

# Traffic Event Detection as a Slot Filling Problem

Xiangyu Yang<sup>a,b,\*</sup>, Giannis Bekoulis<sup>a,b</sup>, Nikos Deligiannis <sup>a,b</sup>

<sup>a</sup>*ETRO, Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussels, Belgium*  
<sup>b</sup>*imec, Kapeldreef 75, B-3001 Leuven, Belgium*

---


## Abstract

Social media platforms, such as Twitter, can be used to extract information related to traffic events. Previous works focused mainly on classifying tweets into predefined categories (i.e., traffic or non-traffic) without many details of traffic events. However, extracting traffic-related fine-grained information from tweets is essential to build an intelligent transportation system. In this work, we address for the first time the problem of detecting traffic events using Twitter as two subtasks: (i) identifying whether a tweet is traffic-related or not as a text classification subtask, and (ii) extracting more fine-grained information (i.e., “what”, “when”, “where”, and the “consequence” of the traffic event) as a slot filling subtask. We also publish two Dutch Traffic Twitter datasets from Belgium and the Brussels capital region. We propose using deep learning based methods that process the two subtasks separately or jointly. Experimental results indicate that the proposed architectures achieve high performance scores (i.e., more than 95%  $F_1$  score) on the constructed datasets for both subtasks, even in a transfer learning scenario. In addition, incorporating tweet-level information in each of the tokens comprising the tweet (for the BERT-based model) can lead to a performance improvement for the joint setting. Our datasets and code are available on GitHub<sup>1</sup>.

*Keywords:* traffic event detection, slot filling, text classification, deep learning

---

\*Corresponding author

*Email addresses:* [xyanga@etrovub.be](mailto:xyanga@etrovub.be) (Xiangyu Yang), [gbekouli@etrovub.be](mailto:gbekouli@etrovub.be) (Giannis Bekoulis), [ndeligia@etrovub.be](mailto:ndeligia@etrovub.be) (Nikos Deligiannis )

<sup>1</sup><https://github.com/Glovesme/TrafficEventDetectionBE>

---

## 1. Introduction

In smart cities, digital technologies (e.g., [Shafiq et al. \(2020\)](#); [Xu et al. \(2022\)](#)) are exploited to improve the citizens' quality of life; this includes building efficient transport networks ([Xu & Hu, 2022](#)). Indeed, in several cities around the world, people experience traffic conditions, such as traffic jams or accidents that can negatively impact their life ([Michael & Blason, 2014](#); [Tom, 2020](#)). By building an intelligent system that can provide useful online traffic information, traffic problems can be mitigated. For example, based on real-time traffic information, travelers can find the best route to their destination, commuters can avoid traffic jams, and traffic management systems can easily monitor traffic flows and inform police once there is an incident, so that traffic can be quickly restored. Thus, it is critical to build a smart system that can detect traffic events in a city.

The traditional way to detect traffic incidents is to use sensors and cameras installed in the city. However, developing such devices everywhere in the city is costly and technically challenging. Nowadays, popular platforms, e.g., Google Maps, can provide real-time traffic congestion information, such as when and where traffic congestion is happening, using GPS data. However, these platforms cannot provide detailed information regarding what happened and what are the consequences of unusual traffic events, which can be of interest to the public and the traffic management center of a city. Moreover, scheduled events that can affect traffic, such as a strike, are normally not included in these platforms. Crowd-sourcing tools that use social network users can potentially address these limitations. Waze, for example, is a successful crowd-sourcing navigation app whose users can report traffic incidents such as accidents, road closures, and police traps. However, it has two limitations as pointed out by [Dabiri & Heaslip \(2019\)](#); [D'Andrea et al. \(2015\)](#): (1) Waze users have to log in to the application and create reports that fall into one of the predefined traffic incident categories. The events that do not belong to any of the categories will not be reported.

(2) Waze is designed for private cars. There is no shared information regarding public transportation. Social media such as Twitter can potentially address the shortcomings of the aforementioned platforms.

Social media allow users to easily access and provide information regarding anything at any time. Twitter, one of the most popular social media platforms, has around 300 million monthly active users. The reason that Twitter is so popular is that its users can share their thoughts through short texts, which are called tweets. With such a large number of users, Twitter generates a large amount of data, in particular, 500 million tweets every day (Sayce, 2020). Since a large amount of content is produced, a lot of recent research has focused on identifying important patterns in people’s daily life. Twitter has been used for various kinds of natural language processing tasks, such as sentiment analysis (Tang et al., 2014; Wang et al., 2020; Naseem et al., 2021) and event detection (Müller et al., 2020; Ali et al., 2021; Sicilia et al., 2021; Saidi et al., 2022). This research includes extracting events from social media posts (e.g., disasters, emergencies (Castillo, 2016)) and extracting sub-events from other events (e.g., a football match (Bekoulis et al., 2019)).

Twitter can serve as a valuable complementary traffic information source that can be used for traffic event detection. Various types of Twitter users can tweet about traffic events they witnessed or experienced; this includes passengers (other than the drivers) of cars or public transportation means and pedestrians. These different kinds of users and accounts of official channels are tweeting during the day about traffic jams, accidents, and so on. Thus, it is useful to exploit the traffic-related information that comes from Twitter streams in order to build smart traffic event detection systems. According to the work of Dabiri & Heaslip (2019), traffic events can be divided into two categories: *recurring* and *non-recurring* events. Recurring events are those that can be easily predicted by looking into historical data, such as daily rush hours. Non-recurring events refer to unpredictable events such as traffic accidents (car crashes), weather-caused issues, and natural disasters.

The limitation of using Twitter as an information stream for traffic event

detection is that often people that are directly involved in the traffic event, especially drivers, are unable to use Twitter to report such an event. However, since Twitter has a large user base, other people who witness the traffic event can also post about it to inform others. Naturally, when nobody tweets about the traffic event, no information will be shared about it on Twitter. This is of course a limitation that other platforms face as well (e.g., Waze). Nevertheless, due to the large number of users on Twitter in cities, one can expect that a large amount of traffic events will be reported. Therefore, exploiting the information coming from the Twitter stream can provide complementary information for building a smart traffic event detection system.

In this work, we focus on detecting traffic-related events in Belgium and the Brussels capital region from the Twitter stream. Specifically, the goal of our traffic event detection system is to use Twitter to identify traffic events and determine information like the time (“when”), the location (“where”), the type (“what”), and the “consequence” of the reported traffic event. To do so, we collect and annotate two traffic-related Dutch Twitter datasets from Belgium and the Brussels capital region. Our datasets contain tweet-level information about whether a tweet is traffic-related or not. In case a tweet is traffic-related, more fine-grained information is included, that is, “when”, “where”, “what”, and the “consequence” of the traffic event. We can approach this problem as a series of two subtasks, namely (i) text classification and (ii) slot filling. The two subtasks can be considered either as two independent subtasks or can be addressed jointly. The goal of subtask (i) is to assign a set of predefined categories (i.e., traffic-related and non-traffic-related) to a textual document (i.e., a tweet in our case). For subtask (ii), the goal is to identify text spans inside the traffic-related tweets that answer the predefined questions defined above. Researchers have identified the benefit of training related subtasks together in a joint setting since the interactions between the subtasks are taken into account (see for instance prior work on multitask learning (Caruana, 1997), entity recognition and relation extraction (Miwa & Bansal, 2016; Bekoulis et al., 2018b), and tree-structured prediction (Bekoulis et al., 2018a)). Although the traffic de-

tection system presented in our research focuses on the Brussels capital region (and in Belgium) and is in Dutch, the principles of the system can be easily extended to other cities. Our trained traffic detection system can also be used as a pre-annotation tool to help annotators to annotate a large-scale traffic Twitter dataset without using a lot of time and effort.

To summarize, the key contributions of our work are as follows:

- We define a new traffic event detection problem. Since there is no benchmarking dataset in this new research direction, we publish two Dutch Traffic Twitter datasets (one with tweets from the Brussels capital region and one with tweets from Belgium) annotated with class- and span-level information for each tweet (as described above). That way, we promote the research of traffic event detection from Twitter streams.
- We propose to solve the problem of detecting traffic-related events from Twitter streams through a series of two subtasks, namely, text classification and slot filling. The text classification task aims to assign a class to a tweet. The slot filling task extracts useful fine-grained information about traffic events from traffic-related tweets. We experiment with several deep learning based architectures (e.g., BERT-based models), and we solve each subtask either independently or in a joint setting. Furthermore, we propose the end-to-end Joint Enhanced BERT-based model that can incorporate the entire information of the tweet into each of its composing tokens to address the two subtasks jointly. This model is able to outperform all other models in the joint setting. Compared to training two independent models for the two subtasks, the joint model is easier to deploy in real-world applications since it only needs to be trained once and can detect the overall tweet category and fine-grained information at once.
- We carry out extensive experiments, and our experimental study indicates the effectiveness of our BERT-based methods over other studied baseline models (e.g., LSTM-based models) for detecting traffic-related events on Twitter.

The rest of this work is organized as follows. Section 2 reviews the related work on traffic event detection from Twitter streams. Section 3 introduces the newly defined task of identifying fine-grained traffic information from traffic-related tweets and describes the annotation process. Section 4 describes the various proposed architectures for solving the task defined in Section 3. Section 5 describes the experimental setup for the proposed methods, introduces the experimental evaluation for the tasks, and showcases the performance of the proposed models. Section 6 concludes our work and discusses future research.

## 2. Related Work

A lot of research in the NLP community has been focused on detecting events from social media (e.g., Naseem et al. (2019); Bekoulis et al. (2019); Dabiri & Heaslip (2019); Müller et al. (2020)). There are two types of event detection, namely, specified and unspecified, as indicated in the work of Saeed et al. (2019). In specified event detection, the event types are determined upfront, and there is a wide range of event types, such as earthquake (Sakaki et al., 2010, 2012), traffic (Ali et al., 2021; Dabiri & Heaslip, 2019), epidemic (Zong et al., 2020; Müller et al., 2020), sports news (Adedoyin-Olowe et al., 2016), etc. In unspecified event detection, there is no prior information about the event types. The techniques used for event detection can be classified into two categories: unsupervised and supervised approaches. Unsupervised approaches usually involve various clustering algorithms (e.g., cluster tweets that contain the top- $k$  bursting keywords from Twitter streams (Li et al., 2012)). Supervised approaches are mainly used for specified event type detection, and the task is mostly framed as a text classification (Dabiri & Heaslip, 2019; Alomari et al., 2019) or as a slot filling problem (Zong et al., 2020). In this paper, we propose new models for solving the newly defined task of identifying fine-grained information from traffic-related tweets. We formulate the task of identifying whether a tweet is traffic-related or not into a text classification problem and the task of identifying fine-grained information as a slot filling problem. In the literature, the text

classification and slot filling tasks are often handled in a joint setting, and the joint task is named as joint intent detection and slot filling (Weld et al., 2021). The intent is the intention of the speaker in an utterance, and the prediction of the intent is mostly treated as a text classification task (Larson et al., 2019). In the following subsections, we present related work on the task of traffic-related event detection using text classification methods (Section 2.1), recent work on the slot filling task (Section 2.2), and relevant work on the task of joint intent detection and slot filling (Section 2.3).

