5thICITB

Detecting Traffic Event from Social Media Texts with Concatenated Word Embeddings

Zeynep OZER[1], Ilyas OZER[2*]

[1]Department of Management Information Systems, Bandirma Onyedi Eylul University, Balikesir, Turkey
[2]Department of Computer Engineering, Bandirma Onyedi Eylul University, Balikesir, Turkey

ABSTRACT
In recent years, efforts to detect traffic events from social media platforms have accelerated due to their extensive coverage and low costs. In the studies conducted to date, tweets have been converted into numerical vectors using the bag-of-words representations. However, bag-of-words do not take into account the order of words and have several problems, such as sparsity. Also, last studies have used supervised deep learning architectures and generic word embeddings, which obtained from sources like Wikipedia. Word embeddings obtained by using this type more formal spelling corpora is successful in representing the general meanings of words, while there are limitations in terms of both coping with noise in user-generated texts and representing domain-specific meanings of words. In this study, to overcome these problems, a domain-specific word embedding created for the traffic area consisting of approximately 1.5 M tweets and its concatenated with generic word embedding. Besides, two datasets were created, which are composed of 2 and 8 classes. Then, the concatenated word embedding tested on these datasets using a convolutional neural network (CNN) and long short-term memory (LSTM) architectures. Experimental results show that the proposed approach on the generated dataset provides a significant improvement over state-of-the-art methods.

Keywords: Traffic event detection, Domain-specific word embedding, Twitter, Deep learning

1. INTRODUCTION

Traffic congestions create serious problems that lead to loss of time and money both for drivers and traffic managers [1][2]. Traffic congestions can categorize into two main groups: recurring and non-recurring congestions [2]. The general source of recurring congestion based on several factors such as the choice of route, people's daily use habits and it repeats day by day. On the other hand, the main cause of non-recurring traffic congestion is due to special reasons such as traffic accidents, bad weather conditions, vehicle failures, road construction and maintenance also

* Corresponding author.
E-mail address: iozer@bandirma.edu.tr (I. Ozer).

these special causes make up almost half of the total traffic congestion [2] [3]. Therefore, timely, accurate and cost-effective detection of these special events is a crucial issue for traffic management. An effective solution to this issue would be very beneficial for reducing traffic congestions, especially in metropolises. Besides, it allows choosing alternative routes for drivers, finally to reduce the losses due to the congestions.

For decades, people have worked on systems to determine the type and location of traffic events in real-time [2]. Commonly used methods for detecting traffic events in developed systems based on measuring and evaluating parameters such as traffic density, flow and speed, with using physical sensors such as imaging sensors, acoustic sensors, magnetic sensors and passive infrared sensors. Measurements usually made by placing these sensors at a fixed point. The data obtained from scattered sensors in real-time evaluated in a broad temporal and spatial spectrum through data mining applications. These systems usually work successfully on the highway and main arteries but have poor performance in detecting events in local arteries [2]. Besides, used systems for traffic event detection are generally required high investment and maintenance costs. Also, the limited coverage of these systems can lead to delays in event detection [4]. In particular, the causes of congestions in the local arteries can be varied, such as faulty parking, pedestrian traffic, and municipal work. In this case, considering the current coverage of the sensors, it makes it challenging to detect traffic events.

The main reason for the low coverage of these physical sensors is that they are placed only rarely on the main arteries because of the high installation, maintenance, repair and operating costs. The cost of installation and maintenance of an inductive loop detector placed at an intersection is the US $ 9 500 to $ 16 700 per year [5][6]. Therefore, methods based on physical sensors may not always be the appropriate solution due to factors such as high-cost disadvantages and limited spatial coverage. However, in recent years a large amount of data produced by online communities can provide an alternative solution or that supports existing physical sensors, for detecting and evaluating traffic events with wide spatial coverage and low costs [7].

The enormous increase in the use of smartphones and mobile devices in recent years has become a promising alternative approach to collecting information about traffic events [8]. Twitter is one of the platforms that often share information with smartphones and mobile devices. Twitter allows users to obtain public tweets free of charge via REST API and Streaming API. Using the REST API application, messages that have been shared in the last seven days can be retrieved via keywords, while streaming API allows being obtained the shared tweets in real-time.

A series of studies based on text mining and machine learning algorithms have been conducted to detect traffic-related events from unstructured tweet texts. Already, text mining and machine learning have to deal with many challenging tasks. One of these challenges is the representation of variable length tweet text with fixed-length vectors. Bag-of-words is one of the most commonly used methods for vector representation of tweet texts in detecting traffic-related events. In this approach, each of the words in the documents treated as a unique property, and each of these unique properties can represent as binary or a function of the number of repetitions in the corresponding document or tweet (e.g., term frequency-inverse document frequency TF-

IDF). In the bag-of-words method, firstly a list is created using each unique word in the input data. The total number of elements in this list determines the size of the vector representation. The bag-of-words model is an unordered document representation approach of words so that in the vector representation obtained by this approach, no information stored about the orders of words found in the sentence. The important thing is whether the words found in the document or how many times seen. Accordingly, bag-of-words can be used together with different N-gram models to use partial position information of words. In both cases, the semantic and syntactic relationships largely lost due to the incomplete use of the temporal order of words in the tweet. Besides, each tweet consists of a much smaller number of words compared to the entire corpus. As a result, a rather large size sparse matrices obtained, which consist of many zeros.

On the other hand, in recent studies in the traffic event detection field, to overcome these problems approaches based on word embeddings and supervised deep learning architectures have used for tweet modelling. Word embedding is vital for the representation of tweet texts with low dimensional spaces, besides for representation in the semantic relationship between the words that generate the tweet. In this direction, general-purpose generic word embeddings commonly used in traffic studies [9][10]. Generic word embeddings are very successful in representing the general meaning of words. In this case, it may be particularly advantageous for the representation of non-traffic-related tweets. However, words can have different meanings, specifically for the domain in which they used, so generic word embeddings have significant limitations in representing domain-specific meanings [11]. Also, social media messages (SMMs) contain many noisy texts such as abbreviations, diacritic character problems, and the absence of vowels. In this case, it would be quite challenging to represent such noisy texts with generic word embeddings, especially from sources with more formal writing, such as Wikipedia data.

On the other hand, normalization can be performed to reduce noise in SMMs. However, it has numerous disadvantages, as the used language in social media platforms is constantly changing over time, as well the normalization process in morphologically rich and agglutinative languages is a quite challenging task [12]. Besides, if the number of classes increases depending on the desire to be detected in special cases related to traffic events such as accidents, road works and weather conditions, the domain-specific word embeddings may provide stronger word representations.

In this study, an approach proposed that uses together generic word embedding and domain-specific word embedding, which obtained by using unlabeled data related to the traffic domain, to detect traffic events from Twitter messages. That way, it is aimed to provide a more powerful representation of noisy texts and specific meanings of the words for the traffic domain, while ensuring that their generic meanings are not lost especially for non-traffic-related tweets. The proposed approach tested on two datasets obtained from Turkish Twitter messages. The first dataset divides to tweets into 2 different classes, as the traffic-related and non-traffic-related. On the other hand, other datasets consist of 8 different classes for detecting special cases related to traffic events such as accidents, weather conditions, roadworks and external events. Also, FastText method was used to obtain word embeddings, CNN and LSTM models used in the

classification process. In the tests performed on both datasets, and the model, which concatenated generic and domain-specific word embeddings performed better than the models using these word embeddings separately.

The main contributions of this paper are as follows:

- Domain-specific word embedding created for the traffic area, which consisted of approximately 1.5M unlabeled tweets by conducting a query based on traffic-related Twitter accounts and keywords.
- An approach was proposed to provide a stronger word embedding representation, which used generic and domain-specific word embeddings, together.
- The classification of traffic-related tweets with deep learning architectures is commonly done by a small number of classes, like two or three. In this study, to detect traffic-related events in more detail, classification has been made under 8 different classes.

The remaining parts of this article are organized as follows. Related works are given in Section 2; details of corpora and datasets are given in Section 3; Methodology is given in Section 4; Experimental results are given in Section 5, and Section 6 contains the Conclusion.

2. RELATED WORKS

Millions of users share their daily lives on social media platforms, and one of the main issues is traffic-related events [6]. Users, immediate sharing of traffic-related events makes these platforms an important source of data for real-time detection of traffic-related events [13].

A tweet contains a lot of information such as user profile, publishing time and location information, as well as text information. Therefore, text information or non-textual features can be used in the analysis of tweets [9]. However, tweet text stands out as the most prominent feature [9].

