifying or identifying the killer roads. Chaturvedi et al. [30] collected Twitter data to analyze citizens' opinions on several transportation-related issues. The authors implement a dictionary-based approach (context-free) for the desired sentiment analysis task.

Rojas et al. [31] proposed a way to analyze Spanish traffic-related tweets. The approach includes implementing several modules: data preprocessing, location identification, and classification module. Yao et al. [32] proposed a novel application by estimating the one day ahead traffic density from the Twitter data of the previous day. The proposed model aims to determine traffic characteristics for better traffic management practices effectively. Sveta Milusheva [33] proposed a low-cost approach to effectively utilize Twitter data for urban planning activities. The approach involved implementing spatial clustering to identify crash activities and their precise location for better help and management activities.

Azhar et al. [34] proposed a deep learning-based approach to predict traffic accidents. The approach integrated tweets text with additional information such as sentiment, weather, and geocoded for accidents detection. The performance assessment on the real-time dataset shows that the approach achieved 94% accuracy at the desired task, and it outperforms the existing techniques. Deb et al. [35] performed a comparative analysis of different contextual and context-free embeddings. The approach signifies the importance and performance of contextual embeddings like BERT (Bidirectional Encoder Representations from Transformers) in the natural language processing task. Also, it involved discussing various challenges and opportunities associated with contextual embeddings.

3 Methodology

In the present study, an integrated model has been developed to detect traffic events and locations, which consist of four major phases mentioned below and the corresponding diagramatic representation is shown in Fig. 1.

- Using Twitter Streaming API for Collection of real-time tweets.
- Pre-processing of tweets and location Extraction
- Identifying the type of word-embeddings and Deep learning models to be used
- Implementing Deep Learning models for the target Classification Task and Evaluating Experimental Results

3.1 Collection of tweets

Twitter is a giant blogging application and social networking site where users post and share messages known as "tweets". For accessing Twitter, Tweepy (A reliable and easy-to-use library that can be used to access Twitter API in python language) is used with geocoding "30.7333,76.7794, 8km" (i.e., Chandigarh tri-city area with radius 8 km). People use government handles to report agencies about the several transportation-related issues faced in their daily lives. The relevant hashtags for the data collection on the transportation-related issues/events have been identified using the trending Tweets posts. Moreover, the handles/hashtags related to the government agencies (Chandigarh road safety, awareness, and traffic management) have been used to collect the data. The keyword/hashtags used in the current study to collect dataset are (["accident", "caraccident", "trafficchd", "parkingspace", "potholes", "roadsafetyawareness", "roadsafetyawareness", "ssptfcchd", "crash", "traffic", "personalinjury", "roadjam", "trafficjam", "trafficlights", "roadblock", "chandigarhtraffic", "chandigarhaccidents", "chandigarhaccidents", "rashdriving", "chandigarhtrafficlife", "baddriver"]). The tweets were crawled for over a time period of four months and around 10k tweets are collected related to the keywords. The Twitter API provides us with many attributes, but we have only stored the attributes relevant to us. The real-time streaming tweets are then

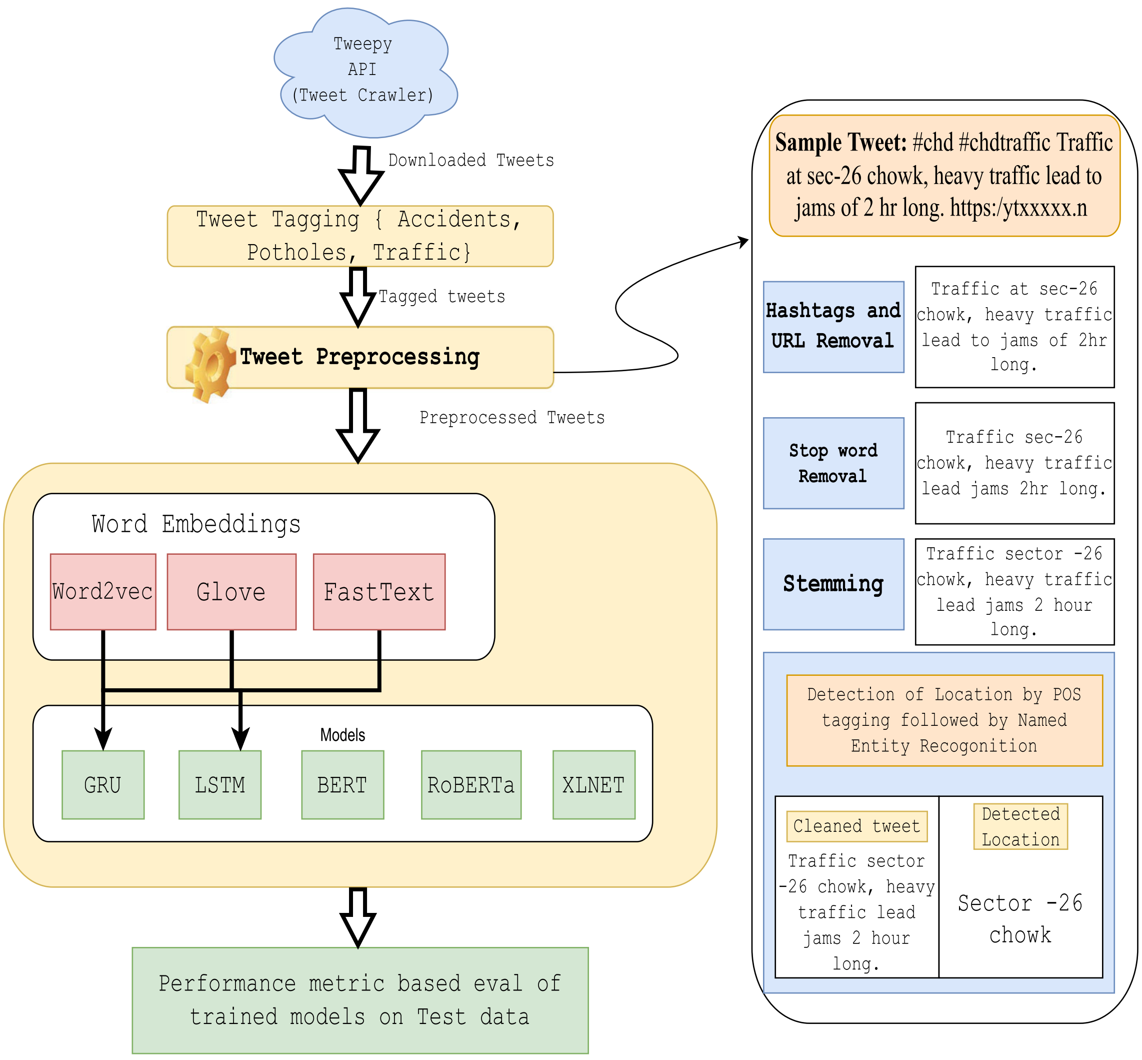

Tweepy API (Tweet Crawler)
Downloaded Tweets
Tweet Tagging { Accidents, Potholes, Traffic}
Tagged tweets
Tweet Preprocessing
Preprocessed Tweets
Word Embeddings
Word2vec
Glove
FastText
Models
GRU
LSTM
BERT
RoBERTa
XLNET
Performance metric based eval of trained models on Test data
Sample Tweet: #chd #chdtraffic Traffic at sec-26 chowk, heavy traffic lead to jams of 2 hr long. https:/ytxxxxx.n
Hashtags and URL Removal
Traffic at sec-26 chowk, heavy traffic lead to jams of 2hr long.
Stop word Removal
Traffic sec-26 chowk, heavy traffic lead jams 2hr long.
Stemming
Traffic sector -26 chowk, heavy traffic lead jams 2 hour long.
Detection of Location by POS tagging followed by Named Entity Recogonition
Cleaned tweet
Traffic sector -26 chowk, heavy traffic lead jams 2 hour long.
Detected Location
Sector -26 chowk

Fig. 1 Methodology of the traffic event detection pipeline

stored with the help of SQLAlchemy [36], (SQLAlchemy gives application developers the full power and is flexible integrated Python SQL toolkit with Object Relational Mapper). The information collected in the crawled tweets includes tweets geolocation, tweet text, tweet id, and search word as a Relational Database. The word cloud representation of the collected tweets is shown in Fig. 2.

