Information Systems Frontiers (2021) 23:115–134
https://doi.org/10.1007/s10796-020-09994-3

Mining Social Networks to Detect Traffic Incidents

Sebastián Vallejos[1] · Diego G. Alonso[1] · Brian Caimmi[1] · Luis Berdun[1] · Marcelo G. Armentano[1] · Álvaro Soria[1]

Published online: 24 February 2020

Abstract
Social networks are usually used by citizens to report or complain about traffic incidents that affect their daily mobility. Automatically finding traffic-related reports and extracting useful information from them is not a trivial task, due to the informal language used in social networks, to the lack of geographic metadata, and to the large amount of non traffic-related publications. In this article, we address this problem by combining Machine Learning and Natural Language Processing techniques. Our approach (a) filters publications that report traffic incidents in social networks, (b) extracts geographic information from the textual content of the publications, and (c) provides a broadcasting service that clusters all the reports of the same incident. We compared the performance of our approach with state of the art approaches and with a popular traffic-specific social network, obtaining promising results.

Keywords Social networks · Natural language processing · Machine learning · Traffic incident detection

1 Introduction

One of the biggest problems in large cities is the planning of urban mobility. This issue arises, among other reasons, due to the existence of traffic incidents. In (Kamran & Haas, 2007), a traffic incident is defined as '*an unexpected event that temporarily disrupts the traffic flow on a segment of a roadway*'. Traffic incidents usually affect the daily journeys of many people. For example, traffic incidents such as car crashes or road closures represent obstacles for drivers that need to commute through the affected areas. In this context, counting with any information about the existence of traffic incidents becomes crucial to citizens in order to avoid the affected areas and improve urban mobility. However, traffic incidents are unexpected and unpredictable, making it difficult for citizens to be aware of their existence before starting a journey.

With the popularization of social networks, many citizens have adopted them as a source of information about traffic incidents. Citizens use to follow traffic reporters from social networks to check for any incident that might affect their journey, especially before departing. One of the main benefits of using social networks as a source of information, is that they allow ordinary people to take the role of reporters. For example, many people who witness or hear about a traffic incident tend to report it on social networks. Moreover, when citizens are stuck in a traffic jam, for example, they usually complain about the situation in social networks. The main advantage of social networks with respect to other sources of information is that citizens usually report incidents as soon as they occur, even often before online traffic news websites (D'Andrea, Ducange, Lazzerini, & Marcelloni, 2015). In this sense, social networks are promising sources of information for understanding a city's status and its transportation system (Sumalee & Ho, 2018).

The main problem with using social networks as a source of information about traffic is that people find it impractical. For example, it is uncomfortable and dangerous to read posts on social networks while driving. Moreover, in general purpose social networks, users can post about any subject, including their feelings, experiences, and knowledge to share with each other at any time (Zhou & Jin, 2011). Thus, it is difficult for users to filter and/or to find traffic reports in the information stream. Furthermore, the publications made in social networks usually use informal language, presenting grammatical and typographical errors, word abbreviations, incorrect capitalization, among other irregularities. These informalities hinder the automatic extraction of information.

In 2009 Google™ turned to crowdsourcing to improve the accuracy of its traffic predictions (Barth 2009). The traffic

✉ Sebastián Vallejos
sebastian.vallejos@isistan.unicen.edu.ar

[1] ISISTAN Research Institute, CONICET-UNICEN, Campus Universitario, Paraje Arroyo Seco, Tandil, Buenos Aires, Argentina

functionality provided by Google Maps continuously combines the data coming in from all existing mobile users on the road and sends it back by way of those colored lines on the traffic layers. The red color indicates traffic delays (not necessarily an incident), the orange color indicates medium amount of traffic, the green color indicates no traffic delays and the gray color indicates that Google doesn't have enough data to estimate the traffic flow for a particular section of road. Initially Google Maps provided a layer with information about delays in the roads, but it didn't provide real-time information about specific incidents (if any existed). In 2013, Google acquired Waze,[1] a traffic-specific social network, and enhanced Google Maps with some of the traffic update features provided by Waze (McClendon 2013). This acquisition allowed Google Maps to report real time incidents on the map from data reported to Waze.

Waze is probably one of the most popular e traffic-specific social networks. Waze is a traffic-specific social network with a community-driven GPS navigation app. This social network allows their users to interactively report traffic events in a map, such as car crashes, road closures and police traps, among others. In this way, users can check the map to be aware of traffic incidents, which is much simpler than reading tons of publications. However, Waze does not consider traffic-related reports of citizens that are not users of their app, losing a large number of traffic reports shared in other general purpose social networks.

In this context, we present a novel approach for detecting, interpreting, geolocating and disseminating traffic incidents written in natural language reported in social networks. To accomplish this goal, our approach combines Machine Learning (ML) and NLP techniques. First, our approach filters traffic-related reports from social networks data streams by using ML text classifiers. Then, our approach extracts information of the traffic incidents by using NLP techniques, such as an ad hoc Named Entity Recognizer (NER), pattern matching, and geocoding. It is worth noting that our approach is robust to different phrasing variations, grammatical and typographical errors, words abbreviations and capitalization irregularities, among other particularities that social network publications usually present. Finally, our approach geolocates and disseminates the detected traffic incidents in different ways, such as showing them in an online interactive map and broadcasting them in social networks.