In one of the pioneering studies on the detection of traffic events from tweets [13], proposed a three-stage approach to the real-time display system, which including getting tweets based on various search criteria, pre-processing tweets using text mining methods, and finally classification of tweets. In this context, two different datasets prepared, and it was aimed to detect traffic events from tweets. The matter considered as a binary classification problem in the first dataset and tweets grouped under two class as traffic-related and non-traffic-related. The first dataset consists of a total of 1330 tweets. Of these tweets sharing in a balanced way, 665 of them traffic-related and 665 of them non-traffic-related tweets. On the other hand, the second dataset composed of 3 different classes. These classes are external events (football matches, concerts, etc.), traffic jams or accidents, and finally non-traffic-related tweets. Similarly, it is created a balanced way and consists of 999 tweets each class include 333 tweets. On the other hand, SMMs unstructured and irregular texts, that contain abbreviations, misspellings, grammatical errors, non-dictionary words and extremely short in their nature [13][14]. Therefore, detecting events

from SMMs is a much more challenging task than determining from well-formed traditional media such as blogs and emails [13][15]. For these reasons, they used text mining techniques to analyze the information these techniques derived from the fields of data mining, machine learning, statistics and natural language processing [13]. The received data, via the Twitter API, was cleaned in the pre-processing stage by using the regular expression filter to clear information such as user name, time information, and hashtag. After this stage, tokenization, stop-words removing, stemming and stem filtering procedures were performed respectively in the remaining words and feature vector representations were made by using uni-gram and bi-gram. In the last stage, the classification of tweets performed by using support vector machines (SVM). As a result, the classification performance of 95.75% obtained in the first dataset, consisting of a binary classification problem. In the last stage, tweet classification performed by support vector machines (SVM). Classification of three class dataset, the success rate decreased by approximately 7% and realized as 88.89%. Nowadays, social media platforms host not only ordinary users but also many other users such as public institutions, large media companies and government elders. The shares of such users contain less noisy data compared to ordinary users. That is to say, spelling and grammar rules are relatively more respected. Since the data generated by such sources are more reliable and contain very little noise, so some studies directly take into account these types of user messages in SMMs [16]. In this respect, they proposed a methodology for the interpretation of traffic tweets, considering only the shares of public institutions and news sites on their Twitter accounts [16]. Accordingly, traffic-related events defined in six different classes, these classes include restriction, accident, vehicle failure, traffic situation, weather and other unclassified traffic-events. They achieved %73 success rate in their tests. The main source of motivation develops a tool for tracking truck fleets to provide more efficient fleet management with the obtained data, moreover reduced costs and ensure timely delivery of products to customers. In this direction, they conducted tests on medium-sized liquid gas and large-scale fuel distribution companies, consisting of 500 and 5000 trucks, respectively. Noted that the proposed model reduced fuel delivery time, and consequently, customer satisfaction increased. The biggest problem with the used approach is that they only consider institutional Twitter accounts. Therefore, shared data from ordinary users not taken into consideration. Ordinary user data contains a high level of noise, but it's very critical for real-time event detection. Because maybe serious time differences between the actual time of the event and to reach the institutional accounts. This cause the system to deviate from its real purpose. While the system can successfully detect the event through institutional accounts, the event may have already occurred and ended. Therefore, provide all users data is an important issue in terms of spatial and temporal representation capability.

Rather than detecting all traffic events, directly identifying more specific sub-issue, such as traffic accident or vehicle failure, also one of the significant tasks at times. In particular, traffic accidents can cause significant traffic congestion at an unexpected time. In this direction, studies have been conducted to identify only traffic accidents, from social media data [17]. In this context, vehicle accidents classified into three classes [17], these groups collision, immobilized

vehicle and vehicle fire. In their study [17], they primarily identify the words commonly used in real news texts together with the accident. From this, formed a keyword series of 100 words, including police, road, lane and dead. In the extraction of keywords, words that are commonly used linguistically, expressions indicating the characteristics of the event and giving information about the geographical location not included. The second step was to identify the tweets associated with these keywords. In this direction, tweets filtered according to keywords and then filtered tweets randomly selected and labelled whether they related to traffic accidents. In line with this labelling, the most commonly used words in the accident-related tweets determined. As a result, they obtained a dataset of 900 tweets associated with the accident, the entire dataset comprised of approximately 20 thousand tokens. The proposed approach mainly based on identifying keywords related to traffic accidents. In this context, the tokens having the highest TF-IDF index were selected and used as a feature after the roots found. Also, words that not have a significant linguistic meaning and not indicate a notable event removed. In a more comprehensive study following this [2], a recursive data collection process was used to create a dictionary containing traffic-related words. With the Semi-Naïve-Bayes method, the tweets classified into two groups: traffic-related and non-traffic-related. Afterwards, related to traffic events tweets classified into 5 different categories such as accidents, road works and obstacles by using supervised latent Dirichlet allocation.

Traffic congestion on highways is one of the significant problems of big cities. Many sensors can be used to assess traffic congestion, as indicated in the above sections. However, another approach is whether or not congestion can be evaluated from data obtained from Twitter [18]. In the study conducted to evaluate traffic congestion with Twitter messages [18], they proposed a model to determine the level of traffic congestion using different types of Twitter data and C4.5 decision trees. With the proposed approach, they tried to estimate the extent of the traffic congestion level over the next 30 minutes. The used features are, day of the week, hours of the day, minutes and tweet density. To determine the intensity of the tweets, they used predetermined Twitter accounts and tweets tagged with location information. They collected 1821 tweets related to road names and traffic conditions and separated them according to periods. As a result, they classified traffic congestion status at 3 different congestion levels and obtained a value of 0.892 on the average F1 score.

In recent years, deep learning methods have successfully applied in many areas such as metro passenger flow prediction [19], speed prediction on traffic networks [20], and sound event classification [21]. Similarly, studies on the use of deep learning methods have been started to detect traffic events [9][10] in both studies, classification procedures performed with deep learning architectures by using word embeddings instead of bag-of-words. In the first study, two separate word embeddings models used for classification [9], which are word2vec and FastText. In the proposed models, publicly available word vectors used for word2vec and Wikipedia data used to obtain FastText model. In the other study [10], the word2vec model obtained by using 3 billion microblog data covering 1 million 70 thousand user accounts.

In general, the determination of traffic events from text data in SMMs, the bag-of-words method or the determination of keywords based on statistical approaches has widely used. Feature vectors obtained by these methods generally classified by methods such as SVM and decision trees. However, in the recent studies, word embeddings approaches, which significantly reduce the vector space dimension of the words, have been used to obtain the features, and these features have been classified using deep learning architectures. Word embeddings and deep learning based methods provided better results than traditional methods [9][10]. However, it is very important to provide effective representation in the stage of obtaining word embeddings. Problems such as abbreviations, conscious/unconscious misspellings and the use of diacritic characters due to mobile device usage frequently encountered in SMMs. Besides, both words and abbreviations can have different meanings in different domains. For this reason, in this study, proposed the approach of creating a domain-specific word embeddings for the traffic area obtained from a large amount of unlabelled data based on keyword and key user. Also, proposed the approach of obtaining a feature vector using generic and domain-specific word embeddings together.

3. CORPORA & DATASETS

This section provides information about traffic datasets and corpora created within the scope of the study. Firstly, two traffic datasets, which are composed of Turkish SMMs, are introduced for the classification of traffic data. Afterwards, the information provided about the corpora used in obtaining word representations.

3.1. Traffic Datasets

Within the scope of this study, two datasets consisting of Turkish tweets prepared for the classification of traffic-related tweets. In the first of these datasets, the issue considered as a binary classification problem and tweets divided into two different classes: traffic-related and non-traffic-related. Thus, it is aimed to evaluate whether the tweet is related to traffic. On the other hand, in the other dataset, the issue is dealt with as a multi-class classification problem consisting of 8 different categories to more clearly identify traffic-related situations such as accidents, road works, external events and weather conditions.

Table 1: Keywords used to obtain traffic-related tweets.

sürücü (driver)	kaza (accident)	yol (way)
otoban (highway)	otoyol (motorway)	kavşak (intersection)
köprü (bridge)	arıza (failure)	tünel (tunnel)
trafik (traffic)	araç (vehicle)	şerit (lane)
yoğunluk (density)	araba (car)	çalışma (work)

yanyol (side road)	konvoy (convoy)	hasarlı (damaged)
kuyruk (tailback)	kaldırım (sidewalk)	gişe (tollbooth)

Twitter REST API used in the creation of the dataset and performed a query based on the keyword. Keywords consist of frequently mentioned words in traffic news. Table 1 lists the keywords used in queries. An "or query" used, which contains all 21 words. Therefore, each tweet in the dataset includes at least one of these words.