3.2 Pre-processing of tweets and location extraction

It is an utmost as well as a challenging task to predict emotional variability due to the unusual, short text type (wassup, bruh, etc.), short length, and slang text for tweets. In many forums, the language used which describes the sentiments of users is less formal. Users create their names and spelling as punctuation marks, misspellings, pronouns, URLs, and terms and use abbreviations of different types.

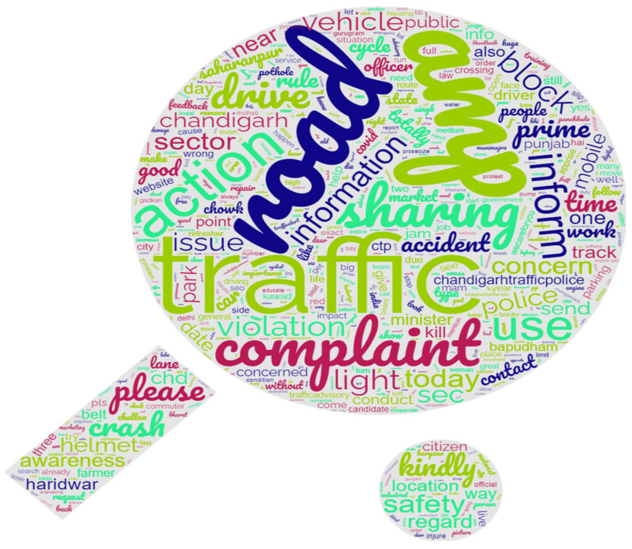


Fig. 2 Word cloud representation of frequently occurring words

Therefore, these kinds of text need to be corrected. Henceforth, for textual analysis, HTML characters, thumbnails, stop words, punctuation marks, slang words, URLs, etc., need to be removed [37, 38].

Before preprocessing the data for Sentimental Analysis, a model is trained for Named Entity Recognition (NER) which helps in extracting the location from the textual content. POS tagging is done on sample tweets to identify the locations, which are then trained as a NER model. All of this combined is a step-by-step process, explained as follows:

- **Hashtag elimination:** Hashtags "#" have been recognized by Twitter/ social media users to be used with a connected or relevant keyword/ word to specify their tweets and display them to a greater audience without problems in Twitter search. So, hashtags symbols have been eliminated as they carry zero relevance.
- **URL removal:** URL addresses are links to significant web resources. But in general, they don't have the relevant information. Therefore, the URL will be removed.
- **Handle removal:** "@" is used in Tweet by the user to tag or refer other Twitter users to follow the tweet, which is also irrelevant to our study. Hence, "@" symbols are eliminated from the tweets.
- **Remove stop words:** In any language, stop words are commonly used in every line. While their use in the language is substantial, they carry zero weightage to our study. Stop words commonly consist of adverbs, articles, conjunction, etc., Stop-word removal is an important step in text pre-processing and the mandatory stop-word list is used from the NLTK (Natural Language Toolkit) library.
- **Disposal of punctuation marks:** These symbols, which consist primarily of periods, commas, and parentheses, were standard punctuation marks used to separate sentences so that the elements could be understood quickly. However, these elements do not provide factual data on a regular basis. Therefore, these punctuation marks have been removed from the tweet.
- **Stemming:** Stemming is a process that is used to reduce each word to its base form. The aim is to group words with similar semantics to the standard format in order to train the model more efficiently.
- **Detection of location:** A considerable number of tweets have a location in the textual content. It has been observed that only a few tweets have geolocation tagged with them, but the count or percentage was not considerable. Considering location from the textual content is much more accurate as people mention it more often. In this context, Named Entity Recognition (NER) is deployed to spot the area. The NER identifies the organizations named in the text and classifies them by organization, place, and categories of people. However, the NER method fails because many localities and road names in India are named after some renowned person. For example, "Too much traffic near Elante mall in Ashok Nagar stuck around 2 hours in a jam". NER identifies "Ashok Nagar" as a person, not a location in this example.

3.3 Identifying the type of word-embeddings and deep learning models to be used

With the clean dataset obtained after pre-processing, it has been processed through different word embeddings and deep learning models. Convolutional Neural Networks (CNN) [25] and Recurrent Neural Networks (RNNs) [39] are two broadly used deep learning architectures with first-rate success within the textual content type problem. But CNN and RNN do not learn on text; they need numerical/ floating-point numbers, which computers understand. To convert these texts numerically,Word Embeddings are used.

3.3.1 Word embeddings

Word embeddings [16] are used to convert words and complete sentences into a numerical representation. As computers apprehend the language of zeros and ones, we strive to encode words with us to numbers so that the computer can learn them and process them. But studying and processing are not the most straightforward matters which are needed to be done by computers. Additionally, computer systems are needed to build a connection between each word in a sentence or document and other words in the same text.

Word embeddings are supposed to apprehend the textual context of the paragraph or preceding sentences alongside capturing the semantic and syntactic properties and similarities.

For example: "The Policeman was standing beside the road while the pedestrian couldn't cross the road due to traffic."

...then, we will take the two subjects (Policeman and Pedestrian) and switch them within the sentence making it:

"The Pedestrian was standing beside the road while the Policeman couldn't cross the road due to traffic."

The semantic or meaning-related relationship is preserved in both sentences, i.e., policeman and pedestrian are humans. And the sentence makes sense. In addition, the sentence also preserved syntactic relationships, i.e., rule-based relationships or grammar.

To find that semantic and syntactic relationship, The mapping of sentences to numbers is not the only thing needed. Thus, a better and more extensive illustration of these numbers representing semantic and syntactic properties is needed. Hence, word embeddings are used to generate these kinds of relationships. The different Word Embeddings used in this paper are explained as follows:

- **Word2vec model** [40]: It is a language model of the Neural Network with two-layer structures. It takes as input an extensive collection of words and then after the processing of those words, it produces a vector space of several hundred sizes. All the unique words are identified and all of them within the corpus are given a corresponding vector in space. Word vectors are inserted into the vector space so that the percentage of familiar words within the corpus is placed next to each other in the space. Word2vec uses CBOW or Skip-Gram to calculate weights. In CBOW, the words given in the context predict the target word. While in Skip-Gram, when looking at a target word, we do the opposite i.e., prediction of the context words. The use of parameters of various models and specific corpus sizes can significantly affect the quality of the word2vec model. **Parameters:** Accuracy may be advanced in some methods, along with better model architecture choices (CBOW or Skip-Gram), expanded training record sets, increased dimensionality of vectors in various ways, as well as using the algorithm with an expanded window size of words. The aforementioned upgrades come with the cost of extended computational complexity and consequently accelerated model generation time. Also, accuracy increases typically because the number of words used increases and the range of dimension increases. If the amount of training data is multiplied with a factor of n, it results in an equal increment in computational complexity as it does with multiplying the number of vector dimensions with the same factor n. In models using massive corpora and a high quantity of dimensions, the Skip-Gram model yields the best overall accuracy, continually produces the highest accuracy on semantic relationships and yields the best syntactic accuracy in maximum instances. However, CBOW costs less per computer and has approximately the same accuracy results.
- **GloVe** [41]: It is a method to capture global corpus statistics. It is an increment to the shallow window approach of the early 2000s in which the semantic meaning is analyzed on a small window around the word in a text sentence. The goal of this word Embedding is to reduce the dimensionality, as it is not possible to feed the network a binary ASCII or something one-hot encoded, based on comprehensive vocabulary because these soon blow up into unacceptably large dimensions that any work simply cannot handle. GloVE aims to improve all of these by capturing the context of the word in the embedding through explicitly capturing co-occurrence probabilities. In GloVE, we incorporate a notion of contextual distance, i.e., the word closer to the tagged word is weighed more than the words which are farther. To make a difference with Word2vec, Word2vec embedding is based on training a neural feed-forward network, while glove embedding is learned based on matrix-based techniques. The technique is based on matrix factorization techniques in the name of the matrix content. It begins to form a large matrix (of words x context) for information that happens together, that is, each ”word“ (lines), counting how often we see this word in another ”total“ (columns) in a large corpus. The number of ”contexts“ is quite large, as it is equal in size. Therefore, this matrix is combined to produce another matrix (as shown in Fig. 3) with low resolution, where each line has now a vector image of the corresponding word. Typically, this is done by minimizing ”reconstruction-loss“. These losses attempt to obtain low-level presentations that may explain many of the variations in high-dimensional data.
- **FastText** [42]: It has the same goal as Word2vec, i.e., to learn the representations of the words. But the key difference is fastText works at a more granular level with letters and grams. Where words are represented by the sum of the character n-gram vectors. For example, the word “accident” is the sum of the vectors of n-grams: “ac”, “acc”, “acci”, “accid”,“accide”,“acciden”,“accident”, “cc”,“cci”,“ccid”,“cciden”,“ccident”,“cid”, “cide”, “ciden”, “cident” , “ide”,“ident”. It is possible to generate better words embeddings for rare words using fastText because even if words are rare their character n-grams are still shared with other words. The main issue regarding FastText is that the memory requirement grows much more if the corpus size grows. So, fastText even being faster and better shall only be used if your data contains more rare words than the common ones.