### 2.1. Traffic Event Detection

The traffic event detection from social media problem is mainly approached as a text classification task in the literature. Machine learning based methods have been exploited to tackle the problem, both the traditional (e.g., Support Vector Machine (SVM) (D’Andrea et al., 2015; Salas et al., 2017), Naïve Bayes (NB) (Gu et al., 2016), Decision Tree (DT) (Wongcharoen & Senivongse, 2016)) and the deep learning ones (e.g., Convolutional Neural Networks (CNNs) (Dabiri & Heaslip, 2019; Chen et al., 2019b), Recurrent Neural Networks (RNNs) (Ali et al., 2021; Dabiri & Heaslip, 2019; Chen et al., 2019b)). Apart from using social media to detect traffic-related events, works (Wang et al., 2022b; Huang et al., 2022) also focus on using other information sources like trajectory data from GPS to detect traffic flows or predict travel time by using deep learning based models.

**Traditional Machine Learning:** D’Andrea et al. (2015) proposed a real-time traffic event monitoring system that can fetch tweets from Italian Twitter streams and classify them into the appropriate classes: traffic and non-traffic. They used the Inverse Document Frequency (IDF) technique to represent tweets as features and built an SVM classifier for classification. Instead of using the IDF technique, Salas et al. (2017) utilized n-grams as features for an SVM classifier to detect traffic incidents. Gu et al. (2016) presented a real-time architecture to detect traffic incidents in Twitter streams. In particular, they established a dictionary of keywords and used the combinations of the keywords to infer traffic

incidents. Based on that dictionary, a tweet is represented as a high dimensional binary vector. Those binary vectors are then fed into an NB classifier to identify whether the corresponding tweets are traffic incidents or not. [Wongcharoen & Senivongse \(2016\)](#) proposed a model to detect the congestion severity levels from Twitter streams. In that work, they considered four attributes (i.e., day of the week, hours of the day, minutes of the hour, and tweets density) to construct a C4.5 DT for congestion severity level prediction. [Wang et al. \(2017\)](#) introduced a Latent Dirichlet Allocation (LDA) model called tweet-LDA to extract traffic alerts and warning topics from Twitter in real-time.

**Deep Learning:** [Zhang et al. \(2018\)](#) developed a traffic accident detection system that uses traffic-related tokens (e.g., accident, car, and crash) as features to train a Deep Belief Network (DBN). The selection of tokens is based on a coefficient that measures the association between the labels (accident or not) and the tokens. [Chen et al. \(2019b\)](#) built a binary classification system to detect traffic-related information from Weibo (a Chinese social media platform). They applied continuous bag-of-words (CBOW) to learn word embeddings to represent words in microblogs. Then, they used the learned word embeddings as input to CNNs, Long Short-Term Memory (LSTM) networks, and their combined LSTM-CNN architecture to detect traffic-related microblogs. [Dabiri & Heaslip \(2019\)](#) proposed to address the traffic event detection problem on Twitter as a text classification problem using deep learning architectures. In particular, in their work, they first collected and labeled a traffic-related Twitter dataset from the USA, which contains three classes: non-traffic (events that are not related to traffic), traffic incident (non-recurring events such as car crashes, traffic signal problem, and disabled vehicles), and traffic information & condition (recurring events such as traffic congestion, daily rush hours, and traffic delays). After that, they used pre-trained word embedding models (word2vec ([Mikolov et al., 2013](#)) and FastText ([Bojanowski et al., 2017](#))) to get tweet representations. CNNs and RNNs were deployed on top of the word embeddings layer to extract traffic-related tweets. [Ali et al. \(2021\)](#) presented an architecture to detect traffic accidents and analyze traffic conditions directly from social networking data.



They first collected traffic information from Twitter and Facebook by using a query-based search engine. Then, Ontologies and Latent Dirichlet Allocation (OLDA) were used to automatically label each sentence with either the traffic or the non-traffic class labels. Finally, they used FastText with bidirectional LSTMs (BiLSTMs) to detect domain-specific event types (e.g., traffic accidents and traffic jams) and predict user sentiments (i.e., positive, neutral, or negative) towards those traffic events. [Chang et al. \(2022\)](#) proposed using a CNN + LSTM model first to detect traffic-related Weibos with location information and then using a keyword matching method to identify accident-related Weibos.

Previous work on traffic event detection using social media mainly focuses on classifying tweets into two classes, traffic-related and non-related ([D’Andrea et al., 2015](#); [Salas et al., 2017](#); [Gu et al., 2016](#); [Zhang et al., 2018](#); [Chen et al., 2019b](#); [Dabiri & Heaslip, 2019](#); [Ali et al., 2021](#); [Chang et al., 2022](#)). However, extracting precise information regarding a particular traffic event from tweets is also very crucial. In this paper, we aim to identify not only the traffic-related tweets but also the fine-grained information which can provide valuable and accurate information in practical applications. Thus, we address the traffic event detection problem by (i) determining whether a tweet is traffic-related or not, and (ii) detecting fine-grained information from tweets. The fine-grained information (e.g., “where” or “when” an event has happened) could help us decide the nature of the event (e.g., whether it is traffic-related or not). Moreover, in the case that the event is traffic-related, it could also help us to decide whether we should identify text spans about the fine-grained information for the slot filling task. We conduct extensive experiments, and we study the two subtasks (i.e., text classification and slot filling) either separately or in a joint setting to identify whether there is a benefit by explicitly sharing the layers of the neural network between the subtasks. The way that we formulate the traffic event detection problem has not been studied before in the traffic event detection domain, and we hope that this could boost future research on traffic event detection using social media.

## 2.2. Slot Filling

In Natural Language Understanding (NLU), slot filling is a task whose goal is to identify spans of text (i.e., the start and the end position) that belong to predefined classes directly from raw text. The slot filling task is mainly used in the context of dialog systems where the aim is to retrieve the required information (i.e., slots) out of the textual description of the dialog.

Slot filling is usually formulated as a sequence labeling task, and neural network based models have mainly been proposed for solving it. In particular, [Vu \(2016\)](#) proposed a bidirectional sequential CNN model that predicts the label for each slot by taking into account the context (i.e., previous and future) words with respect to the current word and the current word itself. [Kurata et al. \(2016\)](#) developed the encoder-labeler LSTM that first uses the encoder LSTM to encode the entire input sequence into a fixed length vector. This vector is then passed as the initial state to the labeler LSTM to perform the sequence labeling task. This model is able to predict slot labels while taking into account the whole information of the input sequence. [Zhu & Yu \(2017\)](#) introduced the BiLSTM-LSTM, an encoder-decoder model that encodes the input sequence using a BiLSTM and decodes the encoded information using a unidirectional LSTM. They also designed a so-called focus mechanism that is able to address the alignment limitation of attention mechanisms (i.e., cannot operate with a limited amount of data) for sequence labeling. [Zhao & Feng \(2018\)](#) presented a sequence-to-sequence (Seq2Seq) model along with a pointer network to improve the slot filling performance. To predict slot values, the model learns to either copy a word (which may be out-of-vocabulary (OOV)) through a pointer network, or generate a word within the vocabulary through an attentional Seq2Seq model. [Korpusik et al. \(2019\)](#) compared a set of neural networks (CNN, RNN, BiLSTM, and Bidirectional Encoder Representations from Transformers (BERT)) on slot filling tasks. Their results indicate that the BERT-based models outperform the other studied architectures. Recently, [Zong et al. \(2020\)](#) published the COVID-19 Twitter Event Corpus, which has 7,500 annotated tweets and includes five event types (TESTED POSITIVE, TESTED

NEGATIVE, CAN NOT TEST, DEATH, and CURE&PREVENTION). For each event type, a set of slot types is predefined for slot filling tasks (e.g., for the TESTED POSITIVE event, the goal is to identify slot types like “who” (i.e., who was tested positive), “age” (i.e., the age of the person tested positive), and “gender” (i.e., the gender of the person tested positive)). They proposed a BERT-based model that treats each slot filling task in each event type as a binary classification problem. Compared to their model, in [Yang et al. \(2020\)](#), we proposed a multilabel BERT-based model that jointly trains all the slot types for a single event and achieves improved slot filling performance. In this paper, we modify existing slot filling techniques, and we apply them in the context of traffic event detection from Twitter streams.

### *2.3. Joint Intent Detection and Slot Filling*

The tasks of intent detection and slot filling have also been studied in a joint setting. Given an utterance, intent detection aims to identify the intention of the user (e.g., book a restaurant), and the slot filling task focuses on extracting text spans that are relevant to that intention (e.g., place of the restaurant, timeslot). By training the two tasks in a joint setting, the model is able to learn the inherent relationships between the two tasks of intention detection and slot filling. This approach can further improve the overall performance of the joint task and the performance of each independent subtask ([Caruana, 1997](#); [Weld et al., 2021](#)). The two types of methods that are mainly exploited when solving the two tasks simultaneously are the RNN-based and the attention-based approaches.

**RNN-based:** [Zhou et al. \(2016\)](#) proposed a hierarchical LSTM model which has two LSTM layers. The final hidden state of the bottom LSTM layer is used for intent detection, while that of the top LSTM layer with a softmax classifier is used to label the tokens of the input sequence. [Hakkani-Tür et al. \(2016\)](#) developed a single BiLSTM model that concatenates the hidden states of the forward and the backward layers of an input token and passes these concatenated features to a softmax classifier to predict the slot label for that

token. A special tag is added at the end of the input sequence to capture the context of the whole sequence and detect the class of the intent. [Zhang & Wang \(2016\)](#) proposed a bidirectional gated recurrent unit (GRU) architecture that operates in a similar way to the work of [Hakkani-Tür et al. \(2016\)](#) for labeling the slots. However, the intent detection is done in a different way since [Zhang & Wang \(2016\)](#) used a max-pooling layer for all the hidden states, and then they applied a softmax function on top of the max-pooling layer. [Firdaus et al. \(2018\)](#) introduced an ensemble model that feeds the outputs of a BiLSTM and a BiGRU separately into two multi-layer perceptrons (MLP). The outputs of the MLPs are concatenated, and a softmax classifier is used for predicting the intent and the slots simultaneously.