The dataset used in the binary classification task is a balanced dataset of 2000 tweets, 1000 of which are traffic-related and 1000 of which are non-traffic-related. Moreover, Tweets of multi-class classification task obtained from binary classification datasets. Here, traffic-related tweets are divided into seven separate categories to identify traffic-related special cases further. Also, the eighth group consists of non-traffic-related tweets. The classes used in the multi-class classification task are:

- Road Construction Maintenance and Repair Works
- External Events Causing Traffic
- Weather Conditions
- Vehicle Failure
- Driver Error
- Accident
- General
- Non-traffic-related

Tweets reported on planned or unplanned road works included in the road construction maintenance and repair works class. Traffic tweets related to external events such as meetings, concerts and matches added to the external events causing traffic. The weather conditions include traffic tweets with meteorological conditions such as fog, rain and snow. In the vehicle failure heading, there are tweets concerning the vehicle malfunctions reported by the users. The driver error group includes shared tweets on lane violation, faulty parking and speed violation. The accident class tweets include tweets related to traffic accidents. Finally, tweets in the general class include traffic congestions related shares that do not indicate a special case. On the other hand, in this dataset, only 200 non-traffic tweets were included. This is to prevent excessive unbalanced data between classes. Table 2 shows how many tweets are in each class. Besides, the dataset created with at least 100 tweets from each class.

The authors of the article manually label each tweet. Tweets that both authors agreed in their class included in the study.

Data Class	**Number of Tweets**
Road Construction Maintenance and Repair Works	100
External Events Causing Traffic	100
Weather Conditions	100

Table 2: Class-based distribution of the number of tweets of the 8-class traffic dataset.

Vehicle Failure	282
Driver Error	100
Accident	119
General	199
Non-traffic-related	200

3.2. Corpora

Word embeddings are one of the important parameters that seriously affect the classification performance. Word embeddings, obtain vector representations of words by using the corpus as input. Accordingly, the vector similarities of the words in the corpus increases or decreases depending on the state of coexistence. Therefore, the corpus is significant to ensure the correct representation ability. The importance of choosing the right word embeddings becomes even more critical, especially due to the considerable increase in noise level in user-generated content. Word embeddings obtained by using more formal spelling corpora, such as Wikipedia data or text on news sites, are successful in representing the general meaning of words, while there are limitations in terms of both coping with noise in user-generated texts and representing domain-specific meanings of words. However, in domain-specific word embeddings obtained from the user-generated noisy text, the words around the wrong and correct spellings of a word are substantially similar. As shown in Table 3, for word embeddings obtained from unlabeled data specific to the domain used in this study, for the words of the "şoför (chauffeur)" and one of the commonly used misspelling forms of "şöför (chauffeur)", 21 of the 50 words that most closely resemble are the same. In other words, the words collected around the misspelling forms are very similar to the words in the correct spelling form. In this case, the use of domain-specific word embeddings from large unlabelled data makes it very advantageous in terms of classification performance.

Table 3: Common similar words for "şoför" and "şöför" spellings. (* incorrectly spelled)

acemi (noice)	muavin (assistant)	şoförü (chauffeur of)
aracı (his/her vehicle)	müşteri (customer)	şoförün (chauffeur 's)
bayan (lady)	neyinpeşindeacaba* (what's he up do)	şöförü* (chauffeur of)
direksiyonda (at the wheel)	sürücü (driver)	taksi (taxi)
kadın (woman)	sürücüleri (drivers)	taksici (cab driver)
kullanan (user)	şoförler (chauffeurs)	taksiciler (cab drivers)
motorsiklet (motorcycles)	şoförleri (their chauffeurs)	yolcu (passenger)

On the other hand, the types of misspelling encountered in social media can vary greatly. Diacritic character problem is one of the most common problems in SMMs due to keyboard layout in mobile devices [12]. Diacritics are some signs that are added to the letters and change their voice and used in many languages such as Turkish, French, Greek, Hungarian and Spanish. The problem of diacritic characters is the use of ASCII (American Standard Code for Information Interchange) equivalents instead of diacritic characters. Also, the diacritic character problem often encountered in more than one word in the sentence. About this situation, Table 4 shows similar words for the "araç (vehicle)" word and ASCII form of this word ("arac"), which obtained in the same way as Table 3. While the same words gathered around the wrong and correct spelling in "şoför" and "şöför", the situation differs slightly for the diacritic character problem. As shown in Table 4, the words similar to the ASCII form of the "araç", also are in ASCII forms. For example, the "minibus (minibus)" is similar to the "araç", while the ASCII form "minibus" is similar to the "arac". That is, while the correct forms of the words are gathered around the correct spelling of the "araç", the ASCII forms are collected around the spelling of the "arac". Moreover, both tables show one meaning of the words specific to the traffic field. However, many words in the tables have different meanings such as "araç (tool)" and "aracı (middleman)"

Table 4: Common similar words for "araç" and "arac" spellings [22].

Common similar words for "arac"	**Common similar words for "araç"**
araci	aracı (his/her vehicle)
araclar	araçlar (vehicles)
aracin	aracın (vehicle's)
araclarin	araçların (of vehicles)
otobus	otobüs (bus)
araclari	araçları (their vehicles)
arabanin	arabanın (car's)
araclara	araçlara (to vehicles)
suruculer	sürücüler (drivers)
minibus	minibus (minibus)
sirket	şirket (company)
arabalarin	arabaların (of cars)
aracta	araçta (on the vehicle)

Word embeddings from formal texts usually do not include misspelt forms of words. As a result, classification performance is negatively affected. In this direction, a corpus was created by collecting tweets related to the traffic domain. A similar method to the traffic dataset followed

when creating a domain-specific corpus and tweets containing the keywords found in Table 1 were collected. Besides, tweets shared from directly related to the traffic accounts such as @ibbtrafikradyo, @radyotrafik and @radyotrafik06 or tagged these accounts collected without using keywords. Unlike the traffic datasets, Scrapy data scraping tool developed in Python was used instead of Twitter APIs to obtain tweets. As a result, a total of 1.5M tweets were collected, including traffic-related keywords or directly related to traffic-related accounts. After the pre-processing of the domain-specific traffic tweets, vector representations of the words obtained by FastText method. During the pre-processing steps, all Uniform Resource Locator (URL) statements converted to text expressions in the form of "url". Apart from this, all punctuation cleared after all text converted to lower case. Finally, expressions are containing numeric characters omitted. Figure 1 shows the steps for creating a domain-specific word embeddings.
To create a generic Turkish corpus, the corpus* from the Hürriyet newspaper archives and the Turkish Wikipedia Dump (trwiki-20150121-pages-meta-current) dataset used in the study in [23] used as a corpus. Besides, some e-books have been added to the corpus to increase its representation power. As a result, a corpus created included in 1249914 different and total of 200822716 words.

4. METHODOLOGY

The approach used in the classification of traffic tweets in this study is based on the concatenation of generic and domain-specific word embeddings to provide a stronger representation capability. Many studies in the literature show that word embeddings obtained from large-sized corpora perform quite strongly in natural language processing tasks [24][25]. However, in many studies, domain-specific word embeddings have shown better results than generic word embeddings derived from large-sized corpora[11][26]. As mentioned in the above sections, both approaches have advantages and disadvantages over each other. In this respect, both generic and domain-specific word embeddings are used together in this study.

In word embeddings, each word represented by a fixed-length numeric vector. However, a word in the classification data may not be among the word embeddings. In this case, it is possible to represent the word with a vector to be generated randomly or with a vector consisting only of zeros. In this study, the use of a vector consisting of zeros adopted. On the other hand, in this study, the vector representations of the words are obtained by combining two separate vectors, as shown in Figure 1. Besides, the detailed stages of the proposed traffic event detection system shown in Figure 2. Hence, L_d is the vector length of the domain-specific word embedding, and L_j is the vector length of the generic word embedding. In this case, it is possible to define the vector length L_w of each word in the classification data, as follows:

* http://gurmezin.com/derlemtr-projesi/. Access time: 19.08.2019

$$L_w = L_d + L_j \quad (1)$$

So, if a word in the classification data not included in both word embeddings, it is represented by a completely zero vector. Otherwise, only the portion of the concatenated vector that not represented in the word embedding consists of zeros.