In addition to these, other state-of-the-art natural language processing models like BERT and RoBERTa are also used for sentiment classification, which are discussed later in this paper.

3.3.2 Recurrent neural networks (RNNs)

Recurrent neural networks [43, 44] are a type of Deep Neural network mainly used for natural language processing tasks. To get an idea about it, a few applications of

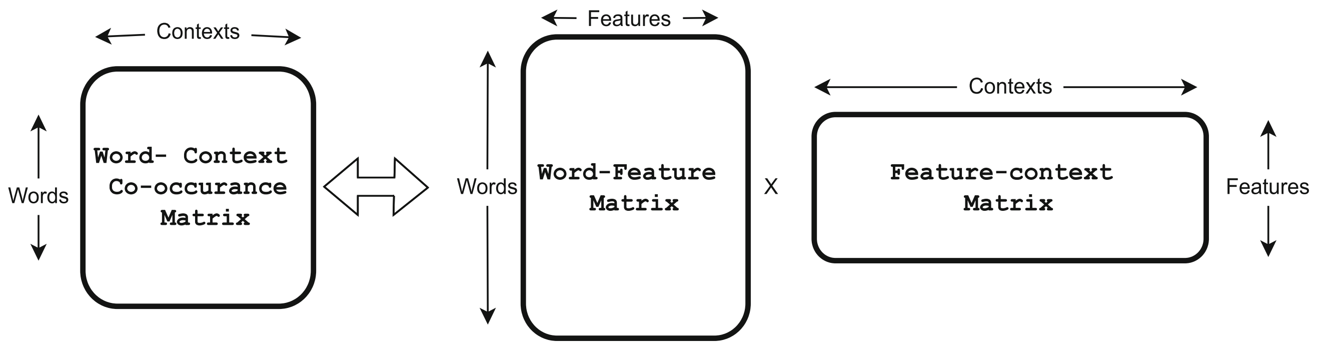


Fig. 3 Embedding matrix representation

RNN are (a) Sentence auto-completion used in Gmail services. (b) Google language translator (c) Sentiment Analysis. RNNs are used for these types of problems because these are sequence modeling problems, i.e., saving the output of a particular layer and feeding it back to the input in order to predict the output. In Fig. 4, $h(t-1)$ and $h(t)$ are hidden states, while $x(t)$ is the input word. $h(t-1)$ and $x(t)$ go through the weighted sum and then through the activation function, i.e., tanh in case of RNNs.

RNNs themselves face the vanishing gradient problem. To overcome this, there are particular types of RNNs like LSTM and GRU that address the issue of short-term memory.

3.3.3 LSTM

LSTM [44, 45] or long short-term memory is a special type of RNN that solves traditional RNN's short-term memory problem. In RNNs, there is the cell state h, i.e., the short-term memory, while in LSTM, we introduce a new state c, i.e., the long term memory as shown in Fig. 5. The state of the cell is like a transmission belt, which goes down straight across the chain, with just a little junction of the line. It makes it easy for information to flow freely. LSTM cell has majorly three types of gates, namely, forget gate, input gate, and output gate. Forget gate helps in discarding previous memory, which is no longer helpful. Input gate adds the memory of the new relevant word that appeared. Output gate does the weighted sum of the previous hidden state and the new long-term memory and feeds the new hidden state as the input for the next state. A typical working model of LSTM is shown in Fig. 5.

3.3.4 GRU

GRU or Gated Recurrent Units [44, 46] is a version of RNN, a newer version than LSTM, and was invented in 2014. While LSTM has a separate long-term and short-term memory, GRU is a modified or lightweight version that combines these long-term and short-term memory into its hidden state. GRU consists of two gates, namely, the update gate and the reset gate (as shown in Fig. 6). Update gate knows how much of the past memory is to be retained, and reset gate knows how much past memory to forget. Due to the simpler architecture of GRUs, they are faster to train.