**Attention-based:** Attention mechanisms have also been exploited for jointly learning the relationships between the two studied subtasks. In particular, [Liu & Lane \(2016\)](#) proposed an attention-based bidirectional RNN (BRNN) model that takes the weighted sum of the concatenation of the forward and the backward hidden states as an input to predict the intent and the slots. [Li et al. \(2018\)](#) proposed the use of a BiLSTM model with a self-attention mechanism ([Vaswani et al., 2017](#)) and a gate mechanism to solve the joint task. One self-attention mechanism is used at the words and the characters level of the input sequence to obtain a semantic representation of the input. Then, these representations are fed into a BiLSTM, and the final hidden state is then used for intent detection. Another self-attention layer is applied between the intermediate states of the BiLSTM, and the intermediate states are combined with the predicted intent for labeling the slots. [Goo et al. \(2018\)](#) introduced an attention-based slot-gated BiLSTM model. In that model, the embeddings of the input sentence are fed into a BiLSTM, and then a weighted sum of the BiLSTM intermediate states (i.e., the slot context vector) is used for predicting the slots. The final state of the BiLSTM (i.e., the intent context vector) is used for predicting the intent. A slot gate is added to combine the slot context vector with the intent context vector, and the combined vector is then fed into a softmax to predict the current slot label. [Chen et al. \(2019a\)](#) proposed a joint BERT model for

solving the joint task of intent detection and slot filling. The model predicts the intent based on the final hidden state of the [CLS] token, and the final hidden state of each token is used for predicting the slot labels. The two tasks are trained jointly by using a joint loss (i.e., one for each subtask). Our joint model is conceptually related to that of [Chen et al. \(2019a\)](#). Unlike the work of [Chen et al. \(2019a\)](#), which does not exploit the use of the intent information, we incorporate intent information into each token to improve the performance of the slot filling subtask. This also improves the overall performance of the joint task (i.e., intent detection and slot filling).

### 3. Problem Formulation and Datasets

In this section, we define the traffic event detection problem from Twitter streams and explain that this problem can be addressed by the two subtasks of text classification and slot filling. We then present the way that the two datasets (i.e., the one for Belgium and the other for the Brussels capital region) have been constructed (i.e., data collection and annotation process).

#### 3.1. Subtasks

By formulating the problem into a series of two subtasks (text classification and slot filling), we give an answer to two questions: (i) whether the given tweet is traffic-related (or not), and (ii) whether more precise/fine-grained information can be identified regarding a traffic-related event (from the corresponding tweet). The two subtasks for a particular tweet are illustrated in [Figure 1](#).

**Text Classification:** The goal of this subtask is to distinguish traffic-related tweets from non-traffic-related tweets. Since there are two classes, this subtask, in its essence, is a binary classification task. That is, given a tweet  $t$ , the classification model  $f(t) \rightarrow \{0, 1\}$  predicts whether a tweet is traffic-related or not, where 1 means *traffic-related*, and 0 means *non-traffic-related*.

**Slot Filling:** This subtask aims at extracting fine-grained events from traffic-related tweets. We are interested in four types of fine-grained events, specifically:

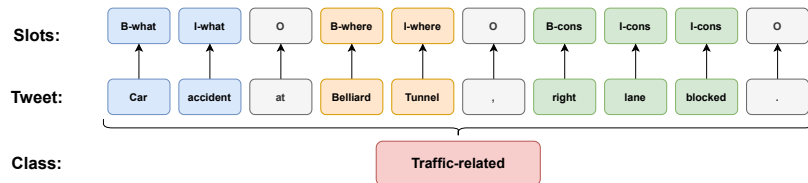


Figure 1: An example tweet where the two subtasks of text classification and slot filling are illustrated. The output for the text classification problem is the class label of the tweet (i.e., traffic-related). The output for the slot filling problem is for each token the slot filling label (e.g., “what”, “where”) encoded using the BIO encoding scheme.

we are interested in identifying “when” (i.e., the exact time that the traffic-related event has happened (as described in the corresponding tweet)), “where” (i.e., the location that the traffic-related event has happened, “what” (i.e., the type of the incident that has happened, e.g., accident, traffic jam) and the “consequence” of the aforementioned event (e.g., lane blocked). We frame the slot filling problem into a sequence labeling task. Given the input sequence of tokens  $X = (x_1, x_2, \dots, x_n)$ , the goal is to map the input sequence  $X$  into a tagged (labeled) sequence  $Y = (y_1, y_2, \dots, y_n)$  of the same length, where  $n$  is the number of tokens within the sequence. We employ the BIO (beginning-inside-outside) scheme for tagging the sequences, and more details about that can be found in Figure 1. The BIO scheme is a tagging format for labeling tokens according to the positions of the tokens within a chunk. The O tag indicates that the corresponding token is outside of the chunk. The B- prefix before the tag indicates that the corresponding token is at the beginning of the chunk. The I- prefix before the tag indicates that the corresponding token is inside a chunk. Thus, there is the constraint that the tag with the I- prefix should always come after the tag with the B- prefix or the I- prefix (of the same type, e.g., where).

### 3.2. Datasets

We have constructed two annotated Dutch datasets from the Twitter stream. The first one is from Belgium, the “Belgian Traffic Twitter Dataset” (BE

	BE dataset	BRU dataset
<b>Class</b>	<b># of tweets</b>	
traffic-related	5,386	3,213
non-traffic-related	5,237	3,313
Total	10,623	6,526
<b>Slot</b>	<b># of slots</b>	
where	5,305	3,144
what	5,121	2,963
when	5,111	3,030
consequence	3,939	2,551

Table 1: Statistics of the BE dataset (i.e., the Belgian Traffic Twitter Dataset) and the BRU dataset (i.e., the Brussels Traffic Twitter Dataset).

dataset), and the other is from the Brussels capital region, the “Brussels Traffic Twitter Dataset” (BRU dataset). In fact, the BRU dataset is part of the BE dataset, and we extract the BRU dataset out of the BE dataset to study the traffic events in a particular part of the country, the Brussels capital region. The tweets within the two datasets range from 2015 to 2019. The two datasets also contain the annotations that consist of two parts: the class of each tweet (whether a tweet is traffic-related or not) and the slots of the fine-grained information (i.e., “when”, “where”, “what”, and “consequence”). In Figure 2, an example of an annotated tweet is illustrated.

**Data Collection:** Dutch and French are the two most common languages used in Belgium. Since Brussels is an international city, English is also commonly used. Thus, we decided to track tweets in Dutch, French, and English from Belgium. The first step in collecting a large Twitter dataset from Belgium is to use the Twitter API to harvest tweets from Belgium in the specified time period (i.e., from 2015 to 2019). Similar to the work of [Dabiri & Heaslip \(2019\)](#), where they establish a list of traffic-related keywords to filter out traffic-related tweets, we establish our own traffic-related keywords in Dutch, French, and English. Detailed information can be found on our GitHub codebase<sup>2</sup>. We then use

<sup>2</sup><https://github.com/Glovesme/TrafficEventDetectionBE>

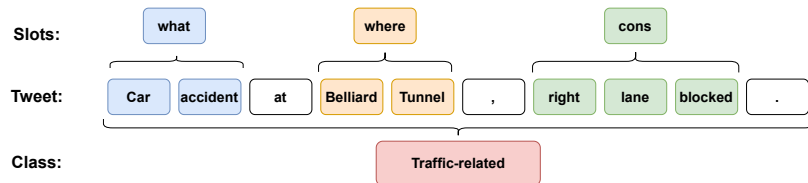


Figure 2: An example of an annotated tweet from our dataset. This is not a fictitious tweet but rather a tweet that we translate from Dutch to English. An annotated tweet contains three parts: the tweet text, the class of the tweet, and different tweet slots.

these keywords on the harvested tweets, and we obtain potential traffic-related tweets in the aforementioned three languages. To investigate the possible distribution of these potential traffic-related tweets regarding the three languages, we translate all the non-English tweets into English (using the translators API for Python) and then build a CNN classifier based on the US traffic dataset from Dabiri & Heaslip (2019). After manually investigating the results from the CNN classifier, we identify that the majority of the real traffic-related tweets out of all the potential traffic-related tweets come from Dutch speakers. Thus, this is why we focus on building a high-quality Dutch annotated Twitter dataset for Belgium.

**Data Annotation:** There are two tasks in the annotation process. The first task is identifying whether a given tweet is traffic-related. The traffic-related tweets can report non-recurring events or recurring events. The recurring events are the events that are predictable, such as traffic congestion, traffic delay, and daily rush hours. The non-recurring events include unpredictable events such as car accidents, traffic signal problems, and disabled vehicles. The second task is to find relevant information (e.g., “when”, “where”) from tweets identified as traffic-related. In order to easily annotate our dataset, we used the TagTog platform<sup>3</sup>, which is an annotation platform that can improve the annotation experience. Apart from using the annotation platform, we also hired a native

<sup>3</sup><https://www.tagtog.net/>



Dutch speaker to help us with the annotation. At the end of the annotation process, we also manually checked the annotation results. The BE dataset contains 10,623 tweets, and the BRU dataset (a part of the BE dataset from the Brussels capital region) contains 6,526 annotated tweets as also reported in Table 1. The problem that we are going to address in this paper is not a straightforward problem to solve (e.g., with a predefined set of keywords). This is because when we pre-filter a large fraction of the tweets with a predefined keyword set, these tweets belong to the traffic-related class. However, when we annotate them, these tweets belong to the non-traffic-related class. Our datasets can help annotators create a large-scale traffic Twitter dataset for the two subtasks. Once a deep model is trained on our datasets, annotators can then use it to filter out useful information about tweets (i.e., the tweet class and the slots) to save time and effort for the annotation process.

#### 4. Proposed Architectures

In this section, we describe the proposed approaches for solving the two subtasks (i.e., text classification and slot filling) either independently or in a joint setting. In Section 4.1, we introduce the three main components (i.e., CNN, LSTM, and BERT) that are mainly exploited by the independent (i.e., the two subtasks are considered separately in Section 4.2) and the joint (i.e., the two subtasks are considered in a joint setting in Section 4.3) models for solving the traffic event detection problem. The joint models are used based on the observation that there are connections between a traffic-related tweet and the fine-grained information (e.g., “when”, “where”, “what”, and “consequence”). Apart from that, we also propose enhanced joint models that can incorporate the whole tweet information into each token in the tweet to boost the overall performance.

##### 4.1. Core Components

###### 4.1.1. Neural-based Methods

**Convolutional Neural Networks (CNNs):** CNNs have been mainly exploited in computer vision tasks (e.g., image classification (Krizhevsky et al.,

2012; Girshick, 2015; He et al., 2020), semantic segmentation (Wang et al., 2018), image super-resolution (Zhang et al., 2020), etc.). However, CNNs have also been extensively used in NLP for tasks, such as text classification (Kim, 2014), sequence labeling (Xu et al., 2018), etc., due to their ability to extract  $n$ -gram features. This kind of model consists of a number of convolutional filters (of various sizes) that are applied on top of the embedding layer. In this work, CNNs are mainly used as a core component for the text classification subtask.