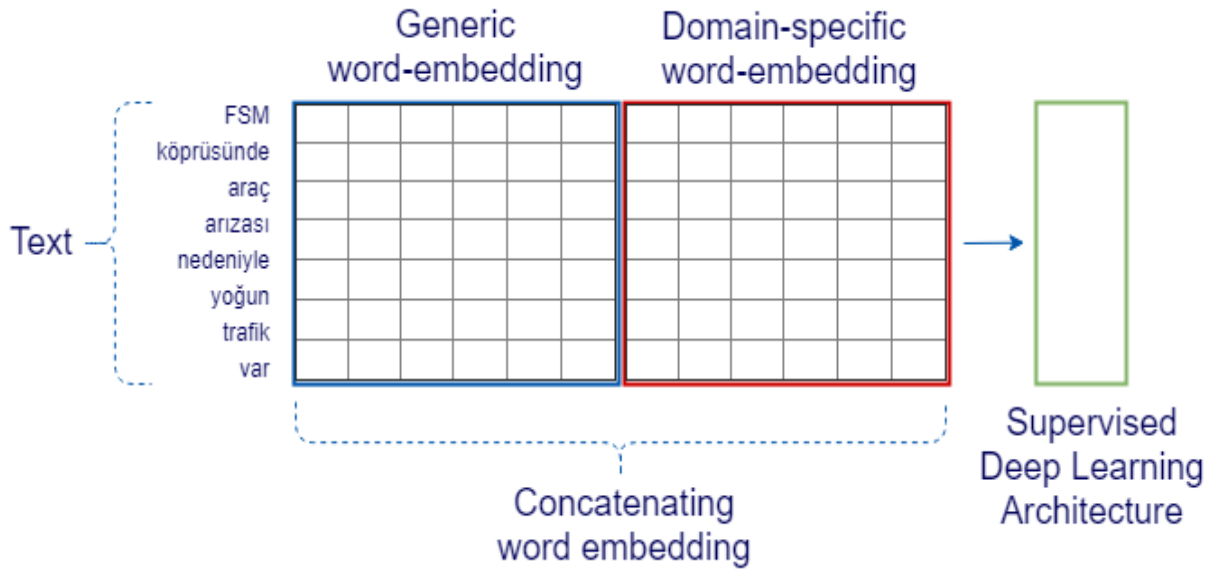

Figure 1: Concatenation of two different word embedding. ”FSM köprüsünde araç arızası nedeniyle yoğun trafik var.(There is heavy traffic on the FSM bridge due to vehicle failure.)”

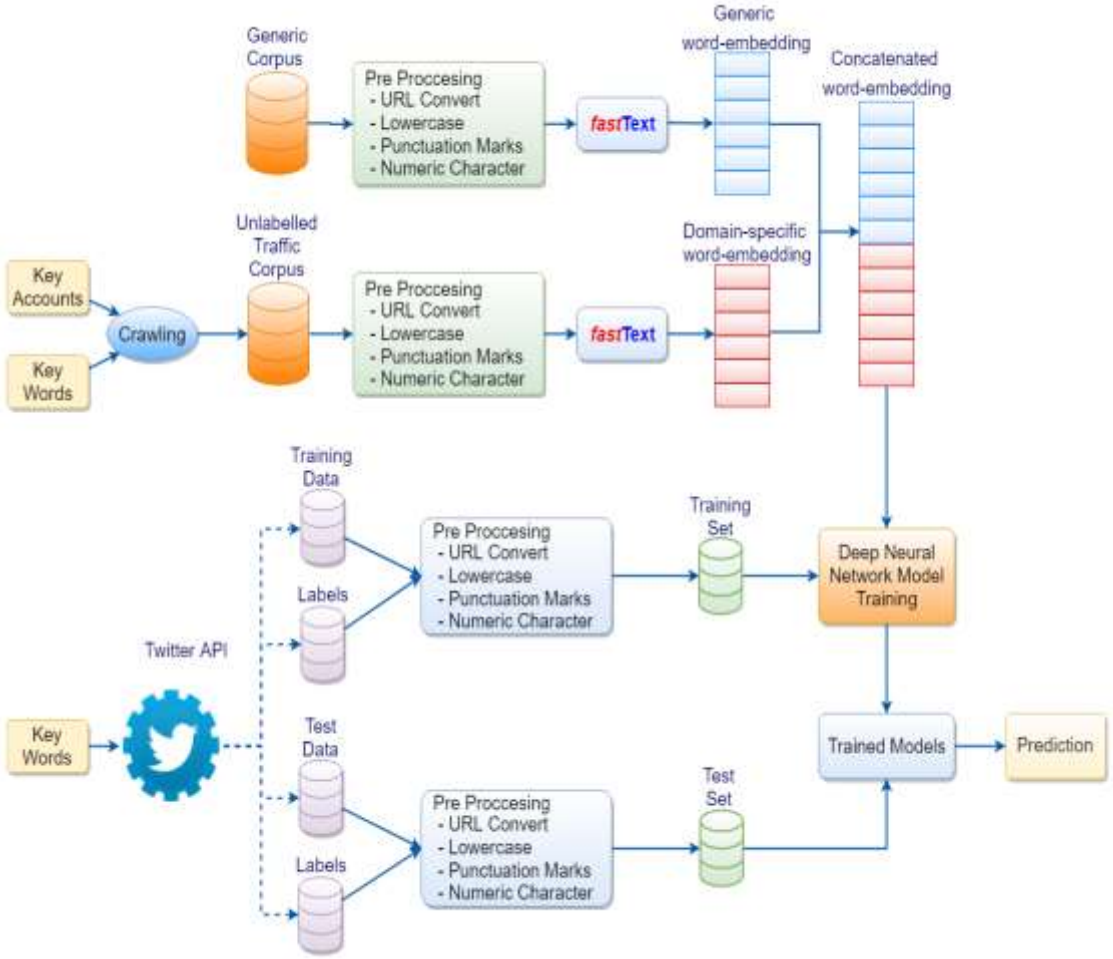

Figure 2: Block diagram of the traffic event detection system.

4.1. FastText

FastText is an extension of word2vec, which was recommended by Facebook in 2016 and treats each word as n-gram characters [27]. Word2vec models ignore the internal structure of words, assigning a separate feature vector for each word, whereas FastText represents each word as a bag of character n-grams. Then, the vector representation of a word expressed as the sum of its n-gram vectors. Specifically, FastText is capable of improving vocabulary representations for morphologically rich languages that contain verbs and nouns in a variety of forms that may rarely occur in the training corpus. FastText has been shown to be more accurate than word2vec vectors in many areas [28][29].

4.2. Convolutional Neural Networks

CNN, a typical multiple layered neural network model, was first proposed for computer vision problems [30] and has been used successfully in many image-related applications. CNNs are largely similar to feedforward neural networks but show differences in the connection between neurons in adjacent layers. CNN models commonly formed by convolution, pooling and fully connected layers.

The filter size and the generated number of maps are used to identify the convolution layer. This layer is the most basic and important unit that forms the CNN. The basic starting point of the convolution layer is that the image of an object is independent of where it is on the image, and under this idea, neurons are connected to only a small part of the inlet and extend across the entire depth of the data applied to the inlet. These approaches to learning the local response by CNNs enable them to capture abstract representations of n-grams through convolutional filters. This has led CNNs to be a great success in natural language applications, including text classification. Then, in the forward propagation phase, dot multiplication is performed between the input data and the filters and a 2-dimensional activation map is created. However, the information required to learn the CNN model is generally nonlinear, but the convolution process is a method involving linear operations such as matrix multiplication and addition. For this reason, the CNN model non-linearity is increased by applying a non-saturation activation function $f(x) = \max(0,\ x)$ with ReLU (Rectified Linear Units). Also, another important layer is the pooling layer in which down-sampling operations performed. There are many pooling methods used in the literature, but the two most commonly used methods are maximum pooling and average pooling. In this study, maximum pooling method used. In this method, the input divided into non-overlapping rectangles, and only the maximum value taken from each sub-part. The neurons in the fully connected layer are connected to each neuron in the layers before and after this layer, as in traditional artificial neural networks, and can be used in the output layer with softmax or another classifier.