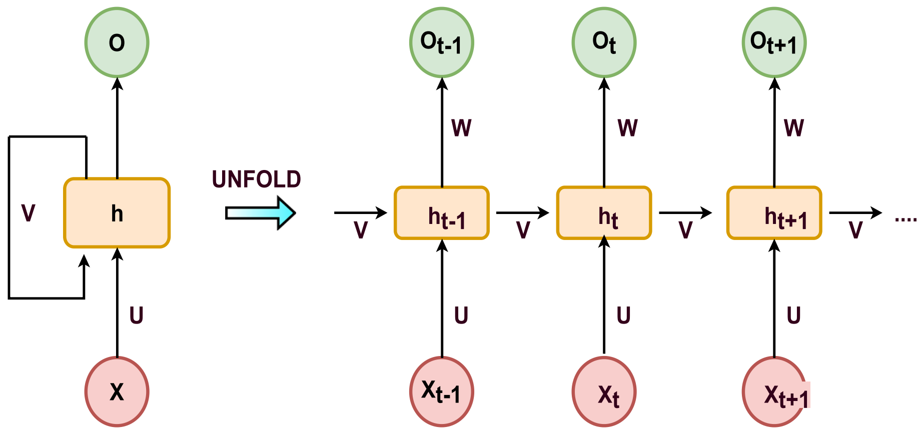


Fig. 4 Recurrent neural networks

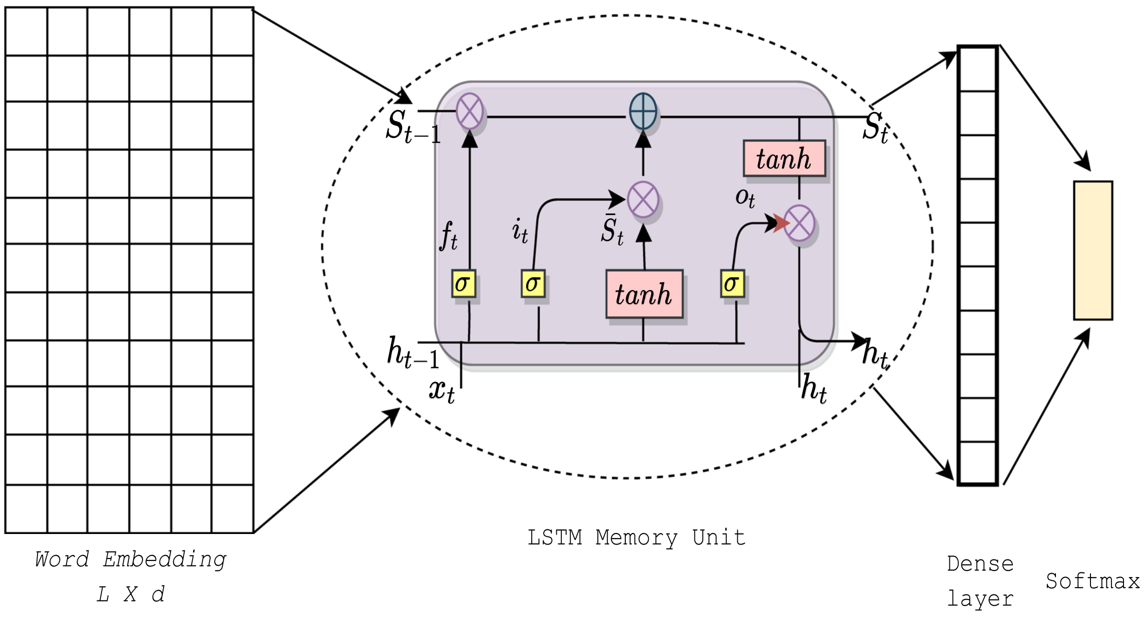
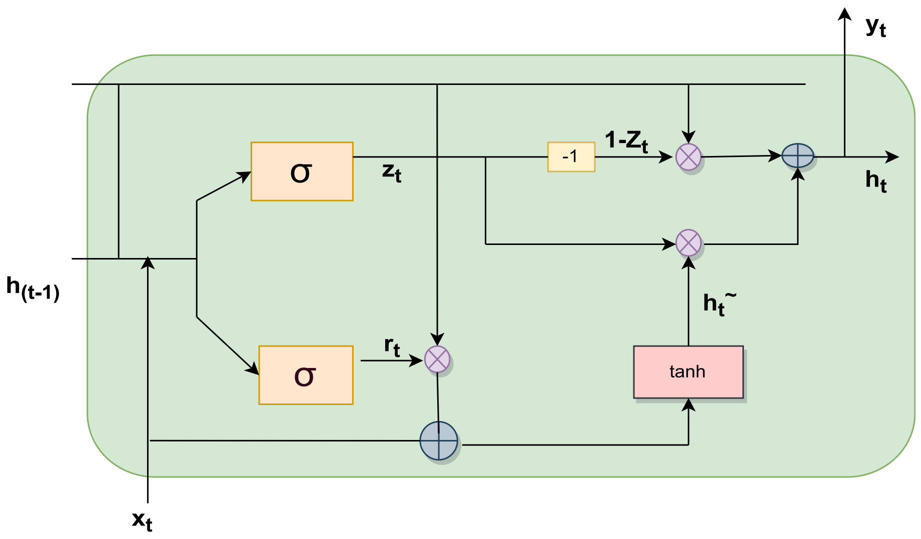


Fig. 5 LSTM network model

Fig. 6 GRU network model



4 Implementation details

The following models are tested against the dataset obtained after the pre-processing step:

1. GloVe with Bidirectional GRU
2. GloVe with Bidirectional LSTM
3. Fast text with Bidirectional LSTM
4. Word2Vec with Bidirectional LSTM
5. Bidirectional Encoder Representations from Transformers (BERT)
6. RoBERTa
7. XLNET

The detailed explanation of these models is given as follows:

4.1 GloVe with bidirectional GRU

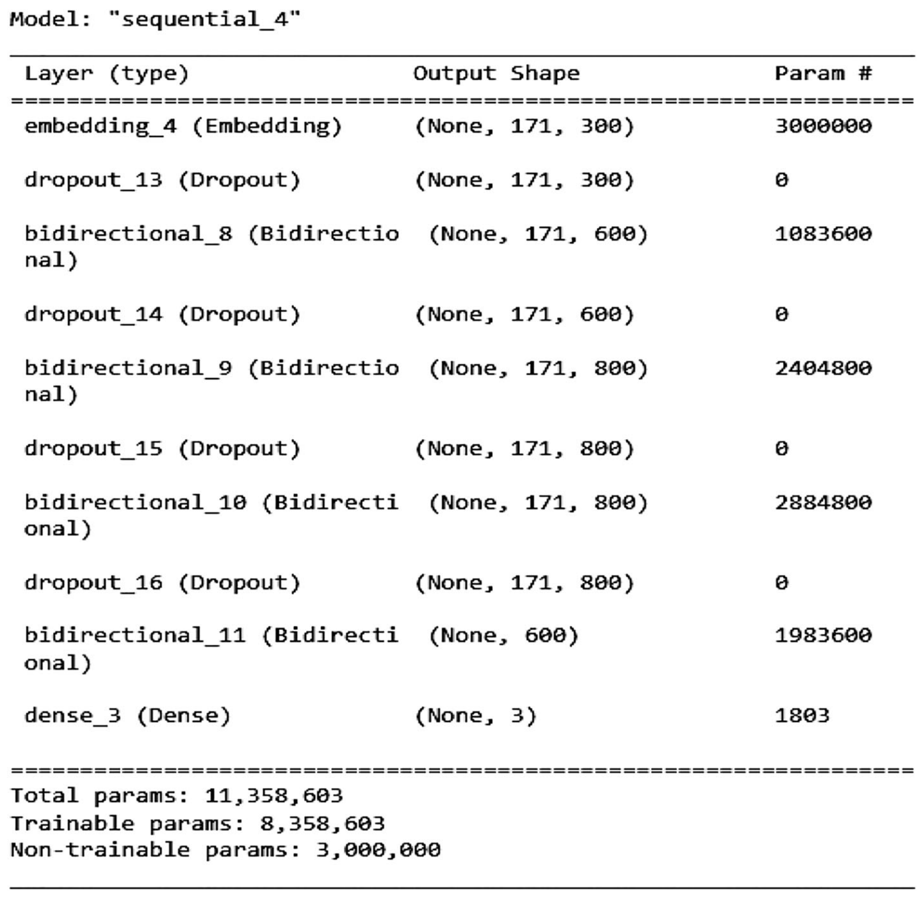
In most of the NLP tasks, texts are converted into integer values that are further fed into models. One way to convert text into a sequence of numbers is with the help of ‘Keras’. ‘Keras’ with its API provides ways to tokenize the texts. Tokenizer in Keras is used to find the frequency of every distinct word and then it arranges them according to it. Starting with 1, an integer value is assigned to every element on the stack. The index mapping dictionary can be accessed by using the ‘word_index’ function. The tokenizing is a two-step process, firstly we used 'fit_on_-texts' which creates word indices by fitting the tokenizer on our training data, followed by the 'text_to_sequences' function which uses the dictionary made in the previous step to transform both train and test data.

Now, 'num_words' attribute is set to 10000 words. 'num_words' is a parameter that defines the maximum number of words to keep, based on the occurrence/frequency at which the word appears. We can also leave the 'num_words' to be assigned as '*None*'; this would enable the tokenizer to select all the words in the vocabulary.

The next step is to pad the sequence, it is required that all the input sequences to the deep learning model need to be exactly the same length. To make this happen, a function has been used that pads the short sequences with zeros. It also crops a sequence if it is longer than a predefined parameter '*maxlen*'. In this model,'maxlen' is set to 171. Now on the tokenized text, we use pre-trained GloVe "*glove-wiki-gigaword-300*". The embedding matrix has a defined shape (vocabulary length $\times$ embedding dimension) which is initialized with zeros. If a word in the function 'word_index'is found to be missing in the embedding vectors from GloVe, the weight of that word remains as zero.

The generated model (Fig. 7) has the first layer as the Embedding layer. The embedding layer by 'Keras' is a tangible layer such that it can also be used without any pre-trained weights. In that case, the Embedding layer is initialized with random weights and will learn an embedding for all of the words in the training dataset. In this exercise, the weights of the Embedding layer are set to the embedding matrix from GloVe pre-trained vectors. Hence, the concept of transfer learning is implemented for achieving the better learning accuracy.