**Long Short-Term Memory (LSTMs):** LSTMs, a variant of Recurrent Neural Networks (RNNs) (Hochreiter & Schmidhuber, 1997), can handle data of sequential nature (i.e., text) and showcase state-of-the-art performance in a number of NLP tasks (e.g., text classification (Zhou et al., 2015), sequence labeling (Lu et al., 2019), fact checking (Rashkin et al., 2017; Bekoulis et al., 2020)). RNNs suffer from the vanishing gradient problem, which harms convergence when dealing with long input sequences. LSTMs use modifications, such as the cell state (i.e., the memory of LSTM), to overcome the vanishing gradient problem. The standard LSTM processes the input sequence from left to right, and for each input token, LSTMs produce a hidden state as output that takes the previous input tokens into account. LSTMs can also be applied from right to left; thus, bidirectional LSTMs (BiLSTMs) can obtain bidirectional information for each input token.

#### 4.1.2. BERT

Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) is a Transformer-based language representation model (Vaswani et al., 2017), where multiple Transformer encoders are stacked one on top of the other and are pre-trained on large corpora. For each input sequence, a special classification token [CLS] is added at the beginning, and a special token [SEP] is added at the end of each sentence. The outputs of the BERT model are high-level deep bidirectional contextual representations of the input tokens.

### BERT-based Pre-trained Models for the Dutch Language

Since both constructed datasets are in Dutch, we use four BERT-based models that are pre-trained on Dutch data. The four pre-trained Dutch models are the following: BERTje (de Vries et al., 2019), RobBERT (Delobelle et al., 2020), mBERT (Devlin et al., 2019), and XLM-RoBERTa (Conneau et al., 2020).

**BERTje** is a Dutch BERT model that is pre-trained on a large and diverse Dutch dataset of 2.4 billion tokens from Dutch Books, TwNC (Ordelman et al., 2007), SoNaR-500 (Oostdijk et al., 2013), Web news and Wikipedia.

**RobBERT** is a Dutch language model based on the Robustly Optimized BERT method (RoBERTa (Liu et al., 2019)) and is pre-trained on the Dutch part of the OSCAR corpus with 6.6 billion words.

**mBERT** is a multilingual BERT model, which also includes the Dutch language. The Dutch part of the model is only pre-trained on Dutch Wikipedia text.

**XLM-RoBERTa** is a multilingual model trained in 100 different languages, which includes Dutch, and is based on RoBERTa.

## 4.2. Independent Models

In this subsection, we describe the models that address the two subtasks independently.

### 4.2.1. Text Classification

**CNN:** The CNN model (Figure 3) consists of four parts: the word embeddings layer, a convolutional layer, a max pooling layer, and a fully connected layer. Specifically, at the word embeddings layer, the input tokens are mapped to word embeddings (i.e., word vectors). Then the convolutional layer extracts  $n$ -gram features from the input, and those features are further processed by a max pooling layer. Finally, the processed features are passed to the fully connected layer to make class predictions for the input sequence.

**LSTM:** The LSTM model shown in Figure 4 consists of three parts: the word embeddings layer, a BiLSTM layer, and a softmax layer. The input sequence is converted into pre-trained word embeddings. Then, the word embeddings are

processed by the BiLSTM layer, and the final hidden states of the forward and the backward LSTMs are concatenated. The concatenated hidden states are then passed to a softmax layer to predict the class of the input sequence.

**CNN + LSTM (Chang et al., 2022):** This model combines Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. It consists of four parts: the word embeddings layer, a CNN layer, an LSTM layer, and a softmax layer. The CNN extracts features from the word embeddings of the input text. Then these features are passed to the LSTM part of the model to get the input text representation, which is finally fed into the softmax layer to obtain the probability distribution over the classes.

**BERT-based Model:** The hidden state of the [CLS] tag encodes the information of the whole input sequence (Devlin et al., 2019). Thus, we use the [CLS] representation to represent the whole sentence for the text classification task. The architecture of the BERT-based model is shown in Figure 5. To predict the class of the input sequence, a softmax layer is applied on top of the hidden state of the [CLS] token:

$$y^c = \text{softmax}(W^c h_{[cls]} + b^c), \quad (1)$$

where  $y^c$  is the prediction of the input sequence,  $W^c$  is the weight matrix,  $h_{[cls]}$  is the hidden state of the special [CLS] token, and  $b^c$  is the bias vector.

#### 4.2.2. Slot Filling

We treat the slot filling task as a sequence labeling problem. For each token within a tweet, we assign the slot label with the highest probability.

**LSTM:** This model has a similar structure to the LSTM model presented for the text classification task in Section 4.2.1. However, in the slot filling case, the concatenated BiLSTM hidden states for each input token are used to predict the tag for each token, as shown in Figure 6.

**LSTM + CRF:** Instead of using a softmax layer on top of the LSTM model that independently predicts the tag for each input token, a conditional random field (CRF) layer (Lample et al., 2016) is employed to capture the relationships

between neighboring tokens. The architecture is shown in Figure 6.

**BERT-based Model:** Figure 7 shows the structure of the BERT-based model for slot filling. The hidden states of the input tokens are used for labeling the tokens. A softmax classifier is added over the hidden state of each corresponding input token to predict the corresponding tag. The BERT-based models use the WordPiece tokenizer (Wu et al., 2016) and can partition one word into several sub-tokens according to the vocabulary of the tokenizer. However, the model itself should output one prediction per word/token, and each token might be split into several sub-tokens. To handle this issue, we only make predictions for the first sub-token of each token, in the case that a token has been split into multiple sub-tokens (by the WordPiece tokenizer), or for the token itself, in the case that the whole token has been retained. The equation for the slot filling task is:

$$y_i^s = \text{softmax}(W^s h_i + b^s), \quad i \in 1 \dots N, \quad (2)$$

where  $y_i^s$  is the tag prediction of the token  $x_i$ ,  $W^s$  is the weight matrix,  $h_i$  is the hidden state of the first sub-token of  $x_i$ , and  $b^s$  is the bias vector.

### 4.3. Joint Models

In this subsection, we describe the joint models for the two subtasks:

**Slot-Gated (Goo et al., 2018):** Built on top of an attention-based BiLSTM architecture, the Slot-Gated model has a slot-gated mechanism, which is designed to learn the relationship between the intent and the slot context vectors to improve the performance of the slot filling task.

**SF-ID Network (E et al., 2019):** Based also on BiLSTMs, the SF-ID network can directly establish connections between the intent detection and the slot filling subtasks. In addition, an iteration mechanism is designed to enhance the interrelated connections between the intent and the slots.

**Capsule-NLU (Zhang et al., 2019):** This model uses a dynamic routing-by-agreement schema to tackle the intent detection and the slot filling subtasks. As a result, the model is able to preserve the hierarchical relationship between the two subtasks.

**Joint BERT** (Chen et al., 2019a): The model, depicted in Figure 8, uses the hidden state of the [CLS] token to perform the intent detection subtask, and the slot filling subtask on the hidden states of the other non-special tokens (i.e., the tokens of the sentence). The two tasks are jointly trained by adopting a joint loss function.

**Joint Enhanced BERT-based Model:** While the previous models have already been proposed in the literature, in what follows, we describe our proposed end-to-end model, the Joint Enhanced BERT-based model, for the joint task. Compared to the independent models for the two subtasks, this end-to-end model is able to address the two subtasks simultaneously and can be deployed very easily by training only once instead of twice. Figure 9 shows the structure of our proposed model. This model takes into account the two following facts: (i) the class of a tweet (i.e., traffic-related or not) is relevant to the slot filling problem (i.e., identify more fine-grained information, e.g., “when”, “where”), and vice versa, and (ii) the hidden state of the [CLS] contains the information of the entire input sequence. Compared to the joint BERT model (Chen et al., 2019a), which only trains the two tasks together using a joint loss without modeling the relationships between them, we incorporate the information of the entire input sequence of the tweet text into each token to improve the performance of the model. For each non-special token, we concatenate its hidden state with the hidden state of the [CLS] tag. The concatenation is used to predict the label for the token, similar to the way that we were using the hidden state of the token in the standard Joint BERT model (described above):

$$h'_i = [h_i, h_{[cls]}], \quad (3)$$

$$y_i^{s'} = \text{softmax}(W^{s'} h'_i + b^{s'}), i \in 1 \dots N, \quad (4)$$

where  $h'_i$  is the concatenation of the hidden states of the  $i_{th}$  token and the [CLS] token,  $y_i^{s'}$  is the tag prediction of the token  $x_i$ ,  $W^{s'}$  is the weight matrix,  $h'_i$  is the hidden state of the first sub-token of  $x_i$ , and  $b^{s'}$  is the bias vector. The goal of

this joint enhanced model is to maximize the conditional probability:

$$p(y^c, y^{s'} | X) = p(y^c | X) \prod_{i=1}^N p(y_i^{s'} | X), \quad (5)$$

where  $y^c$  is the prediction of the input sequence,  $y_i^{s'}$  is the tag prediction of the token  $x_i$ , and  $X$  is the input sequence. We use Equation 1 to predict the class of a tweet.

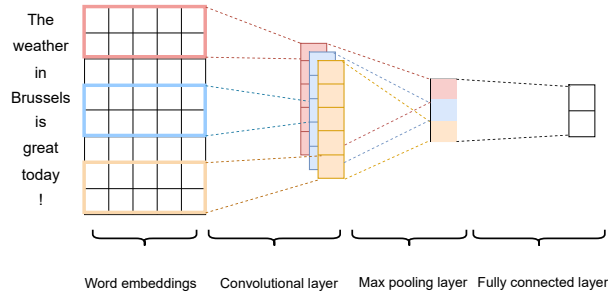


Figure 3: Schema of a CNN model for text classification. The model has four parts: input word embeddings, a convolutional layer, a max pooling layer, and a fully connected layer.

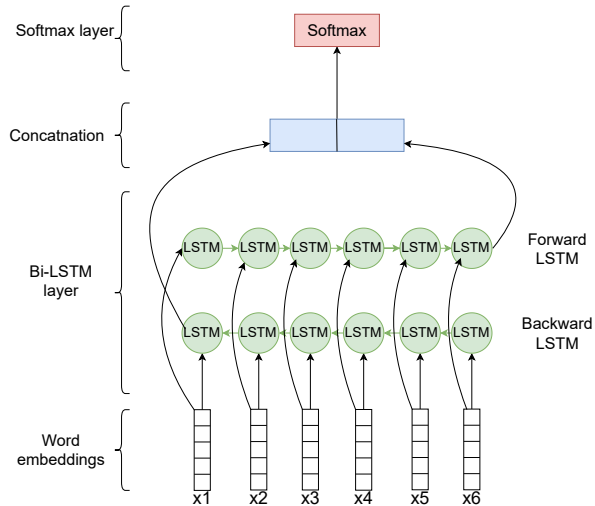


Figure 4: Schema of an LSTM model for text classification. This model consists of three parts: word embeddings, a BiLSTM layer, and a softmax layer.