As mentioned above, the convolution process is the most basic unit of CNN, so we will explain the convolution process in detail. w is a word represented by a V-dimensional vector, and N is the number of words so that the given sentence s is defined as follows:

$$s = w_1 w_2 \dots w_N \tag{2}$$

Since each word is represented by a V-dimensional vector, the sentence can be transformed to the NxV dimensional matrix easily, which is called input map. Also the filter F is a MxVdimensional matrix. Take into consideration just one filter p, it is possible to express the i^{th} element of the feature vector as follows:

$$c_i^p = f(conv([v(w_i) \dots v(w_{i+M-1}), F^p]) + b^p) \tag{3}$$

Where b^p is bias, F^p is filter matrix and finally f is the activation function. In this case $conv(.)$ operation can be defined as follows:

$$conv([v(w_i)\dots v(w_{i+m-1}), F^p]) = \Sigma_{r=1}^{M} F_{(r,;)}^{p} v^T(w_{i+r-1}) \quad (4)$$

If we substitute the equation 4 into equation 3, we obtain:

$$c_i^p = f\left(\Sigma_{r=1}^{M} F_{(r,:)}^{p} v^T(w_{i+r-1}) + b_p\right) \quad (5)$$

Finally, complete feature vector of the convolution layer with filter *p*, obtained as follows:

$$C^p = \left(c_1^p, c_2^p, \dots, c_{N-M+1}^p\right)^T \quad (6)$$

The multilayer structure obtained by combining different layers allows the network to perceive more abstract features and to discover more complex structures. The structure thus obtained enables the CNN to be successful in many applications such as image classification [31][32][33][34], facial expression recognition [35] and scene labelling [36].

4.3. Long-Short Term Memory

While the only input considered by traditional feedforward neural networks is the current samples to which they are exposed, Recurrent neural networks (RNNs) unlike these apply the existing samples as well as the samples they perceive over time. This gives RNNs a great advantage in learning historical experiences over time.

Let $x_i \in \mathbb{R}^d$, and input sequence given as $[x_1, x_2, \dots, x_k]$. Here, different examples may have different sequence lengths, and therefore the *k* value may vary. In each step of the RNN model, a hidden state is generated in the sequence $[h_1, h_2, \dots, h_k]$. Activation of the hidden state at time *t* is calculated as a function of the current input x_t and the revious latent state h_{t-1} as follows:

$$h_t = f(x_t, h_{t-1}) \quad (7)$$

RNNs have a repeating structure, unlike traditional feedforward neural networks. Utilizing this structure, the state information generated by the feedforward network is stored and reapplied to the network with the input information. In other words, RNNs have a memory that keeps what is calculated up to now.

Long-term short-term memory networks called LSTM are a special type of RNN that is capable of learning long-term dependencies. This model, which first proposed in the mid-90s[37], is now widely used. While it is aimed to store and transmit state information of the artificial neural network while processing on arrays in RNNs, it is not possible to transmit without disturbing long-term dependencies as a result of continuous processing of state information. In other words, while short term dependencies in the array transferred successfully, there is a problem in

transferring long term dependencies. LSTMs are designed to address the problem of long-term dependence.

All RNN networks consist of repetitive modules in the form of a chain. In standard RNNs, each of these modules generally consists of a tanh layer or a similar single layer. What distinguishes LSTMs from standard RNNs is that the internal structure of this module consists of 4 separate structures that interact with each other.

LSTM module consists of 3 separate gates. The names of these gates are, forget, input and output. Forget gate decides to how much of the information will be forgotten and how much of it should be transferred to the next stage. It has a sigmoid layer that produces a value between 0 and 1 for this process. 0 means that no part of the information will be transmitted, and 1 means that all information must be transmitted. It is possible to express the mathematical model of the forget gate as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{8}$$

The next step is to decide what information should be stored. At this stage, firstly, the 2nd Sigmoid layer, which is called the input gate, decides which values should be updated. The next tanh layer forms a vector of the new candidate values expressed as $\tilde{C}_t$, and then these two operations are combined. This process is expressed mathematically as follows:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{9}$$

$$\tilde{C} = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{10}$$

After this, the new status information of the memory cell must calculate. In this case, the new status information calculated as follows:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{11}$$

At the last stage, the output of the system h_t calculated. This is done at the output gate, and the output of the system h_t can calculate as follows:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{12}$$

$$h_t = o_t * tanh(C_t) \tag{13}$$

5. EXPERIMENTAL RESULTS

This section will evaluate the performance of the proposed model for detecting traffic events from Twitter messages using the datasets introduced in Section 3.1.

First, information about baseline methods, performance metrics, word embedding parameters, model parameters and training process will give. Then the test results obtained using different settings will be compared.

5.1. Experimental Setup

5.1.1. Baseline Methods

We compare the performance of the proposed model with several baselines methods that commonly used in text classification studies. Here, we prefer the methods commonly used for evaluation, especially in detecting traffic-related events from SMMs. For evaluation, we first use the bag-of-words representations obtained using the uni-gram or bi-gram properties after stop words and punctuation are removed [9][10][38]. Then we compare the results with the commonly used supervised learning algorithms linear SVM [9][10][38], k-nearest neighbor(KNN) and Multilayer Perceptron(MLP) [9][10]. We also test Linear SVM by using word embeddings as input.

5.1.2. Performance Metrics

The performance of the proposed model and the baseline methods are evaluated by four commonly used classification criteria. A true positive (TP) is a result in which the model correctly predicts the positive class. Similarly, a true negative (TN) is a result in which the model correctly predicts the negative class. On the other hand, a false positive (FP) is a result that the model mistakenly predicts a class as positive. And, a false negative (FN) is a result that the model incorrectly predicts a class as negative. As a result, performance metrics obtained as follows:

$$Accuracy=\frac{TP+TN}{TP+FN+FP+TN} \tag{14}$$

$$Precision=\frac{TP}{TP+FP} \tag{15}$$

$$Recall=\frac{TP}{TP+FN} \tag{16}$$

$$F_1=2\cdot\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}=\frac{2TP}{2TP+FP+FN} \tag{17}$$

All experiments were performed using 5-fold cross-validation, and the mean values and standard deviations of the results reported. 80% of the entire dataset is used for training, while the remaining 20% used for testing. In addition to this, the validation dataset formed by using a stratified random selection of 10% of the training dataset for the early-stopping procedure. If the successive 5 epoch classification performance does not increase, the training process terminated.

5.1.3. Learning Word Embedding Representations

The word embeddings used in the classification process obtained by FastText method. The basic length of the vector determined as 100. Moreover, obtained a word embeddings with varying lengths of a vector of 50, 75 and 125, and their classification performances evaluated. Another parameter used for word embeddings is the window size. The window size concept refers to the number of contexts to be considered on the right and left of the word. The window size determined as 5 for all models used for classification. Another parameter used in word embeddings is the minimum number of occurrences. The minimum number of occurrences controls the number of times the word has passed in the corpus, and no vector representation produced for words below the threshold. The minimum number of occurrences used as 1. That is, a vector representation produced for all words in the corpus. The initial learning rate of the system was 0.025. The learning rate is reduced linearly during the training process and the lowest value used as 0.0001. As the last hyperparameter, we determine the number of revolutions. The number of epoch refers to the number of times all data has been trained. The epoch number of the system determined as 5.

The 2-class dataset consists of 10798 unique tokens. The generated generic word embedding includes of 7777 corresponding to 72.02% of these tokens. That is 3021 tokens in the 2-class dataset not found in generic word embedding. Therefore, they represented by vectors consisting entirely of zeros. Domain-specific word embedding includes 10195 corresponding to 94.41%. A total of 603 tokens do not found in domain-specific word embedding. The word embedding created by combining both word embeddings has coverage of 95.53% with 10315 tokens. The number of tokens represented by the completely zero vector is 483. Its less than compared to other word embeddings. The 8-class dataset consists of 5887 unique tokens. The percentage of tokens in generic, domain-specific and concatenated word embeddings is 69.92%, 94.89% and 95.55%, respectively.

5.1.4. Model Parameters

Deep learning models contain a large number of hyperparameters, and there is no precise method of how to determine these parameters. On the other hand, one of the approaches used is to determine the hyperparameters by experimental methods. In this study, important hyperparameters determined by a series of experiments. Experiments performed on both datasets, and the values that provided the highest classification performance in both datasets used in the final model. In this respect, test performed for CNN model by using different values of filter number and size. As shown in Table 5 and Table 6, tests were performed for [5 300] values of the number of filters, while tests performed between [1 5] values within the filter size. For the dataset consisting of two classes, the highest classification performance was achieved with the number of filters 250 and filter size 1. On the other hand, as shown in Table 6, the highest classification performance was obtained for the dataset consisting of 8 classes with the filter numbers 250 and the filter size 1 and 2. In the final model, the number of filters 250 and filter size 1 were used,

which provides the highest classification performance in both datasets. Table 7 shows the parameters related to the final model.

Table 5: Average accuracy results for different filter size and filter numbers on the 2-class dataset.