The embedding layer has a parameter called "trainable" that can be set to TRUE if you want to fine-tune the word embedding. If you don't want the embedding weights to be updated; as in this case, it is set to false. After the Embedding layer, multiple bidirectional GRU layers are added and then a dense layer with "*softmax*" activation to predict the Tweet as traffic, accident, or pothole-related. The optimizer used in compiling is "*adam*" with loss as "sparse_categorical_crossentropy". Further, hyperparameters in model fitting that are epochs, batch size, etc., are tuned by adapting the random search method, generating training and validation accuracy as shown in Fig. 2a.



```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_4 (Embedding)     (None, 171, 300)          3000000

 dropout_13 (Dropout)        (None, 171, 300)          0

 bidirectional_8 (Bidirectio  (None, 171, 600)         1083600
 nal)

 dropout_14 (Dropout)        (None, 171, 600)          0

 bidirectional_9 (Bidirectio  (None, 171, 800)         2404800
 nal)

 dropout_15 (Dropout)        (None, 171, 800)          0

 bidirectional_10 (Bidirecti  (None, 171, 800)         2884800
 onal)

 dropout_16 (Dropout)        (None, 171, 800)          0

 bidirectional_11 (Bidirecti  (None, 600)              1983600
 onal)

 dense_3 (Dense)             (None, 3)                 1803

=================================================================
Total params: 11,358,603
Trainable params: 8,358,603
Non-trainable params: 3,000,000
_________________________________________________________________
```

Fig. 7 GLoVE with GRU network model description

4.2 GloVe with bidirectional LSTM

All things like tokenizing and GloVe model download were kept the same as in the above model. The significant difference comes at the model building step, where LSTM layers are replaced with GRU layers. Now comes the part to define early stopping and model checkpoint function by 'Keras', in early stopping, the parameter mode can be assigned three options—min, max, and auto. In the min mode, the training will stop if the supervised number has ceased to decrease; in the "*max*" mode, it will stop when the rented value has stopped increasing; in the "*automatic*" mode, the direction is automatically directed from the name of the supervised quantity. In our case, the quantity to monitor is validation loss. Furthermore, hyper-parameters are again set with the random search method and accuracy graphs are generated, shown in Fig. 2b.

4.3 Fast text with bidirectional LSTM

After tokenizing and padding the sequences as performed in the previous models, the fastText English word vectors are added from "*cc.en.300.bin*" which have been trained on many different and vast datasets.

We analyzed the tweets on the fitted model and generated the classification report. On further analysis, It was found that the traffic category had an F1-score of 0.99. Fast-text being a better performer on rare words showed positive results, with accuracy's shown in Fig. 2c.

4.4 Word2vec with bidirectional LSTM and GRU

Word vectors for word2vec are used from "*Word2vec-google-news-300*" which is about 1.6GB in size. These word vectors are then trained on both LSTM and GRU. It is observed that GRU did better on validation data than LSTM keeping the same hyper-parameters. The Precision, Recall, and f1-score of all three categories were better than word2vec + LSTM as compared to word2vec + GRU-based architecture. Train and validation accuracy of both the models (Word2vec with LSTM and Word2vec with GRU) are measured on the number of epochs as shown in Fig. 2d, e, respectively.

4.5 Bidirectional encoder representations from transformers (BERT)

BERT [17] as the name stands uses transformers that again learns the contextual relationship between the words. It was introduced in 2018 by Google AI and is a revolutionary model for natural language processing. In vanilla form, a transformer has two different components, an encoder and a decoder. But in this task, we only need encoders to encode our text into numeric data. While providing an input to a BERT model, a single sentence is converted by adding three embeddings—Token embeddings, sentence embeddings, and transformer positional embeddings. In token embeddings '[CLS]' token is added at the beginning, and the '[SEP]' token is added at the end of each sentence. 'CLS' stands for classification and 'SEP' means separator. In sentence Embedding, the sentences are marked as Sentence A or Sentence B, while in transformer positional embeddings, it represents the position of each token in the sequence.

We used a BERT base which is a stack of 12 encoders (BERT large has 24 layers). These layers consist of a large feed-forward neural network, in which each layer applies self-attention and then passes it to the next layer and finally to the next encoder. The final encoder outputs three 3-dimensional arrays (tokens- input_words_ids, masks-input_masks, and segments- segment_ids). We build a BERT model with a max length of 171. The overview of the BERT model is shown in Fig. 8, and the corresponding layered description is depicted in Fig. 9. The evaluated train and validation accuracy of the BERT model are demonstrated in Fig. 2f.

4.6 RoBERTa

RoBERTa [19] or Robustly Optimized BERT Pre-training Approach is an optimized approach of BERT. BERT randomly masks and predicts tokens and this happens a single time at the time of pre-processing. We can increase the number of times the masking occurs but it still remains static. RoBERTa uses a different approach in which masking is done during training, which is also referred to as dynamic masking. i.e., each time a sentence is merged into a mini batch; it is masked, so the number of possible masked versions of each sentence is not tied as in BERT. RoBERTa also removed the next sentence prediction objective in BERT, which is nothing but the ability of the model to predict whether the segments of the target document are from the same or different documents with the loss of Next Sentence Prediction (NSP). Researchers have previously found that removal of NSP loss either improves performance or matches the performance with NSP. Hence, all these optimizations led to the creation of RoBERTa from BERT (Fig. 10).

4.7 XLNET

XLNET [18] is a permutational approach to solve the prediction of word tasks in contrast to BERT which uses a masking approach for NLP. The major difference between BERT and XLNET is BERT uses Auto-encoding, while

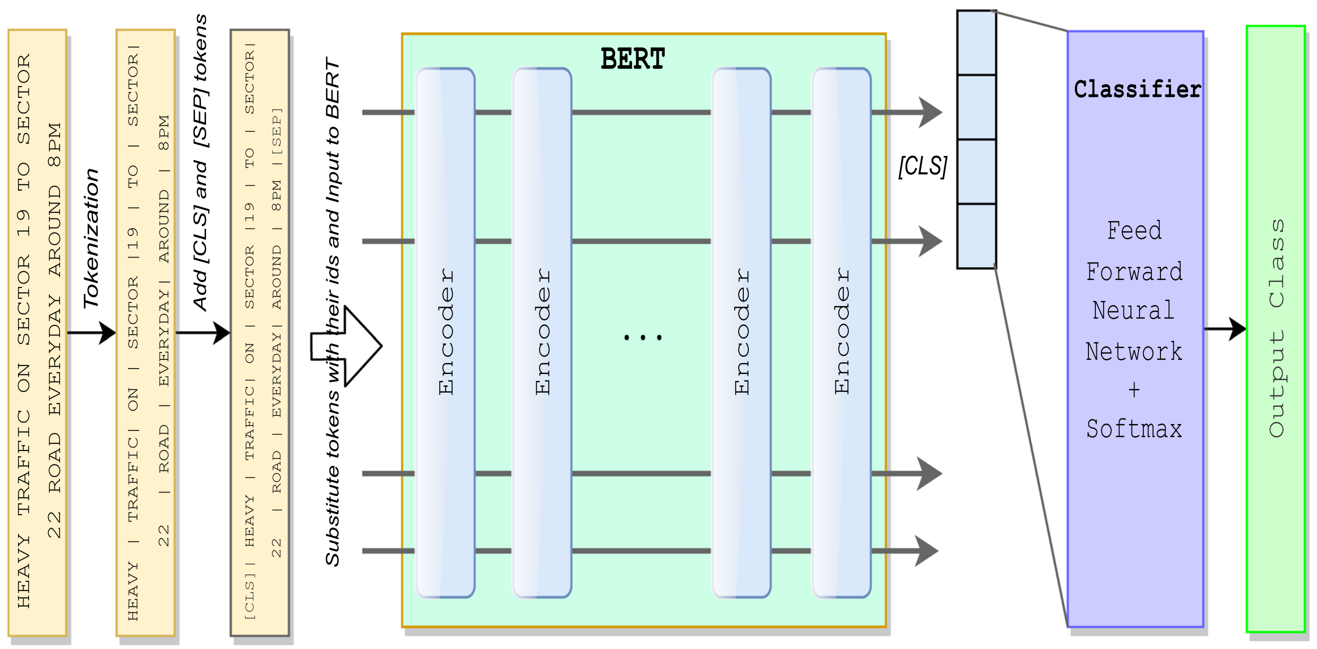
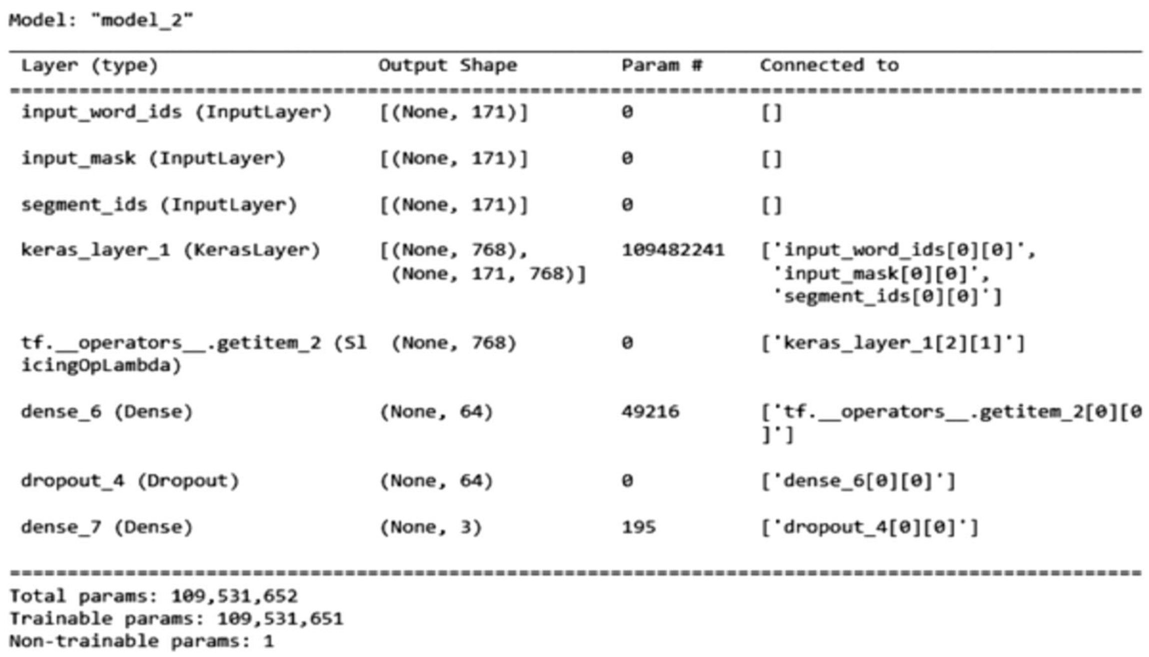