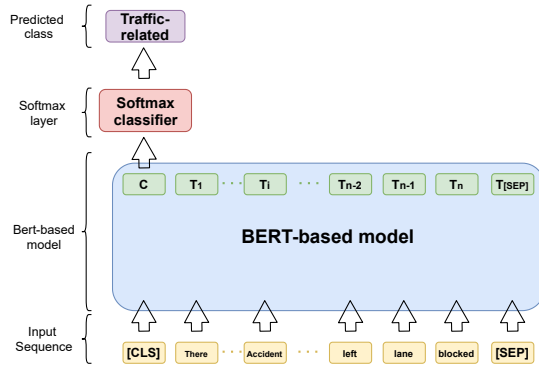


Figure 5: BERT-based model for text classification. The model consists of three parts, the input sequence (yellow rectangles), the BERT-based model (the blue rectangle), and the softmax layer (the red rectangle). The green rectangles represent the hidden states of the corresponding input tokens. The class of the input sequence is predicted by applying a softmax layer on top of the hidden state of the  $[CLS]$  token. The predicted class is shown in the purple rectangle.

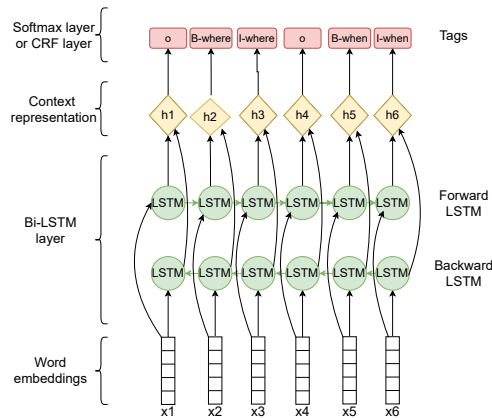


Figure 6: LSTM model for slot filling. This model has three parts: a word embedding layer, a BiLSTM layer, and a softmax or CRF layer. The input sequence is transformed into word embeddings, the embeddings are then passed to a BiLSTM layer, the BiLSTM layer concatenates the hidden states of the forward and the backward LSTMs for each token, and the concatenated features are used in a softmax or CRF layer to predict the tag for each input token.



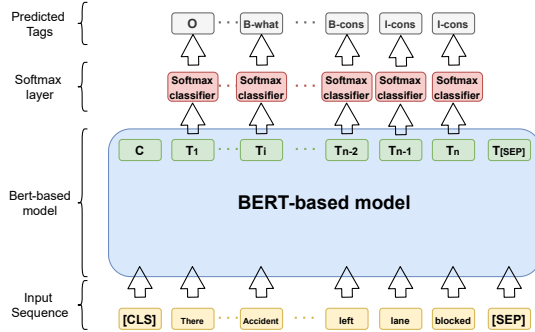


Figure 7: BERT-based model for slot filling. The model consists of the input sequence (yellow rectangles), the BERT-based model (the blue rectangle), and a softmax layer (red rectangles). Each green rectangle contains the hidden state of the corresponding input token. Each grey rectangle is the predicted tag for the corresponding token.

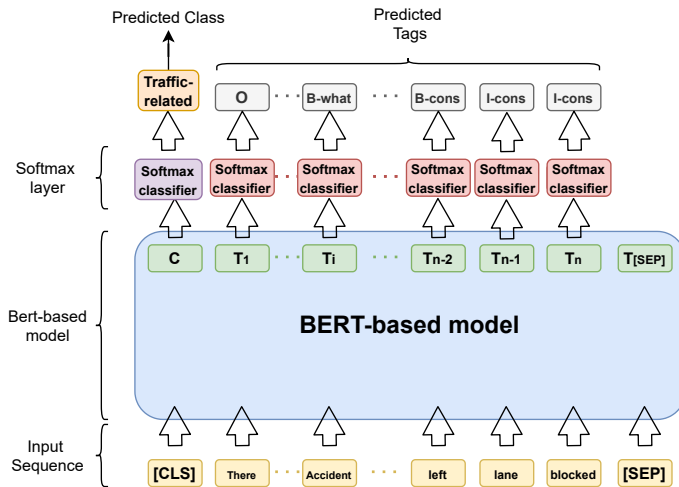


Figure 8: Joint BERT. The model combines the architectures of the independent BERT-based models for text classification and slot filling. The orange rectangle is the predicted class for the input sequence, and each grey rectangle is the predicted tag for the corresponding input token. For this model, the text classification and slot filling tasks are jointly trained.

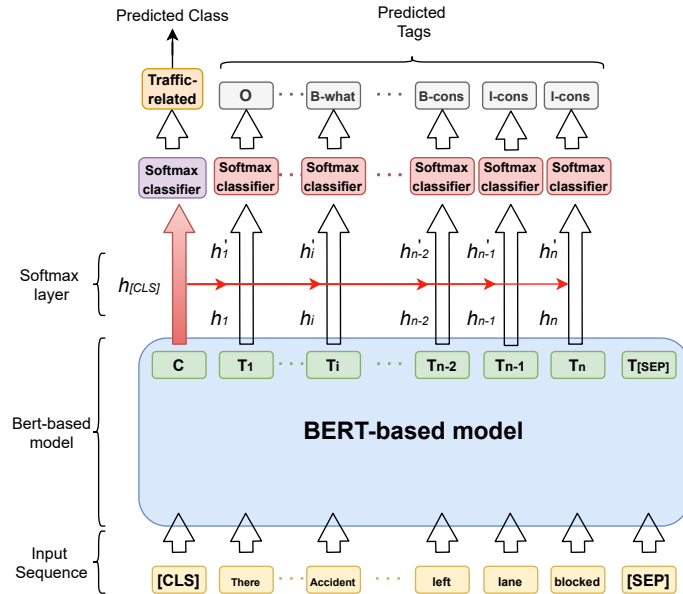


Figure 9: Joint Enhanced BERT-based model for joint text classification and slot filling. This model is similar to the joint BERT architecture but the  $[CLS]$  hidden state is concatenated to each of the hidden states of the tokens of the sentence. The red arrows connect the contextual information of the sentence with each non-special token. Note that the text classification and slot filling tasks are jointly trained.

## 5. Experiments and Results

In this section, we describe (i) the evaluation metrics for all the experimented methods, (ii) the baseline models, (iii) the experimental settings, and (iv) the experimental environment for the various models. Finally, we evaluate the performance of the proposed architectures for solving the traffic event detection problem, and we discuss the results.

### 5.1. Evaluation Metrics

We adopt three evaluation metrics for evaluating the different models on the constructed datasets. We use the  $F_1$  score for the two subtasks of text classification and slot filling, denoted as  $F_{1c}$  and  $F_{1s}$ , respectively. For the joint text classification and slot filling, the sentence-level semantic frame accuracy (SenAcc) score is calculated, indicating the proportion of sentences (out of all sentences) that have been correctly classified. In order for a sentence to be correct, both the class and the slots of the sentence should be identified correctly.

### 5.2. Baselines

We select the non-BERT-based models as the baseline models for the two independent subtasks and the joint task.

#### 5.2.1. Text Classification

For the text classification task, **CNN**, **LSTM**, and **CNN + LSTM** are chosen as the baseline models. The **CNN** and **LSTM** models are exploited in the state-of-the-art traffic event detection work of [Dabiri & Heaslip \(2019\)](#) to identify traffic-related tweets. The **CNN + LSTM** model is used in the state-of-the-art method of [Chang et al. \(2022\)](#) to detect traffic-related Weibos. The details of these models are described in [4.2.1](#).

#### 5.2.2. Slot Filling

In terms of the slot filling task for extracting fine-grained traffic-related information (i.e., “when”, “where”, “what”, and the “consequence” of the traffic event)

from tweets, **LSTM** and **LSTM + CRF**, described in 4.2.2, are selected as the baseline models.

### 5.2.3. Joint Text Classification and Slot Filling

As for the joint text classification and slot filling task, we use LSTM-based models (**Slot-Gated** (Goo et al., 2018), **SF-ID Network** (E et al., 2019), **Capsule-NLU** (Zhang et al., 2019)), and **Joint BERT** (Chen et al., 2019a) as the baseline models. The details of them are included in 4.3.

### 5.3. Experimental Settings

We randomly split the two datasets (i.e., BRU and BE); specifically, we keep 60% for training, 20% for validation, and 20% for the test sets for each of the datasets. Since the URLs do not provide useful information for traffic events, we remove all the URLs from the tweets, similar to the work of Dabiri & Heaslip (2019).

For the non-BERT-based models such as CNN, LSTM for text classification, CNN + LSTM, LSTM for slot filling, and LSTM + CRF, we use the 160-dimensional Dutch word embeddings called Roularta (Tulkens et al., 2016) to convert tweet text into word embeddings, and we set the batch size of these models to 64 or 128. For the CNN model, the filters in the convolutional layer have a size of  $n \times 160$ , where  $n = [3, 4, 5]$ . The number of out channels of the convolutional layer is 100. We apply dropout with a rate of 0.5 on the outputs of the max pooling layer. All the LSTM models for text classification and slot filling have one BiLSTM layer. In the LSTM model for text classification, the input and hidden dimensions of the BiLSTM layer are set to 100 and 256, respectively. Dropout is applied on the output of the BiLSTM layer with a rate of 0.5. For the CNN + LSTM model, the CNN part has the same convolutional layer setting as the CNN model, and the input and hidden dimensions of the BiLSTM layer in the LSTM part are set to 100 and 128, respectively. The dropout rate is 0.5 for the output of the BiLSTM layer of the LSTM part. For these three models (i.e., CNN, LSTM, and CNN + LSTM for text classification),

the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.001 is applied to update the model parameters. For the LSTM and the LSTM + CRF models for slot filling, the input and hidden dimensions of the BiLSTM layer are 160 and 100, respectively. The stochastic gradient descent algorithm with a learning rate of 0.015 is utilized for updating the model parameters. The dropout rate is set to 0.5 for the output of the BiLSTM layer. For the Slot-Gated, SF-ID Network, and Capsule-NLU models, we apply the Roularta embeddings (Tulkens et al., 2016), and use their default settings as described in their papers (Goo et al., 2018; E et al., 2019; Zhang et al., 2019).