		Number of filters								
		5	**10**	**20**	**40**	**80**	**100**	**200**	**250**	**300**
Filter size	**1**	84.7	89.5	91.65	91.4	92.3	92.3	92.15	**92.45**	91.75
	2	90	91.3	91.55	91.8	91.8	91.75	92.1	92.05	91.75
	3	90.65	91.45	91.45	91.9	92.15	91.85	91.95	92.4	91.95
	4	90.9	90.85	91.6	91.8	92	92	92	92.3	91.85
	5	91.15	91.75	91.75	92.15	91.65	91.75	92.15	92.3	91.65

Table 6: Average accuracy results for different filter size and filter numbers on the 8-class dataset.

		Number of filters								
		5	**10**	**20**	**40**	**80**	**100**	**200**	**250**	**300**
Filter size	**1**	63.17	68.00	82.67	86.50	88.25	89.00	89.08	**89.58**	88.67
	2	70.08	74.83	84.25	88.50	88.33	88.92	88.92	**89.58**	89.17
	3	69.75	76.25	84.50	87.92	88.08	88.92	89.17	89.50	88.42
	4	71.25	75.92	84.08	87.33	88.00	89.50	88.75	88.83	89.00
	5	71.17	76.33	84.00	85.75	87.00	87.92	88.25	88.92	88.42

Table 7: Parameters of the Best Performed CNN Model.

Layer	**Type**
0	Input(30, 200)
1	Conv(1, 250)(stride=1)
2	Max-pooling
3	Fully Connected(100)
4	Dropout(0, 2)
5	Fully Connected

One of the most important hyperparameters in the LSTM model is the number of units. Accordingly, tests performed on both datasets, similar to the CNN model, for different unit

numbers [25 200]. Table 8 show the classification achievements for 2 and 8-class datasets. In both datasets, the highest classification performance achieved with the number of units 100. Table 9 shows the parameters for the final model. The weights of the model were initiated using the approach proposed in [39].

Table 8: Average accuracy results for various unit counts on the 2-class and 8-class dataset.

		Number of units						
		25	**50**	**100**	**125**	**150**	**175**	**200**
Dataset	**2-class**	93.80	93.70	**95.15**	94.20	93.80	94.05	93.90
	8-class	89.58	90.25	**92.17**	91.83	91.75	92.08	91.67

Table 9: Parameters of the Best Performed LSTM Model.

Layer	**Type**
0	Input(30, 200)
1	Dropout(0, 2)
2	LSTM (100)
3	Fully Connected

5.1.5. Training Process

The architectures described in the above section are classified using binary cross-entropy for the 2-class dataset and categorical cross-entropy in the 8-class dataset as the loss function. Also, Adam optimizer was used to update model parameters during the back-propagation phase. Adam is an optimization technique that has been widely used in recent deep learning studies [40][Y20]. Similarly to [9][Y5], default parameters were used in the training phase, such as learning rate=0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$. Also, the maximum sequence length determined as 30. The dropout layer was used in both CNN and LSTM architecture, as shown in the above section. Also, an early-stopping mechanism was used to avoid overfitting. If the validation dataset classification performance did not increase 5 consecutive epochs, the training process terminated.

5.2. Results and Discussion

In this section, firstly the performance of the proposed model is compared using generic, domain-specific and concatenated word embeddings with various word embedding lengths. Then, an evaluation of the proposed model with baseline methods presented. In the last stage, model performance evaluated based on subclasses.

5.2.1. Model Performance

Generic word embeddings pre-trained via large-sized corpus provide highly successful results in NLP tasks [9][24][25]. However, several factors may limit performance, such as the noisy structure of social media texts, words contain different meanings specific to the domains, and some of the domain-specific words in the classification data may not include in generic word embeddings. On the other hand, generic word embeddings are also very successful in representing the general meaning of words. Firstly, the effect of generic, domain-specific and concatenated word embeddings on the classification performance evaluated with CNN and LSTM networks. The evaluation process is carried out on two different datasets consisting of two and eight separate classes.

Figure 3 shows the classification performances and F1 scores obtained on the 2-class dataset using the LSTM model for different vector dimensions of generic, domain-specific, and concatenated word embeddings. The columns on the figure show the accuracy, and the F1 scores represented by lines. The expressions represented by 25, 50, 100 and 200 respectively indicate the vector length of generic and domain-specific word embeddings. Since concatenated word embedding obtained by combining domain-specific and generic word embeddings, the vector dimensions are equal to the total length of these two word embeddings. For example, in the case where the vector length of domain-specific and generic word representations is 25, the vector length of the concatenated word embedding is 50. Similarly, in cases where the vector dimensions are 100 for generic and domain-specific word embeddings, the concatenated word embedding length is 200. The highest accuracy rate was 95.15%, which obtained by concatenating generic and domain-specific word embeddings with vector dimensions of 100. The highest F1 score was also achieved by the same model, with 95.17%. Besides, domain-specific word embedding for all vector length have achieved a higher classification performance than generic word embedding. On the other hand, the most top classification performance for all vector length obtained with concatenated word embeddings.

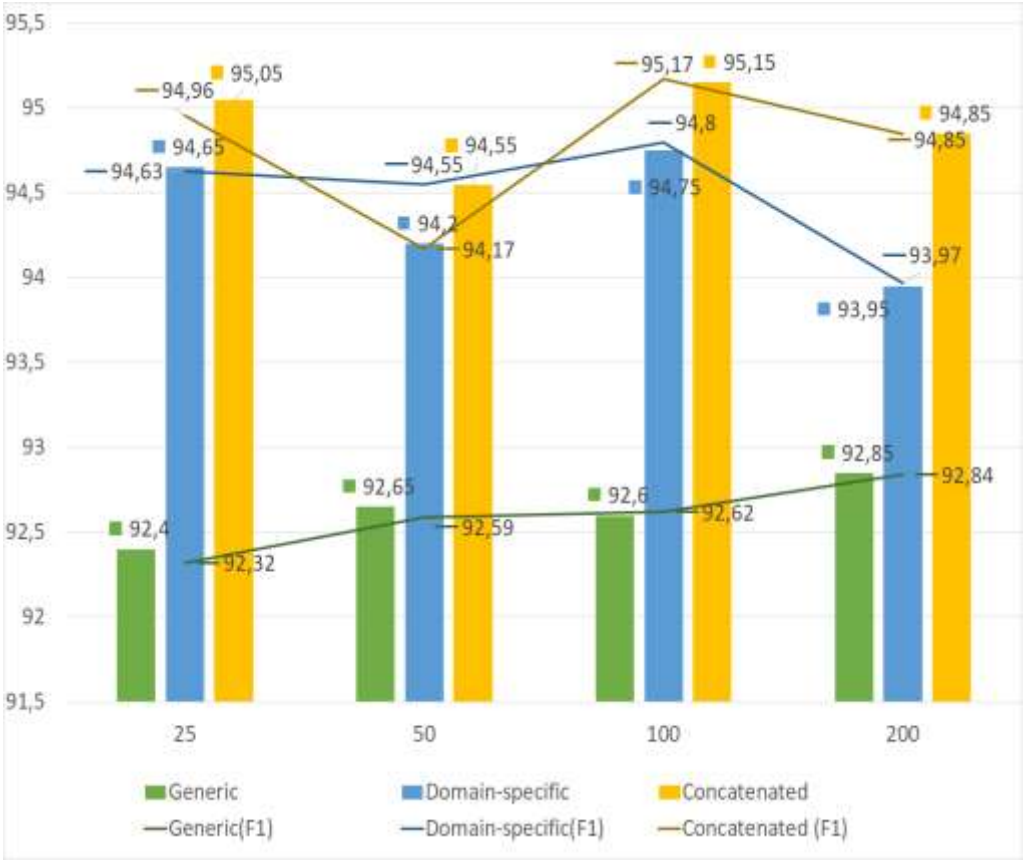

Figure 3: Accuracy and F-measure with various word embeddings models and vector length on the 2-class dataset.

The effect of generic, domain-specific, and concatenated word embeddings on classification performance was evaluated using CNN as well as LSTM. Table 10 shows the accuracy and F1 scores on the 2-class dataset with CNN and LSTM models for vector length 100. Of all three word embeddings, the highest performance achieved with the LSTM model. When the effect of three different word embeddings on CNN performance evaluated, it is seen that similar results obtained with LSTM model. The highest accuracy and F1 score for the CNN model achieved with concatenated word embedding of 92.45% and 92.56%, respectively. As in the LSTM model, the lowest accuracy and F1 values in the CNN model obtained with generic word embedding with 91.8% and 91.94%, respectively. In summary, domain-specific and concatenated word embeddings increased the classification performance by 0.15% and 0.65% in the CNN model according to generic word embedding. However, the rate of increase in classification performance remained lower compared to 2.15% and 2.55% increases in the LSTM model.

Table 10: Accuracy and F-measure of CNN and LSTM models, with various word embeddings models on the 2-class dataset.