HEAVY TRAFFIC ON SECTOR 19 TO SECTOR 22 ROAD EVERYDAY AROUND 8PM
Tokenization
HEAVY | TRAFFIC| ON | SECTOR |19 | TO | SECTOR| 22 | ROAD | EVERYDAY| AROUND | 8PM
Add [CLS] and [SEP] tokens
[CLS]| HEAVY | TRAFFIC| ON | SECTOR |19 | TO | SECTOR| 22 | ROAD | EVERYDAY| AROUND | 8PM |[SEP]
Substitute tokens with their ids and Input to BERT
BERT
Encoder
Encoder
...
Encoder
Encoder
[CLS]
Classifier
Feed Forward Neural Network + Softmax
Output Class

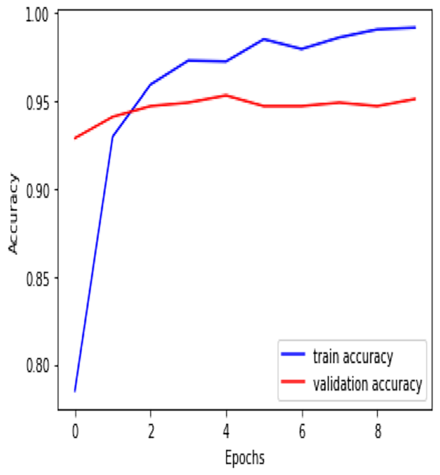
Fig. 8 BERT model



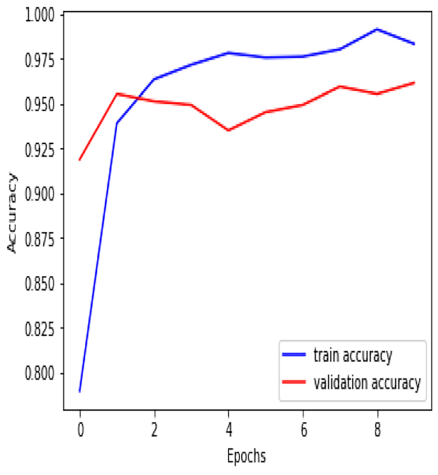
```
Model: "model_2"
____________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
====================================================================================================
 input_word_ids (InputLayer)    [(None, 171)]        0           []

 input_mask (InputLayer)        [(None, 171)]        0           []

 segment_ids (InputLayer)       [(None, 171)]        0           []

 keras_layer_1 (KerasLayer)     [(None, 768),        109482241   ['input_word_ids[0][0]',
                                 (None, 171, 768)]                'input_mask[0][0]',
                                                                  'segment_ids[0][0]']

 tf.__operators__.getitem_2 (Sl  (None, 768)         0           ['keras_layer_1[2][1]']
 icingOpLambda)

 dense_6 (Dense)                (None, 64)           49216       ['tf.__operators__.getitem_2[0][0
                                                                 ]']

 dropout_4 (Dropout)            (None, 64)           0           ['dense_6[0][0]']

 dense_7 (Dense)                (None, 3)            195         ['dropout_4[0][0]']

====================================================================================================
Total params: 109,531,652
Trainable params: 109,531,651
Non-trainable params: 1
```

Fig. 9 Layered description of the BERT model

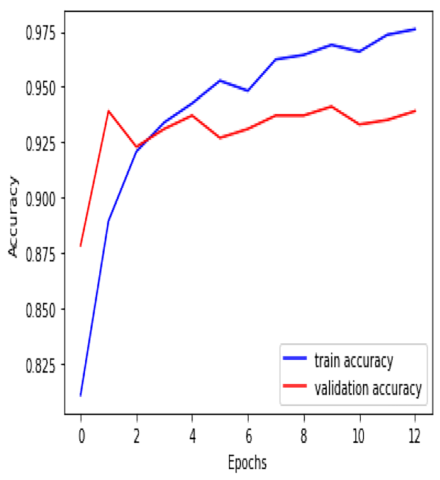
XLNET is an Auto-Regressive Pretraining method. It uses the concept of joint probability in which it calculates the probability of the word token on the basis of all the permutations of word tokens in a sentence. BERT has a limitation in that it cannot justify the interdependence of two masked tokens. XLNET does not use LSTM and only uses self-attention. It has also integrated methods from TransformerXL which can catch dependency from far away. Also, from the available models "Xlnet-large-cased" model is used to generate word embeddings, and predictions are made on the pre-processed dataset. The accuracy of the model is evaluated and is shown graphically in Fig. 2g.