For the BERT-based models, the batch size is set to 32 or 64. The dropout rate is 0.1. The number of epochs is selected from the values 10, 15, 20, 25, 30, and 40. The BERT-based models use the pretrained Dutch BERT-based models (i.e., BERTje (de Vries et al., 2019), RobBERT (Delobelle et al., 2020), mBERT (Devlin et al., 2019), and XLM-RoBERTa (Conneau et al., 2020)). Adam (Kingma & Ba, 2015) is employed to optimize the models’ parameters with an initial learning rate of 1e-4 and 5e-5 for the joint and the independent models, respectively.

#### 5.4. *Experimental Environment*

Regarding the software environment, we implemented our work using PyTorch 1.7.1, Transformers 4.2.2, and Python 3.8.5 on Ubuntu 20.04. The server used to perform all the experiments has the following specifications: AMD Ryzen 9 3950X 16-core processor, 64GB of RAM, and GeForce RTX 3090. The training times for different BERT-based models are reported in Table 2.

#### 5.5. *Results*

##### 5.5.1. *Independent Models*

Table 3 reports the performance of the different models for the tasks of text classification and slot filling on the BRU and the BE datasets, where the two tasks are considered independently. We also report transfer learning experiments (see the BRU→BE column in Table 3 and the transfer learning part of

Model	TC		SF		Joint Task	
	BUR	BE	BRU	BE	BRU	BE
BERTje	297	484	300	493	297	492
RobBERT	325	527	328	537	323	534
mBERT	313	515	317	520	313	524
XLM-RoBERTa	413	667	412	675	407	654

Table 2: The training time in seconds for different BERT-based models for the two independent subtasks and the joint task. TC refers to the text classification task, SF refers to the slot filling task, and the Joint Task refers to the joint text classification and slot filling task. The reported training time for the joint task is for the Enhanced Joint BERT-based models. The training times for the Enhanced models and the non-Enhanced models are almost the same.

this section for more details). For the text classification task, based on the results, we observe that the BERTje model performs best in terms of  $F_{1c}$  score on both datasets. In particular, BERTje scores 98.56% and 97.31% on the BRU and the BE datasets, respectively. The BERT-based models perform better or on par with the non-BERT-based models on both datasets. The reason could be that BERT-based models can better encode the contextual information of the whole input sequence. On the BRU dataset, all the models achieve an accuracy score of around 98%, and for the BE dataset, the score is around 96%. On the slot filling task, the LSTM + CRF model outperforms all the other models on both datasets, and the score is 98.18% on the BRU dataset and 97.22% on the BE dataset. On the BRU dataset, the rest of the models achieve an  $F_1$  score of around 97%. On the BE dataset, the BERTje has a similar  $F_1$  score (97.18%) to the LSTM + CRF model. The rest of the models achieve an  $F_1$  score of around 96%. On both datasets, the BERT-based models can perform on par with the non-BERT-based models. We notice that the LSTM + CRF models have better performance compared to the models that rely only on LSTMs on both datasets. This is due to the fact that the CRF layer can take into account

		Model	$F_{1c}$			$F_{1s}$		
			BRU	BE	BRU→BE	BRU	BE	BRU→BE
Neural- based	CNN	97.96	96.73	95.97	-	-	-	
	LSTM	97.95	96.53	<b>96.56</b>	97.97	96.44	93.03	
	CNN + LSTM	98.19	96.28	96.09	-	-	-	
	LSTM + CRF	-	-	-	<b>98.18</b>	<b>97.22</b>	<b>95.24</b>	
BERT- based	BERTje	<b>98.56</b>	<b>97.31</b>	94.53	97.79	97.18	92.24	
	RobBERT	98.14	96.66	95.81	96.40	96.10	77.46	
	mBERT	98.35	97.08	94.82	97.98	96.87	90.03	
	XLM-RoBERTa	98.20	96.22	95.44	96.76	96.52	80.31	

Table 3: The results of the different models for the text classification and the slot filling tasks on the two datasets, as well as the generalization ability of the best performing BRU models for text classification and slot filling on the BE dataset (except the BRU part, see the BRU→BE column) in terms of  $F_1$  score.  $F_{1c}$  and  $F_{1s}$  indicate the  $F_1$  score for the text classification and the slot filling tasks, respectively. The two tasks are considered independently.

the relationships of the outputs between sequential tokens (i.e., a B-where tag cannot be followed by an I-when tag). The fact that the different independent models can achieve high  $F_1$  scores means that the traffic event detection problem from Twitter can be solved as the two independent subtasks we formulate.

**Transfer Learning:** Table 3 shows the results of the best performing BRU models on the BE dataset (except for the BRU part of the dataset, see the BRU→BE column) for the tasks of text classification and slot filling in terms of  $F_1$  score. For the text classification task, the results indicate that the non-BERT based models generalize better on this task, with 96.56% for the LSTM model, 96.09% for the CNN + LSTM model, and 95.97% for the CNN model in terms of classification  $F_{1c}$  score. However, the BERT-based models still generalize well with around 95% classification  $F_{1c}$  score. For the slot filling task, the LSTM + CRF has the best generalization ability. The reason could be that the CRF layer can learn the relationships between the neighboring tokens. The

	Model	BRU			BE			BRU→BE		
		F <sub>1c</sub>	F <sub>1s</sub>	SenAcc	F <sub>1c</sub>	F <sub>1s</sub>	SenAcc	F <sub>1c</sub>	F <sub>1s</sub>	SenAcc
LSTM-based	Slot-Gated	95.09	97.10	92.76	96.23	96.13	92.85	94.17	76.20	66.29
	SF-ID Network	97.23	97.03	94.10	96.07	96.67	93.84	84.78	83.29	69.03
	Capsule-NLU	97.75	97.72	95.79	96.71	96.76	94.21	95.92	91.45	87.60
Joint BERT	BERTje	98.34	97.31	95.79	96.71	96.81	93.93	94.18	92.11	86.23
	RobBERT	97.90	96.46	94.10	96.31	96.21	92.90	94.62	74.15	70.88
	mBERT	97.89	97.42	95.71	96.95	96.63	94.26	94.58	92.58	87.01
	XLM-RoBERTa	97.97	96.53	94.10	96.51	96.24	93.13	92.99	76.09	53.14
Enhanced BERT	BERTje	<b>98.49</b>	97.52	95.87	<b>97.13</b>	<b>97.10</b>	<b>94.63</b>	95.27	93.78	88.67
	RobBERT	98.12	96.65	94.72	96.53	96.40	93.41	94.96	80.00	74.84
	mBERT	97.92	<b>98.29</b>	<b>96.40</b>	96.82	96.78	94.59	<b>96.56</b>	<b>94.13</b>	<b>89.94</b>
	XLM-RoBERTa	98.20	96.65	94.56	96.26	96.18	92.76	95.76	74.21	72.57

Table 4: The results of different models for joint text classification and slot filling on the two datasets, as well as the generalization ability of the best performing trained BRU joint models on the BE dataset (except for the BRU part, see the BRU→BE column) in terms of F<sub>1</sub> score and sentence-level semantic frame accuracy (SenAcc). F<sub>1c</sub> and F<sub>1s</sub> indicate the F<sub>1</sub> score for the text classification and the slot filling tasks, respectively.

RobBERT model achieves the worst generalization performance. This is mainly because the RoBERTa model needs more data to be fine-tuned.

### 5.5.2. Joint Models

Table 4 shows the performance of the joint models (Slot-Gated, SF-ID Network, Capsule-NLU, Joint BERT-based, Enhanced Joint BERT-based) on the BRU and the BE datasets in terms of F<sub>1</sub> score and semantic accuracy (SenAcc). In the joint setting, the higher the semantic frame accuracy score, the better the joint model performance. On the BRU dataset, the Enhanced Joint mBERT achieves the best semantic accuracy score (i.e., 96.40%) as well as the best F<sub>1s</sub> score (i.e., 98.29%). The Enhanced Joint BERTje model has the best F<sub>1c</sub> score (i.e., 98.49%). However, on the BE dataset, the Enhanced Joint BERTje is the best performing model, and it achieves scores of 97.13% in terms of F<sub>1c</sub> score, 97.10% in terms of F<sub>1s</sub> score, and 94.63% in terms of SenAcc. Among the three non-BERT-based models, the Capsule-NLU model achieves the best results on both



datasets, and it performs on par with the BERT-based models. That is due to the ability of the model to capture hierarchical semantic relationships between the two subtasks. In other words, there are strong relationships between a traffic-related tweet and the fine-grained information (e.g., “when”, “where”, “what”, and “consequence”). As we observe in the results, the Enhanced Joint BERT-based models have better performance than the Joint BERT-based models. This indicates that the proposed Enhanced models are more effective for the joint task, mainly because the model can incorporate the entire sentence information into each token to boost the model performance. The BERT-based models use a self-attention mechanism to capture the information of each token by considering all other tokens in a tweet. They can represent each token by its context, while the LSTM-based models use fixed pretrained word embeddings (e.g., word2vec (Mikolov et al., 2013)) to represent each token. When solving the two subtasks, the Joint BERT-based models can represent the entire tweet and each token within the tweet by considering all the tokens in the tweet. We believe this is the reason why the Joint BERT-based models and the Enhanced variants can achieve better results than the LSTM-based models. Thus, we conclude that the Enhanced Joint BERT-based models, especially the BERTje and mBERT variants, can be used as strong baselines for the joint text classification and slot filling tasks for the traffic event detection problem in Belgium and in the Brussels capital region. Furthermore, the results of the different joint models prove the effectiveness of formulating traffic event detection as a joint task of the two subtasks.

**Transfer Learning:** Table 4 shows the generalization ability of the best performing BRU models for the joint task of text classification and slot filling. The Enhanced Joint BERT-based models obtain the best generalization performance not only compared to the Joint BERT-based models but also to the other models proposed for solving the task. This implies that the Enhanced Joint BERT-based model improves the model generalization ability by taking the whole information about the sentence for each token into account. Due to its architecture, the Capsule-NLU model can preserve the hierarchical relation-

ships between the two subtasks. That way, it is able to outperform all the other LSTM-based models and perform on par with the Joint BERT-based models.

### *5.6. Performance of the Independent and the Joint Models*

The results in Tables 3 and 4 indicate that the joint models perform on par with the independent models on the two subtasks. The benefit of joint training is not evident in that particular problem as it was in the case of other NLP problems (e.g., [Miwa & Bansal \(2016\)](#); [Hashimoto et al. \(2017\)](#)). We hypothesize that this is because the performance of all the models (joint and independent) for the two subtasks is already high in terms of  $F_1$  score and SenAcc. Thus, there is very little room for improvement in the joint models for the traffic event detection problem. However, since the joint model can address the two subtasks simultaneously, one advantage of training a joint model (for that particular problem) is that these models can be deployed easier since we have to train only one model instead of two models, i.e., one for each subtask. As shown in Table 2, compared to the time needed for the two independent BERT-based models for the two subtasks, the joint BERT-based models need only half of the time to train on the corresponding datasets.