	CNN		LSTM	
	Acc	**F1**	**Acc**	**F1**
Generic	91.8(1.08)	91.94(1.08)	92.6(1.19)	92.62(1.09)
Traffic	91.95(1.07)	92.08(1.06)	94.75(0.47)	94.8(0.36)
Concatenated	92.45(0.57)	92.56(0.47)	**95.15(0.52)**	**95.17(0.43)**

Figure 4 shows the accuracy and F1 scores of the tests performed on the 8-class dataset using the LSTM model and different word embeddings, similar to the 2-class dataset. Domain-specific word embeddings for all vector length showed higher performance than generic word embeddings. However, the concatenated word embeddings provided the highest performance for all vector length. The highest accuracy and F1 scores of all models were obtained by concatenating generic and domain-specific word embeddings with vector dimensions of 100, 92.17% and 91.79%, respectively.

As can be seen in Table 11, the classification performances obtained by CNN on the 8-class dataset are similar to the results of the 2-class dataset. The highest accuracy in CNN achieved through concatenated word embeddings. For concatenated word embeddings, 88.58% and 88.3% values obtained in performance and F1 scores, respectively. As with the 2-class dataset, the contribution of domain-specific and concatenated word embeddings to the performance of CNN remains lower than LSTM.

Table 11: Accuracy and F-measure of CNN and LSTM models, with various word embeddings models on the 8-class dataset.

	CNN		LSTM	
	Acc	**F1**	**Acc**	**F1**
Generic	87.67(1.35)	87.41(115)	88.42(2.31)	88.57(248)
Traffic	88.17(1.66)	87.99(1.39)	90.58(11)	89.66(1.37)
Concatenated	88.58(1.68)	88.3(1.41)	**92.17(11)**	**91.79(1.32)**

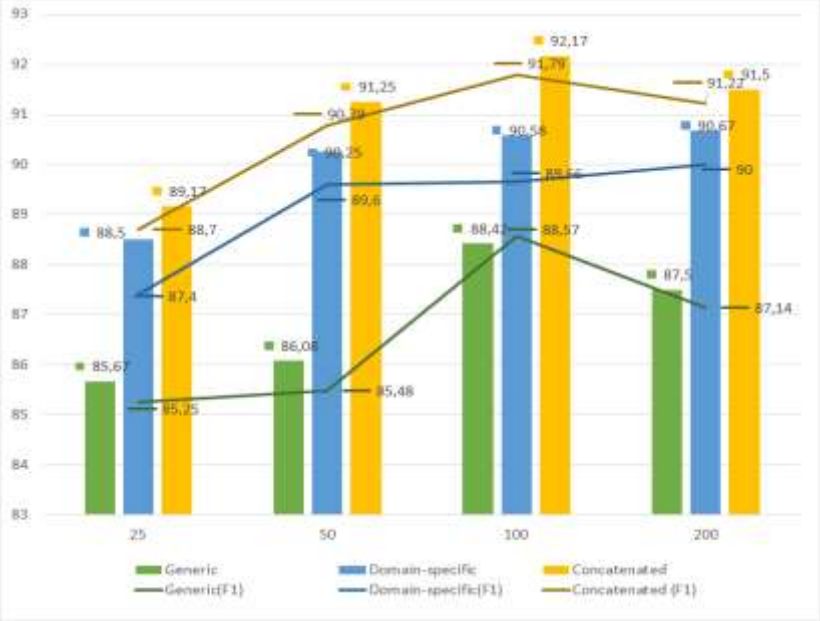

Figure 4: Accuracy and F-measure with various word embeddings models and vector length sizes on the 8-class dataset.

5.2.2. Comparison results

Table 12 shows the evaluation of the classification results of both 2-class and 8-class datasets with baseline methods. Each value represents the average of the scores obtained by applying 5-fold cross-validation. It also refers to the standard deviation of the values shown in parentheses. The features obtained by using TF-IDF with word-level uni-gram and bi-gram models classified with supervised learning algorithms such as SVM, KNN and MLP. Also, in the GenericWE / SVM model, SVM is used with generic word embedding. In both classification datasets, the highest values in terms of both accuracy and F1 scores obtained by using concatenated word embeddings and LSTM. With the proposed approach, the classification performance of the 2-class dataset was improved by 2.95%, equivalent to a relative error reduction of 37.82% compared to the closest result. On the 8-class dataset, a performance increase of 4.42% achieved, which corresponds to a relative error reduction rate of 36.08%.

Table 12: Performance comparison of the proposed model and baseline methods.

	2 - Class		**8 - Class**	
	Acc	**F1**	**Acc**	**F1**
Uni-gram / SVM	91.55(1.16)	91.3(1.5)	87.75(1.25)	87.75(1.1)

Bi-gram / SVM	84.45(1.39)	8160(1.55)	72.08(2.19)	7279(2.2)
GenericWE / SVM	79.8(1.08)	79.72(1.62)	55.92(2.62)	52.37(3.64)
Uni-gram / KNN	83.1(2.42)	84.82(2.6)	70.42(2.43)	71.96(2.67)
Uni-gram / MLP	92.2(0.84)	92.01(1.03)	77.08(2.83)	76.54(2.76)
Concatenated / LSTM	**95.15(0.52)**	**95.17(0.43)**	**92.17(1.1)**	**91.79(1.32)**

5.2.3. LSTM Model Performance Evaluation Based on Subclasses

In this section, the classification performance of the proposed model is evaluated based on each subclass for the 8-class dataset. The highest F1 and recall scores were obtained on the vehicle failure class, 97.07% and 97.92% respectively, as shown in Table 13. The highest precision value obtained in weather conditions class with 98.92%. On the other hand, the lowest F1 and precision values belong to the general class of 84.76% and 80.53% respectively. As mentioned in the above sections, the tweets in the general class that usually contain traffic congestion-related shares that do not indicate a special case. When the details examined, it is seen that the tweets belonging to this group mixed with all classes except weather conditions because of having many common words about traffic.

Table 13: Proposed model performance evaluation, based on subclasses for the 8-class dataset.

	Recall	**Precision**	**F1**
Road Construction Maintenance and Repair Works	90.32	87.5	88.89
External Events Causing Traffic	86.92	93.0	89.86
Weather Conditions	92.0	**98.92**	95.34
Vehicle Failure	**97.92**	96.25	**97.07**
Driver Error	92.73	95.33	94.01
Accident	90.83	92.37	91.6
General	89.47	80.53	84.76
Non-traffic-related	90.52	94.09	92.27

6. CONCLUSION

In this study, proposed an approach that uses generic and domain-specific word embeddings together to detect traffic-related events from Twitter messages. Accordingly, two datasets prepared. In the first dataset, tweets divided into two classes: traffic-related and non-traffic-related. In the other dataset, 8 different classes were created to detect more specific situations such as accidents, road works, driver errors and weather conditions. Besides, prepared a corpus including approximately 1.5 M tweets to obtain domain-specific word embeddings. Vector

representations of words attain from the corpora using FastText. Then, the word embeddings obtained by concatenating generic and domain-specific word embeddings are classified with CNN and LSTM architectures. Concatenated word embeddings performed better classification performance than both generic and domain-specific word embeddings. In the LSTM model, a concatenated word embedding provide a classification performance of 95.15% and 92.17% in the 2-class and 8-class datasets, respectively. Furthermore, when compared with state-of-the-art methods, the classification performance of the 2-class dataset was improved by 2.95%, which corresponds to a relative error reduction of 37.82% to the closest result. On the 8-class dataset, a performance increase of 4.42% achieved, which corresponds to a relative error reduction rate of 36.08%.

REFERENCES

[1] C. Gutierrez, P. Figuerias, P. Oliveira, R. Costa, R. Jardim-Goncalves, Twitter mining for traffic events detection, in: Proceedings of the 2015 Science and Information Conference, SAI 2015, Institute of Electrical and Electronics Engineers Inc., 2015: pp. 371–378. doi:10.1109/SAI.2015.7237170.

[2] Y. Gu, Z. Qian, F. Chen, From Twitter to detector: Real-time traffic incident detection using social media data, Transportation Research Part C: Emerging Technologies. 67 (2016) 321–342. doi:10.1016/j.trc.2016.02.011.

[3] U.S Department of transportation, Traffic Congestion and Reliability : Trends and Advanced Strategies for Congestion Mitigation., 2012. doi:10.1080/1063073012005218.

[4] K. Fu, C.T. Lu, R. Nune, J.X. Tao, Steds: Social Media Based Transportation Event Detection with Text Summarization, in: IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, Institute of Electrical and Electronics Engineers Inc., 2015: pp. 1952–1957. doi:10.1109/ITSC.2015.316.