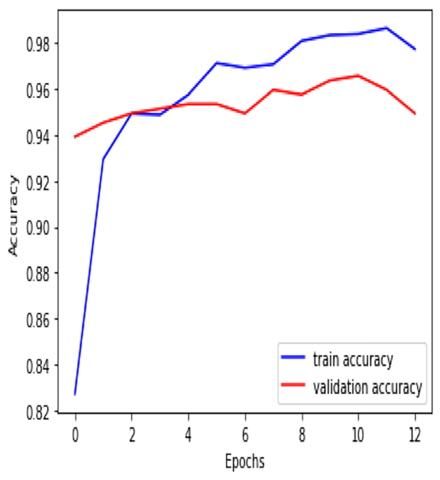
(a) GloVE - GRU



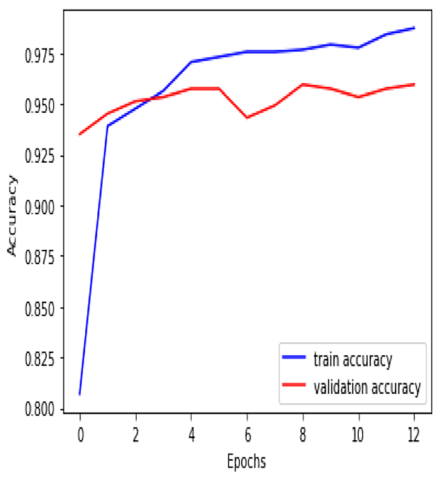
(b) GLoVE -LSTM



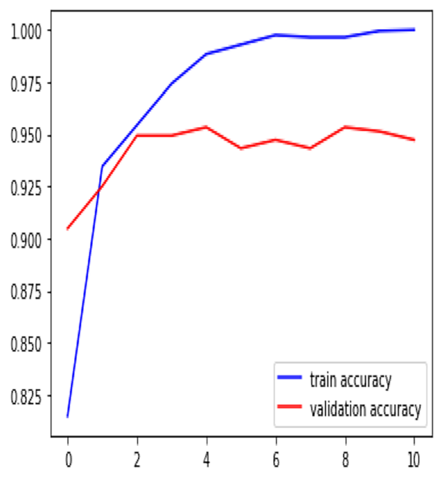
(c) FastText -LSTM



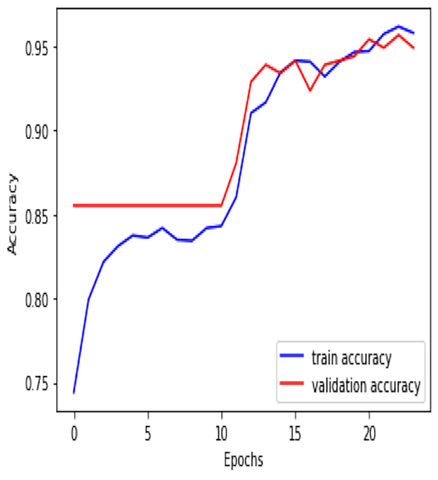
(d) Word2vec - LSTM



(e) Word2Vec - LSTM



(f) BERT



(g) XLNET

Fig. 10 Models' performances (RMSE) comparison

5 Results and discussion

In this section, to begin with, we have assessed the models or architecture of all the models mentioned above with the means of graphs (Fig. 2a–g). We have discussed the various evaluation parameters used to evaluate the models. To conclude, the evaluation of models is conducted on real-world problems of traffic, accidents, and potholes. The evaluation metrics that are used for classification problems are discussed as follows:

- **Accuracy** [49]; It is defined as the ratio of the number of correctly classified cases to the total number of cases predicted. Accuracy is not a reliable measure of the model if the dataset is imbalanced. It is given as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

- **Precision** [49]: It is calculated as the ratio of true positive to the total of true positive and false positive (i.e., the number of predicted positives).

$$\text{Accuracy} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

- **Recall** [49]: It is calculated as the number of true positives divided by the total number of actual positives. It is good to have a higher recall in cases like medical diagnosis.

$$\text{Accuracy} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

- **F1-Score** [49]: The F1-score is the harmonic mean of precision and recall. It is highest when precision is equal to recall.

$$\text{Accuracy} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

The above listed performance measures have been evaluated on the collected dataset. The categories are labeled as: **0**-*ACCIDENTS*, **1**- *TRAFFIC*, **2**- *POTHOLES*. The evaluation results are listed in Tables. Measuring the accuracy (Table 1) of different models evaluated in this data, it has been observed that the accuracy of these models does not vary that much. This is due to the imbalanced distribution of data among the three categories. Anyhow, GloVe, word2vec with BiLSTM and RoBERTa has shown the same test accuracy of 96% which doesn't differ much from the rest of the models. So, we shifted to other evaluation metrics.

Table 1 Accuracy of several models implemented in the current research study

Word embedding/ Model	Accuracy
GloVe + BiGRU	0.95
GloVe + BiLSTM	0.96
FastText + BiLSTM [31]	0.95
Word2vec +BiLSTM [47]	0.96
BERT [13] [48]	0.95
XLNET	0.95
Proposed approach (RoBERTa)	0.97

This paper has three use cases, i.e., Traffic, accidents, and potholes. Depending on the use case, we studied which evaluation metric is a better measure to evaluate the model. Considering accidents, it is about saving lives and is an issue of high importance. Therefore, Recall (Table 2) is a good measure as it is higher when false negatives are less. Furthermore, the model of category 0-Accidents is good to be measured by the recall, and it is found that word2vec gives the highest recall among the eight models evaluated.

The next two categories are Traffic and potholes, Traffic is a problem that has to be precisely measured to reduce the unnecessary manpower and development costs. Again, we found that word2vec slightly outperforms the other models with a precision of 0.98. In the end, potholes are somewhat a problem that can be addressed instantly or can have a delayed solution. We can choose F1-score to compare the models.On analysis, RoBERTa gave the highest F1 score (Table 3) for category 2-Potholes.

6 Conclusion

The proposed research has suggested the use of the data fetched from Twitter to formulate highly accurate deep learning models. The approach targets areas affected by various traffic-related issues in a four-stage architecture format to treat the said issues with an increased level of efficiency. The first problem that this study encountered was the abundance of meta-information that resided within the data fetched from Twitter. The data while being fetched from the Twitter API was simultaneously labeled which further helped the program to categorize the information among three broad categories (accidents, potholes, traffic). Secondly, the data were used to construct distributed vector representations using different word embeddings (Word2-vec, Glove, FastText). These embeddings were then combined with different deep learning models in many possible permutations. Furthermore, many different approaches like- the self-attention mechanism of the transformers (BERT and RoBERTa) and per-mutational approach (as of XLNET) are also applied for the sentiment classification task. Finally, all these techniques are compared and evaluated with different evaluation parameters (Accuracy, Precision, Recall, and F1-measure) to conclude the best

Table 2 Recall

Word embedding/model	RECALL				
	0-Accident	1-Traffic	2-Potholes	Macro avg	Weighted avg
GloVe + BiGRU	0.79	0.99	0.44	0.74	0.95
GloVe + BiLSTM	0.82	0.99	0.56	0.79	0.96
FastText + BiLSTM	0.80	0.98	0.44	0.74	0.95
Word2vec +BiLSTM	0.82	0.99	0.56	0.79	0.96
BERT	0.79	0.98	0.44	0.74	0.95
XLNET	0.76	0.99	0.44	0.73	0.95
Proposed approach (RoBERTa)	0.79	0.99	0.6	0.79	0.96

Table 3 F1-score

Word embedding/model	F1-score				
	0- Accident	1-Traffic	2-Potholes	Macro avg	Weighted avg
GloVe + BiGRU	0.83	0.97	0.62	0.8	0.95
GloVe + BiLSTM	0.86	0.97	0.67	0.83	0.95
FastText + BiLSTM	0.83	0.97	0.57	0.79	0.94
Word2vec +BiLSTM	0.88	0.98	0.67	0.84	0.96
BERT	0.83	0.97	0.62	0.8	0.94
XLNET	0.83	0.97	0.62	0.8	0.95
Proposed approach (RoBERTa)	0.85	0.98	0.75	0.86	0.96

technique for the given task of real-time traffic detection. From the experimental evaluation, it has been observed that the RoBERTa language model-based pipeline outperforms all exisiting techniques by achieving 97% accuracy, 96% Recall and 95% F1-Score. Furthermore, the evaluation for our study can be concluded with the argument that tweets having fixed-length limits the user to generate lengthily context. Hence, many of the models which detect the syntactic relationships at a greater level are of the same use as traditional models. Thus, word2vec gives better results in two out of three cases.

Data availability statement The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The author(s) declare(s) that there is no conflict of interest.