The performance of the various models is high, and this could be potentially explained by the large number of traffic-related tweets that are coming from various organizations. Although the tweets (from the organizations) are not structured, they follow specific patterns (e.g., the “where” slots are usually followed by the “when” slots). This is also the case for the tweets presented in the work of [Dabiri & Heaslip \(2019\)](#), where the authors focused their study in the US, and the text classification performance in their work was also high. It is worthwhile mentioning, though, that this is not a disadvantage since it is crucial to have high  $F_1$  scores in a problem where the data could be used in real-time applications (e.g., send alert notifications to users).

## 6. Conclusion and Future Work

In this work, we define the new problem of detecting traffic-related events on Twitter that consists of two subtasks: (i) identify whether a tweet contains traffic-related events as a text classification task, and (ii) identify fine-grained traffic-related information (e.g., “when”, “where”) as a slot filling task. We also publish the two constructed Dutch traffic Twitter datasets to promote further research on detecting traffic-related events from social media.

We extensively experiment with several architectures for addressing the two subtasks either separately or in a joint setting. In this paper, we investigate several baseline methods, and we propose a certain modification in the BERT-based model (see our Enhanced Joint BERT-based model in Section 4.3), where we take into account the entire sentence information for each token to further improve the performance of the model. Experimental results indicate that the proposed Enhanced Joint BERT-based model is able to outperform all the other studied architectures for solving the task at hand jointly in terms of the semantic accuracy score in three scenarios, i.e., datasets from Belgium (BE), and the Brussels capital region (BRU), and the transfer learning experiment BRU→BE. The proposed model can achieve the best semantic accuracy scores, that is, 96.40% (with mBERT), 94.63% (with BERTje), and 89.94% (with mBERT), for the three scenarios.

The current traffic event detection models are mainly built for Dutch tweets. As there are three official languages in Belgium, in future work, we aim to work on multilingual models which can handle various languages. That way, we expect that these models can also be exploited to detect traffic-related events in other countries. Moreover, since Twitter is a noisy source of information, building a language model that is based on a large Twitter corpus, similar to the work of Müller et al. (2020), can potentially help us to improve the performance of our models. Another future work direction could be combining our traffic event detection system with downstream tasks, such as traffic knowledge graph and traffic forecasting (Jin et al., 2022; Xu & Liu, 2021; Wang et al., 2021; Jiang

& Luo, 2022; Wang et al., 2022a).

### Acknowledgements

This work has been supported in part by the MobiWave Innoviris Project and in part by the Flemish Government, under the “Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen” programme.

### References

- Adedoyin-Olowe, M., Gaber, M. M., Dancausa, C. M., Stahl, F., & Gomes, J. B. (2016). A rule dynamics approach to event detection in twitter with its application to sports and politics. *Expert Systems with Applications*, 55, 351–360.
- Ali, F., Ali, A., Imran, M., Naqvi, R. A., Siddiqi, M. H., & Kwak, K.-S. (2021). Traffic accident detection and condition analysis based on social networking data. *Accident Analysis & Prevention*, 151, 105973.
- Alomari, E., Mehmood, R., & Katib, I. (2019). Road traffic event detection using twitter data, machine learning, and apache spark. In *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCOM/IOP/SCI)* (pp. 1888–1895). IEEE.
- Bekoulis, G., Deleu, J., Demeester, T., & Develder, C. (2018a). An attentive neural architecture for joint segmentation and parsing and its application to real estate ads. *Expert Systems with Applications*, 102, 100–112.
- Bekoulis, G., Deleu, J., Demeester, T., & Develder, C. (2018b). Joint entity recognition and relation extraction as a multi-head selection problem. *Expert Systems with Applications*, 114, 34–45.

- Bekoulis, G., Deleu, J., Demeester, T., & Develder, C. (2019). Sub-event detection from twitter streams as a sequence labeling problem. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 745–750). Minneapolis, Minnesota: Association for Computational Linguistics. URL: <https://www.aclweb.org/anthology/N19-1081>. doi:10.18653/v1/N19-1081.
- Bekoulis, G., Papagiannopoulou, C., & Deligiannis, N. (2020). A review on fact extraction and verification. *arXiv preprint arXiv:2010.03001*, (pp. 1–35).
- Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 135–146. URL: <https://aclanthology.org/Q17-1010>. doi:10.1162/tacl\_a\_00051.
- Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1), 41–75.
- Castillo, C. (2016). *Big Crisis Data: Social Media in Disasters and Time-Critical Situations*. Cambridge University Press. doi:10.1017/CB09781316476840.
- Chang, H., Li, L., Huang, J., Zhang, Q., & Chin, K.-S. (2022). Tracking traffic congestion and accidents using social media data: A case study of shanghai. *Accident Analysis Prevention*, 169, 106618. URL: <https://www.sciencedirect.com/science/article/pii/S0001457522000549>. doi:<https://doi.org/10.1016/j.aap.2022.106618>.
- Chen, Q., Zhuo, Z., & Wang, W. (2019a). Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*, (pp. 1–6).
- Chen, Y., Lv, Y., Wang, X., Li, L., & Wang, F. (2019b). Detecting Traffic Information From Social Media Texts With Deep Learning Approaches. *IEEE Transactions on Intelligent Transportation Systems*, 20, 3049–3058.

- Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 8440–8451). Online: Association for Computational Linguistics. URL: <https://aclanthology.org/2020.acl-main.747>. doi:10.18653/v1/2020.acl-main.747.
- Dabiri, S., & Heaslip, K. (2019). Developing a Twitter-based traffic event detection model using deep learning architectures. *Expert systems with applications*, 118, 425–439.
- D’Andrea, E., Ducange, P., Lazzerini, B., & Marcelloni, F. (2015). Real-Time Detection of Traffic From Twitter Stream Analysis. *IEEE Transactions on Intelligent Transportation Systems*, 16, 2269–2283.
- Delobelle, P., Winters, T., & Berendt, B. (2020). RobBERT: a Dutch RoBERTa-based Language Model. In *Findings of the Association for Computational Linguistics: EMNLP 2020* (pp. 3255–3265). Online: Association for Computational Linguistics. URL: <https://aclanthology.org/2020.findings-emnlp.292>. doi:10.18653/v1/2020.findings-emnlp.292.
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4171–4186).
- E, H., Niu, P., Chen, Z., & Song, M. (2019). A novel bi-directional interrelated model for joint intent detection and slot filling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 5467–5471). Florence, Italy: Association for Computational Linguistics. URL: <https://aclanthology.org/P19-1544>. doi:10.18653/v1/P19-1544.

- Firdaus, M., Bhatnagar, S., Ekbal, A., & Bhattacharyya, P. (2018). A deep learning based multi-task ensemble model for intent detection and slot filling in spoken language understanding. In L. Cheng, A. C. S. Leung, & S. Ozawa (Eds.), *Neural Information Processing* (pp. 647–658). Cham: Springer International Publishing.
- Girshick, R. B. (2015). Fast R-CNN. *2015 IEEE International Conference on Computer Vision (ICCV)*, (pp. 1440–1448).
- Goo, C.-W., Gao, G., Hsu, Y.-K., Huo, C.-L., Chen, T.-C., Hsu, K.-W., & Chen, Y.-N. (2018). Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)* (pp. 753–757).
- Gu, Y., Qian, Z., & Chen, F. (2016). From Twitter to detector: real-time traffic incident detection using social media data. *Transportation Research Part C-emerging Technologies*, *67*, 321–342.
- Hakkani-Tür, D., Tür, G., Celikyilmaz, A., Chen, Y.-N., Gao, J., Deng, L., & Wang, Y.-Y. (2016). Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech* (pp. 715–719).
- Hashimoto, K., Xiong, C., Tsuruoka, Y., & Socher, R. (2017). A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (pp. 1923–1933). Copenhagen, Denmark: Association for Computational Linguistics. URL: <https://aclanthology.org/D17-1206>. doi:10.18653/v1/D17-1206.
- He, K., Gkioxari, G., Dollár, P., & Girshick, R. B. (2020). Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*, 386–397.
- Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*, 1735–1780.

- Huang, L., Yang, Y., Chen, H., Zhang, Y., Wang, Z., & He, L. (2022). Context-aware road travel time estimation by coupled tensor decomposition based on trajectory data. *Knowledge-Based Systems*, 245, 108596. URL: <https://www.sciencedirect.com/science/article/pii/S0950705122002672>. doi:<https://doi.org/10.1016/j.knosys.2022.108596>.
- Jiang, W., & Luo, J. (2022). Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 207, 117921. URL: <https://www.sciencedirect.com/science/article/pii/S0957417422011654>. doi:<https://doi.org/10.1016/j.eswa.2022.117921>.
- Jin, G., Liu, C., Xi, Z., Sha, H., Liu, Y., & Huang, J. (2022). Adaptive dual-view wavenet for urban spatial-temporal event prediction. *Information Sciences*, 588, 315–330. URL: <https://www.sciencedirect.com/science/article/pii/S0020025521013062>. doi:<https://doi.org/10.1016/j.ins.2021.12.085>.
- Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1746–1751). Doha, Qatar: Association for Computational Linguistics. URL: <https://aclanthology.org/D14-1181>. doi:[10.3115/v1/D14-1181](https://doi.org/10.3115/v1/D14-1181).
- Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *CoRR*, *abs/1412.6980*.
- Korpusik, M., Liu, Z., & Glass, J. (2019). A Comparison of Deep Learning Methods for Language Understanding. *Proc. Interspeech 2019*, (pp. 849–853).
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60, 84 – 90.



- Kurata, G., Xiang, B., Zhou, B., & Yu, M. (2016). Leveraging Sentence-level Information with Encoder LSTM for Semantic Slot Filling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (pp. 2077–2083).
- Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 260–270). San Diego, California: Association for Computational Linguistics. URL: <https://aclanthology.org/N16-1030>. doi:10.18653/v1/N16-1030.
- Larson, S., Mahendran, A., Peper, J., Clarke, C., Lee, A., Hill, P., Kummerfeld, J. K., Leach, K., Laurenzano, M., Tang, L., & Mars, J. (2019). An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction. *ArXiv, abs/1909.02027*.
- Li, C., Li, L., & Qi, J. (2018). A self-attentive model with gate mechanism for spoken language understanding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 3824–3833). Brussels, Belgium: Association for Computational Linguistics. URL: <https://aclanthology.org/D18-1417>. doi:10.18653/v1/D18-1417.
- Li, C., Sun, A., & Datta, A. (2012). Twevent: Segment-based event detection from tweets. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management CIKM '12* (p. 155–164). New York, NY, USA: Association for Computing Machinery. URL: <https://doi.org/10.1145/2396761.2396785>. doi:10.1145/2396761.2396785.
- Liu, B., & Lane, I. (2016). Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling. *Interspeech 2016*, (pp. 685–689).
- 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*, (pp. 1–13).