[5] G. Leduc, Road Traffic Data : Collection Methods and Applications, EUR Number: Technical Note: JRC 47967. JRC 47967 (2008) 55. doi:JRC 47967 - 2008.

[6] S. Xu, S. Li, R. Wen, Sensing and detecting traffic events using geosocial media data: A review, Computers, Environment and Urban Systems. 72 (2018) 146–160. doi:10.1016/j.compenvurbsys.2018.06.006.

[7] A. Doan, R. Ramakrishnan, A.Y. Halevy, Crowdsourcing systems on the world-wide web, Communications of the ACM. 54 (2011) 86–96. doi:10.1145/1924421.1924442.

[8] X. Zheng, W. Chen, P. Wang, D. Shen, S. Chen, X. Wang, Q. Zhang, L. Yang, Big Data for Social Transportation, IEEE Transactions on Intelligent Transportation Systems. 17 (2016) 620–630. doi:10.1109/TITS.2015.2480157.

[9] S. Dabiri, K. Heaslip, Developing a Twitter-based traffic event detection model using deep learning architectures, Expert Systems with Applications. 118 (2019) 425–439. doi:10.1016/j.eswa.2018.10.017.

[10] Y. Chen, Y. Lv, X. Wang, L. Li, F.Y. Wang, Detecting Traffic Information From Social Media Texts With Deep Learning Approaches, IEEE Transactions on Intelligent Transportation Systems. (2018). doi:10.1109/TITS.2018.2871269.

[11] Z. Jiang, L. Li, D. Huang, L. Jin, Training word embeddings for deep learning in biomedical text mining tasks, in: Proceedings - 2015 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2015, Institute of Electrical and Electronics

Engineers Inc., 2015: pp. 625–628. doi:10.1109/BIBM.2015.7359756.

[12] G. Eryiğit, D. Torunoğlu-Selamet, Social media text normalization for Turkish, Natural Language Engineering. 23 (2017) 835–875. doi:10.1017/S1351324917000134.

[13] E. D'Andrea, P. Ducange, B. Lazzerini, F. Marcelloni, Real-Time Detection of Traffic from Twitter Stream Analysis, IEEE Transactions on Intelligent Transportation Systems. 16 (2015) 2269–2283. doi:10.1109/TITS.2015.2404431.

[14] F. Atefeh, W. Khreich, A survey of techniques for event detection in Twitter, Computational Intelligence. 31 (2015) 133–164. doi:10.1111/coin.12017.

[15] R. Parikh, K. Karlapalem, ET, in: Proceedings of the 22nd International Conference on World Wide Web - WWW '13 Companion, ACM Press, New York, New York, USA, 2013: pp. 613–620. doi:10.1145/2487788.2488006.

[16] F.C. Albuquerque, M.A. Casanova, H. Lopes, L.R. Redlich, J.A.F. De Macedo, M. Lemos, M.T.M. De Carvalho, C. Renso, A methodology for traffic-related Twitter messages interpretation, Computers in Industry. 78 (2016) 57–69. doi:10.1016/j.compind.2015.10.005.

[17] Z. Zhang, Q. He, J. Gao, M. Ni, A deep learning approach for detecting traffic accidents from social media data, Transportation Research Part C: Emerging Technologies. 86 (2018) 580–596. doi:10.1016/j.trc.2017.11.027.

[18] S. Wongcharoen, T. Senivongse, Twitter analysis of road traffic congestion severity estimation, in: 2016 13th International Joint Conference on Computer Science and Software Engineering, JCSSE 2016, Institute of Electrical and Electronics Engineers Inc., 2016. doi:10.1109/JCSSE.2016.7748850.

[19] Y. Liu, Z. Liu, R. Jia, DeepPF: A deep learning based architecture for metro passenger flow prediction, Transportation Research Part C: Emerging Technologies. 101 (2019) 18–34. doi:10.1016/j.trc.2019.01.027.

[20] Z. Zhang, M. Li, X. Lin, Y. Wang, F. He, Multistep speed prediction on traffic networks: A deep learning approach considering spatio-temporal dependencies, Transportation Research Part C: Emerging Technologies. 105 (2019) 297–322. doi:10.1016/j.trc.2019.05.039.

[21] I. Ozer, Z. Ozer, O. Findik, Noise robust sound event classification with convolutional neural network, Neurocomputing. 272 (2018). doi:10.1016/j.neucom.2017.07.021.

[22] Z. Ozer, The effect of normalization on the classification of traffic comments, Karabuk University, 2019.

[23] A. Arslan, DeASCIIfication approach to handle diacritics in Turkish information retrieval, Information Processing and Management. 52 (2016) 326–339. doi:10.1016/j.ipm.2015.08.004.

[24] Y. Kim, Convolutional neural networks for sentence classification, in: EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 2014: pp. 1746–1751. http://arxiv.org/abs/1408.5882 (accessed August 19, 2019).

[25] C. Zhou, C. Sun, Z. Liu, F.C.M. Lau, A C-LSTM Neural Network for Text Classification, (2015). http://arxiv.org/abs/1511.08630 (accessed August 19, 2019).

[26] N. Mahmoudi, P. Docherty, P. Moscato, Deep neural networks understand investors better, Decision Support Systems. 112 (2018) 23–34. doi:10.1016/j.dss.2018.06.002.

[27] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching Word Vectors with Subword Information, Transactions of the Association for Computational Linguistics. 5 (2017) 135–146. doi:10.1162/tacl_a_00051.

[28] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, Bag of tricks for efficient text classification, in: 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference, 2017: pp. 427–431. http://arxiv.org/abs/1607.01759 (accessed August 19, 2019).

[29] X. Yang, C. Macdonald, I. Ounis, Using word embeddings in Twitter election classification, Information Retrieval Journal. 21 (2018) 183–207. doi:10.1007/s10791-017-9319-5.

[30] Y. Lecun, L. Bottou, Y. Bengio, P. Ha, Gradient-Based Learning Applied to Document Recognition 1 Introduction, Most. (1998) 1–75.

[31] A. Krizhevsky, I. Sutskever, G.E. Hinton, S. Levine, C. Finn, T. Darrell, P. Abbeel, P. Pastor, A. Krizhevsky, D. Quillen, V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, R.S. Sutton, A.G. Barto, S. Ioffe, C. Szegedy, ImageNet Classification with Deep Convolutional Neural Networks Alex, Proceedings of the 31st International Conference on Machine Learning (ICML-14). 3 (2012) 322. doi:10.1007/s13398-014-0173-7.2.

[32] J. Yu, D. Tao, M. Wang, Adaptive hypergraph learning and its application in image classification, IEEE Transactions on Image Processing. 21 (2012) 3262–3272. doi:10.1109/TIP.2012.2190083.

[33] F. Nian, T. Li, Y. Wang, M. Xu, J. Wu, Pornographic image detection utilizing deep convolutional neural networks, Neurocomputing. 210 (2016) 283–293. doi:10.1016/j.neucom.2015.09.135.

[34] S. Yu, S. Jia, C. Xu, Convolutional neural networks for hyperspectral image classification, Neurocomputing. 219 (2017) 88–98. doi:10.1016/j.neucom.2016.09.010.

[35] A.T. Lopes, E. de Aguiar, A.F. De Souza, T. Oliveira-Santos, Facial expression recognition with Convolutional Neural Networks: Coping with few data and the training sample order, Pattern Recognition. 61 (2017) 610–628. doi:10.1016/j.patcog.2016.07.026.

[36] P.O. Pinheiro, R.C. Com, R. Collobert, Recurrent Convolutional Neural Networks for Scene Parsing, ArXiv Preprint ArXiv …. cs.CV (2013). http://arxiv.org/abs/1306.2795%5Cnfile:///Users/poliveirap/Dropbox/Library.papers3/Articles/2013/Pinheiro/arXiv preprint arXiv … 2013 Pinheiro.pdf%5Cnpapers3://publication/uuid/6D3F5B9B-4298-4E7E-A14C-52D4048F5573.

[37] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Computation. 9 (1997) 1735–1780. doi:10.1162/neco.1997.9.8.1735.

[38] S. Carvalho, L. Sarmento, R.J.F. Rossetti, Real-Time Sensing of Traffic Information in Twitter Messages, Proceedings of the IEEE ITSC 2010 Workshop on Artificial Transportation Systems and Simulation (ATSS2010), Madeira Island, Portugal. (2010) 19–22.

[39] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. (2010). http://www.iro.umontreal.

[40] D.P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, (2014). http://arxiv.org/abs/1412.6980 (accessed August 19, 2019).