References

1. Sarrab M, Pulparambil S, Awadalla M (2020) Development of an iot based real-time traffic monitoring system for city governance. Glob Transit 2:230–245
2. Zhu F, Lv Y, Chen Y, Wang X, Xiong G, Wang F-Y (2019) Parallel transportation systems: toward iot-enabled smart urban traffic control and management. IEEE Trans Intell Transp Syst 21(10):4063–4071
3. Singh R, Sharma R, Akram SV, Gehlot A, Buddhi D, Malik PK, Arya R (2021) Highway 4.0: Digitalization of highways for vulnerable road safety development with intelligent iot sensors and machine learning. Safety Sci 143:105407
4. Silva BN, Khan M, Han K (2018) Towards sustainable smart cities: a review of trends, architectures, components, and open challenges in smart cities. Sustain Cities Soc 38:697–713
5. Afrin T, Yodo N (2020) A survey of road traffic congestion measures towards a sustainable and resilient transportation system. Sustainability 12(11):4660
6. D'Andrea E, Ducange P, Lazzerini B, Marcelloni F (2015) Real-time detection of traffic from twitter stream analysis. IEEE Trans Intell Transp Syst 16(4):2269–2283
7. Ilieva RT, McPhearson T (2018) Social-media data for urban sustainability. Nat Sustain 1(10):553–565
8. Hysa B, Zdonek I, Karasek A (2022) Social media in sustainable tourism recovery. Sustainability 14(2):760
9. Dixon S (2022) Countries with most twitter users 2022. https://www.statista.com/statistics/242606/number-of-active-twitter-users-in-selected-countries. Accessed: (April 3, 2023)
10. Duan HK, Vasarhelyi MA, Codesso M, Alzamil Z (2023) Enhancing the government accounting information systems using social media information: An application of text mining and machine learning. Int J Account Inf Syst 48:100600
11. Savastano M, Suciu M-C, Gorelova I, Stativă G-A (2023) How smart is mobility in smart cities? an analysis of citizens' value perceptions through ict applications. Cities 132:104071
12. Agarwal A, Toshniwal D, Bedi J, (2019) Can twitter help to predict outcome of, Indian general election: A deep learning based study. Joint Eur Conf Mach Learn Knowl Discov Databases. Springer 2019:38–53
13. Bedi J, Toshniwal D (2022) Citenergy: a bert based model to analyse citizens' energy-tweets. Sustain Cities Soc 80:103706

14. Mohit B (2014) Named entity recognition. In: Natural language processing of semitic languages. Springer, pp 221–245
15. Voutilainen A (2003) Part-of-speech tagging. In: The Oxford handbook of computational linguistics, pp 219–232
16. Torregrossa F, Allesiardo R, Claveau V, Kooli N, Gravier G (2021) A survey on training and evaluation of word embeddings. Int J Data Sci Anal 11:85–103
17. Devlin J, Chang M-W, Lee K, Toutanova K (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint: arXiv:1810.04805
18. Yang Z, Dai Z, Yang Y, Carbonell J, Salakhutdinov RR (2019) Le QV, Xlnet: generalized autoregressive pretraining for language understanding. Adv Neural Inf Process Syst, 32
19. Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L, Stoyanov V (2019) Roberta: a robustly optimized bert pretraining approach. arXiv preprint: arXiv:1907.11692
20. Hu D, Wu J, Tian K, Liao L, Xu M, Du Y (2017) Urban air quality, meteorology and traffic linkages: evidence from a sixteen-day particulate matter pollution event in december 2015, Beijing. J Environ Sci 59:30–38
21. Giuliano G, Lu Y (2021) Analyzing traffic impacts of planned major events. Transp Res Record 2675(8):432–442
22. da Silva Barboza F, Stumpf L, Pauletto EA, de Lima CLR, Pinto LFS, Jardim TM, Pimentel JP, Albert RP, Vivan GA (2021) Impact of machine traffic events on the physical quality of a minesoil after topographic reconstruction. Soil Tillage Res 210:104981
23. Ribeiro Jr. SS, Davis Jr. CA, Oliveira DRR, Meira Jr. W, Gonçalves TS, Pappa GL (2012) Traffic observatory: a system to detect and locate traffic events and conditions using twitter. In: Proceedings of the 5th ACM SIGSPATIAL international workshop on location-based social networks, pp 5–11
24. Gu Y, Qian ZS, Chen F (2016) From twitter to detector: real-time traffic incident detection using social media data. Transp Res Part C Emerg Tech 67:321–342
25. Dabiri S, Heaslip K (2019) Developing a twitter-based traffic event detection model using deep learning architectures. Exp Syst Appl 118:425–439
26. Albuquerque FC, Casanova MA, Lopes H, Redlich LR, de Macedo JAF, Lemos M, de Carvalho MTM, Renso C (2016) A methodology for traffic-related twitter messages interpretation. Comput Ind 78:57–69
27. Essien A, Petrounias I, Sampaio P, Sampaio S (2021) A deep-learning model for urban traffic flow prediction with traffic events mined from twitter. World Wide Web 24(4):1345–1368
28. Alomari E, Katib I, Mehmood R (2020) Iktishaf: a big data road-traffic event detection tool using twitter and spark machine learning. Mobile Netw Appl, pp 1–16
29. Agarwal S, Mittal N, Sureka A (2018) Potholes and bad road conditions: mining twitter to extract information on killer roads. In: Proceedings of the ACM India joint international conference on data science and management of data, pp 67–77
30. Chaturvedi N, Toshniwal D, Parida M (2019) Twitter to transport: geo-spatial sentiment analysis of traffic tweets to discover people‘s feelings for urban transportation issues. J Eastern Asia Soc Transp Stud 13:210–220
31. Suat-Rojas N, Gutierrez-Osorio C, Pedraza C (2022) Extraction and analysis of social networks data to detect traffic accidents. Information 13(1):26
32. Yao W, Qian S (2021) From twitter to traffic predictor: next-day morning traffic prediction using social media data. Transp Res Part C Emerg Technol 124:102938
33. Milusheva S, Marty R, Bedoya G, Williams S, Resor E, Legovini A (2021) Applying machine learning and geolocation techniques to social media data (twitter) to develop a resource for urban planning. PloS One 16(2):e0244317
34. Azhar A, Rubab S, Khan MM, Bangash YA, Alshehri MD, Illahi F, Bashir AK (2022) Detection and prediction of traffic accidents using deep learning techniques. Cluster Comput, pp 1–17
35. Deb S, Chanda AK (2022) Comparative analysis of contextual and context-free embeddings in disaster prediction from twitter data. Mach Learn Appl, p 100253
36. Sqlalchemy MB (2012) In: Brown A, Wilson G (eds) The architecture of open source applications volume II: structure, scale, and a few more fearless hacks. aosabook.org. http://aosabook.org/en/sqlalchemy.html
37. Singh T, Kumari M (2016) Role of text pre-processing in twitter sentiment analysis. Proc Comput Sci 89:549–554
38. Symeonidis S, Effrosynidis D, Arampatzis A (2018) A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis. Exp Syst Appl 110:298–310
39. Tarwani KM, Edem S (2017) Survey on recurrent neural network in natural language processing. Int J Eng Trends Technol 48:301–304
40. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv preprint: arXiv:1301.3781
41. Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1532–1543
42. Bojanowski P, Grave E, Joulin A, Mikolov T (2017) Enriching word vectors with subword information. Trans Assoc Comput Linguist 5:135–146
43. Aggarwal CC et al (2018) Neural networks and deep learning. Springer 10:978–3
44. Bengio Y, Goodfellow I, Courville A (2017) Deep learning, vol 1. MIT press Cambridge, MA, USA
45. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
46. Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint: arXiv:1412.3555
47. Xu M, Zhang X, Guo L (2019) Jointly detecting and extracting social events from twitter using gated bilstm-crf. IEEE Access 7:148462–148471
48. Prasad R, Udeme AU, Misra S, Bisallah H (2023) Identification and classification of transportation disaster tweets using improved bidirectional encoder representations from transformers. Int J Inf Manage Data Insights 3(1):100154
49. Tan P-N, Steinbach M, Kumar V (2016) Introduction to data mining. Pearson Education India