- Lu, P., Bai, T., & Langlais, P. (2019). SC-LSTM: Learning task-specific representations in multi-task learning for sequence labeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 2396–2406). Minneapolis, Minnesota: Association for Computational Linguistics. URL: <https://aclanthology.org/N19-1249>. doi:10.18653/v1/N19-1249.
- Michael, C., & Blason, J. (2014). The 100 most congested cities in Europe and North America. *The Guardian*, . URL: <https://www.theguardian.com/cities/gallery/2014/jul/07/most-congested-cities-europe-america-in-pictures-traffic-jams>.
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. URL: <http://arxiv.org/abs/1301.3781>.
- Miwa, M., & Bansal, M. (2016). End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1105–1116).
- Müller, M., Salathé, M., & Kummervold, P. E. (2020). Covid-twitter-bert: A natural language processing model to analyse Covid-19 content on twitter. *arXiv preprint arXiv:2005.07503*, (pp. 1–6).
- Naseem, U., Razzak, I., & Hameed, I. (2019). Deep Context-Aware Embedding for Abusive and Hate Speech detection on Twitter. *Aust. J. Intell. Inf. Process. Syst.*, 15, 69–76.
- Naseem, U., Razzak, I., Khushi, M., Eklund, P. W., & Kim, J. (2021). Covid-senti: A large-scale benchmark twitter data set for covid-19 sentiment analysis. *IEEE Transactions on Computational Social Systems*, 8, 1003–1015.

- Oostdijk, N., Reynaert, M., Hoste, V., & Schuurman, I. (2013). The Construction of a 500-Million-Word Reference Corpus of Contemporary Written Dutch. In *Essential Speech and Language Technology for Dutch*.
- Ordelman, R., Jong, F. D., Hessen, A. J. V., & Hondorp, G. (2007). TwNC: a Multifaceted Dutch News Corpus.
- Rashkin, H., Choi, E., Jang, J. Y., Volkova, S., & Choi, Y. (2017). Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (pp. 2931–2937). Copenhagen, Denmark: Association for Computational Linguistics. URL: <https://aclanthology.org/D17-1317>. doi:10.18653/v1/D17-1317.
- Saeed, Z., Abbasi, R. A., Maqbool, O., Sadaf, A., Razzak, I., Daud, A., Aljohani, N. R., & Xu, G. (2019). What’s happening around the world? a survey and framework on event detection techniques on twitter. *Journal of Grid Computing*, 17(2), 279–312.
- Saidi, F., Trabelsi, Z., & Thangaraj, E. (2022). A novel framework for semantic classification of cyber terrorist communities on twitter. *Engineering Applications of Artificial Intelligence*, 115, 105271. URL: <https://www.sciencedirect.com/science/article/pii/S0952197622003293>. doi:<https://doi.org/10.1016/j.engappai.2022.105271>.
- Sakaki, T., Okazaki, M., & Matsuo, Y. (2010). Earthquake shakes twitter users: Real-time event detection by social sensors. In *Proceedings of the 19th International Conference on World Wide Web WWW '10* (p. 851–860). New York, NY, USA: Association for Computing Machinery. URL: <https://doi.org/10.1145/1772690.1772777>. doi:10.1145/1772690.1772777.
- Sakaki, T., Okazaki, M., & Matsuo, Y. (2012). Tweet analysis for real-time event detection and earthquake reporting system development. *IEEE Transactions on Knowledge and Data Engineering*, 25(4), 919–931.

- Salas, A., Georgakis, P., & Petalas, Y. G. (2017). Incident detection using data from social media. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, (pp. 751–755).
- Sayce, D. (2020). Twitter Usage Statistics. URL: <https://www.dsayce.com/social-media/tweets-day/>.
- Shafiq, M., Tian, Z., Bashir, A., Du, X., & Guizani, M. (2020). Corrauc: a malicious bot-iot traffic detection method in iot network using machine learning techniques. *IEEE Internet of Things Journal*, *PP*, 1–1. doi:10.1109/JIOT.2020.3002255.
- Sicilia, R., Merone, M., Valenti, R., & Soda, P. (2021). Rule-based space characterization for rumour detection in health. *Engineering Applications of Artificial Intelligence*, *105*, 104389. URL: <https://www.sciencedirect.com/science/article/pii/S0952197621002372>. doi:<https://doi.org/10.1016/j.engappai.2021.104389>.
- Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., & Qin, B. (2014). Learning sentiment-specific word embedding for Twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1555–1565). Baltimore, Maryland: Association for Computational Linguistics. URL: <https://aclanthology.org/P14-1146>. doi:10.3115/v1/P14-1146.
- Tom (2020). TomTom Traffic Index. *TomTom Traffic Index*, . URL: [https://www.tomtom.com/en\\_gb/traffic-index/ranking/](https://www.tomtom.com/en_gb/traffic-index/ranking/).
- Tulkens, S., Emmerly, C., & Daelemans, W. (2016). Evaluating Unsupervised Dutch Word Embeddings as a Linguistic Resource. In N. C. C. Chair), K. Choukri, T. Declerck, M. Grobelnik, B. Maegaard, J. Mariani, A. Moreno, J. Odiijk, & S. Piperidis (Eds.), *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Paris, France: European Language Resources Association (ELRA).

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems*. Curran Associates, Inc. volume 30. URL: <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>.
- de Vries, W., van Cranenburgh, A., Bisazza, A., Caselli, T., van Noord, G., & Nissim, M. (2019). Bertje: A dutch bert model. *arXiv preprint arXiv:1912.09582*, (pp. 1–6).
- Vu, N. T. (2016). Sequential Convolutional Neural Networks for Slot Filling in Spoken Language Understanding. In *Proceedings of Interspeech 2016* (pp. 3250–3254).
- Wang, D., Al-Rubaie, A., Stincic, S., & Davies, J. (2017). Real-Time Traffic Event Detection From Social Media. *ACM Transactions on Internet Technology (TOIT)*, *18*, 1 – 23.
- Wang, L., Niu, J., & Yu, S. (2020). Sentidiff: Combining textual information and sentiment diffusion patterns for twitter sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, *32*, 2026–2039.
- Wang, P.-S., Sun, C.-Y., Liu, Y., & Tong, X. (2018). Adaptive O-CNN: A Patch-Based Deep Representation of 3D Shapes. *ACM Trans. Graph.*, *37*(6). URL: <https://doi.org/10.1145/3272127.3275050>. doi:10.1145/3272127.3275050.
- Wang, S., Lv, Y., Peng, Y., Piao, X., & Zhang, Y. (2022a). Metro traffic flow prediction via knowledge graph and spatiotemporal graph neural network. *Journal of Advanced Transportation*, .
- Wang, S., Zhang, M., Miao, H., Peng, Z., & Yu, P. S. (2022b). Multivariate correlation-aware spatio-temporal graph convolutional networks for multi-

- scale traffic prediction. *ACM Trans. Intell. Syst. Technol.*, 13(3). URL: <https://doi.org/10.1145/3469087>. doi:10.1145/3469087.
- Wang, Z., Jiang, R., Xue, H., Salim, F. D., Song, X., & Shibasaki, R. (2021). Event-aware multimodal mobility nowcasting. In *AAAI Conference on Artificial Intelligence*.
- Weld, H., Huang, X., Long, S., Poon, J., & Han, S. C. (2021). A survey of joint intent detection and slot-filling models in natural language understanding. *arXiv preprint arXiv:2101.08091*, (pp. 1–32).
- Wongcharoen, S., & Senivongse, T. (2016). Twitter analysis of road traffic congestion severity estimation. *2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, (pp. 1–6).
- Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J. R., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., & Dean, J. (2016). Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *ArXiv*, *abs/1609.08144*.
- Xu, G., Dong, J., Ma, C., Liu, J., & Cliff, U. G. O. (2022). A certificateless sign-cryption mechanism based on blockchain for edge computing. *IEEE Internet of Things Journal*, .
- Xu, G., & Hu, X. (2022). Multi-dimensional attention based spatial-temporal networks for traffic forecasting. *Wireless Communications and Mobile Computing*, 2022.
- Xu, H., Liu, B., Shu, L., & Yu, P. S. (2018). Double embeddings and CNN-based sequence labeling for aspect extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (pp. 592–598). Melbourne, Australia: Association for

- Computational Linguistics. URL: <https://aclanthology.org/P18-2094>. doi:10.18653/v1/P18-2094.
- Xu, M., & Liu, H. (2021). A flexible deep learning-aware framework for travel time prediction considering traffic event. *Engineering Applications of Artificial Intelligence*, 106, 104491. URL: <https://www.sciencedirect.com/science/article/pii/S0952197621003390>. doi:<https://doi.org/10.1016/j.engappai.2021.104491>.
- Yang, X., Bekoulis, G., & Deligiannis, N. (2020). imec-ETRO-VUB at W-NUT 2020 shared task-3: A multilabel BERT-based system for predicting COVID-19 events. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)* (pp. 505–513). Online: Association for Computational Linguistics. URL: <https://aclanthology.org/2020.wnut-1.77>. doi:10.18653/v1/2020.wnut-1.77.
- Zhang, C., Li, Y., Du, N., Fan, W., & Yu, P. (2019). Joint slot filling and intent detection via capsule neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 5259–5267). Florence, Italy: Association for Computational Linguistics. URL: <https://aclanthology.org/P19-1519>. doi:10.18653/v1/P19-1519.
- Zhang, J., Wang, Z., Zheng, Y., & Zhang, G. (2020). Cascade Convolutional Neural Network for Image Super-Resolution. *ArXiv*, *abs/2008.10329*.
- Zhang, X., & Wang, H. (2016). A joint model of intent determination and slot filling for spoken language understanding. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence IJCAI'16* (p. 2993–2999). AAAI Press.
- Zhang, Z., He, Q., Gao, J., & Ni, M. (2018). A deep learning approach for detecting traffic accidents from social media data. *Transportation research part C: emerging technologies*, 86, 580–596.

- Zhao, L., & Feng, Z. (2018). Improving slot filling in spoken language understanding with joint pointer and attention. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (pp. 426–431).
- Zhou, C., Sun, C., Liu, Z., & Lau, F. (2015). A C-LSTM Neural Network for Text Classification. *ArXiv*, *abs/1511.08630*.
- Zhou, Q., Wen, L., Wang, X., Ma, L., & Wang, Y. (2016). A Hierarchical LSTM Model for Joint Tasks. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data* (pp. 324–335). Springer.
- Zhu, S., & Yu, K. (2017). Encoder-decoder with focus-mechanism for sequence labelling based spoken language understanding. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (pp. 5675–5679).
- Zong, S., Baheti, A., Xu, W., & Ritter, A. (2020). Extracting COVID-19 Events from Twitter. *arXiv preprint arXiv:2006.02567*, (pp. 1–8).