We validated our approach by carrying out an experiment on Twitter.[2] We focus our analysis on two main factors: the ability of the ML techniques to filter traffic-related posts and the success rate in the semantic interpretation of traffic incidents. For the filtering stage, we evaluated the f-measure with $\beta = 2$ with a benchmarked dataset in English and with a dataset in Spanish reaching 96.4% and 96.7% respectively. We compared the results obtained with our approach with other state-of-art techniques (Dabiri & Heaslip, 2019; Pereira, Pasquali, Saleiro, & Rossetti, 2017; Carvalho, 2010; D'Andrea et al., 2015; Gu, Qian, & Chen, 2016). In the semantic interpretation stage, the proposed approach was able to correctly geolocate 96.33% of the traffic-related publications. Furthermore, we materialized our approach in a tool named Manwë. This tool continuously analyze publications from Twitter looking for reports of traffic incidents that took place in Buenos Aires city (Argentina). We compared the traffic incidents detected by Manwë and by Waze. The comparison showed that both approaches are complementary. In summary, the reported results provide encouraging evidence of the effectiveness and feasibility of the proposed approach.

The remainder of this paper is organized as follows. Section 2 discusses related works about traffic incident detection by analyzing the Twitter stream. Section 3 presents the proposed approach to detect traffic incidents from Twitter publications. Section 4 summarizes the evaluation of this approach. Section 5 introduces Manwë and compares it with Waze. Finally, Section 6 presents conclusions and future works.

2 Related Work

Event detection in general purpose social networks is a difficult task for several reasons: they occur in a small number of posts compared to the millions of posts published; they are generally relevant during a short time lapse; and they belong to a specific region, thus, being relevant to a reduced group of users. The research community has been attracted by this task over the past years and many authors have dealt with the detection of events that involve car crashes, traffic jams, fires, and protests.

Several works in the literature focused on classifying Twitter publications for the detection of traffic incidents using ML techniques. In (Schulz, Ristoski, & Paulheim, 2013), the authors presented an approach for detecting car incidents by classifying textual publications of the Seattle government's Twitter account into different categories. They compared the performance of Support Vector Machines (SVM), Naïve Bayes (NB), and Rule-based classifiers considering different types of NLP-based features previously detected in the publications. However, authors did not address the problems of the 'informal' language used in Twitter. The posts used to detect incidents were well-written and rarely present misspellings, since they were taken from the Seattle government. For extracting features from the posts the authors used the Stanford NER (C. Manning, Surdeanu, Bauer, & Finkel, 2014). Nevertheless, it has been demonstrated that Stanford NER accuracy decreases considerably when it faces the

1 https://www.waze.com/

2 https://www.twitter.com/

'informal' Twitter writing style (Ritter, Clark, Mausam, & Etzioni, 2011). In (D'Andrea et al., 2015), the authors presented a SVM model that recognizes useful keywords from Twitter publications and detects traffic incidents in the Italian road network. This approach is able to detect traffic incidents almost in real time, often before online traffic news websites. However, this approach is limited to detect traffic incidents occurred in the road network (i.e. roads and highways) and it does not detect traffic incidents occurred in the streets and avenues of the cities. In (Kuflik et al., 2017), the authors studied the challenges faced to successfully harvest traffic-related information from social media. In their evaluation, NB classifiers using n-gram features outperformed NB using unigram features, SVM and Decision Tree classifiers. The aforementioned approaches invest computing time in the enrichment of Twitter publications by using NLP techniques in order to improve the classification models. In contrast, the classification stage proposed in our approach aims at avoiding the waste of computing time in the analysis of irrelevant publications. Furthermore, our approach goes a step forward, since we add a semantic stage able to interpret the detected traffic incidents and to geolocate them in an interactive map.

Other authors have proposed different approaches for interpreting traffic incidents reported in social networks. The approach presented in (Endarnoto, Pradipta, Nugroho, & Purnama, 2011) extracts the information related to the traffic status published by the Traffic Management Center of Jakarta (Indonesia) on its Facebook and Twitter accounts. Publications shared by these accounts always have the same syntactic structure and they do not present orthographic or typographical errors. Therefore, this approach would not be able to interpret publications presenting other structures that may be shared by other users. Albuquerque et al. (Albuquerque et al., 2016) introduced a tool for interpreting traffic-related Twitter publications in Portuguese. This tool uses a text classifier during a NER process. Then, the tool uses the named entities to identify traffic incident components, such as the location of the incident. However, the tool focuses on well-written publications without considering spelling mistakes and wrong capitalization of words. This fact limits their approach to publications shared by traffic authorities and news agencies, which usually have fairly regular and simple syntactic structures and rarely contain spelling mistakes or wrong capitalizations. In other words, they did not consider reports made by witnesses, which generally presents several spelling mistakes.

Finally, other works followed a two-step approach that first filters traffic-related publications and then interprets the traffic incidents in the filtered publications. For example, the work introduced in (Wanichayapong, Pruthipunyaskul, Pattara-Atikom, & Chaovalit, 2011), employs a simple heuristic to filter traffic-related publications in microblogs from Thailand. Then, their approach extracts the name of the street, a starting point, and an ending point from a publication's text by means of a simple lookup of each token of the publication in a dictionary of streets and places. Nonetheless, if the tokens have typographical errors or any variation in the way of referring to a street or a place, then these entities would not be detected. In (Gu et al., 2016) authors fetch Twitter publications and use a modification of NB classifier to detect five major incident types. Then, their approach identifies the affected location by means of regular expressions and a fuzzy language matching algorithm. Both aforementioned works use a simple approach for identifying the incident location which is limited to detecting only one single incident per publication. In a more recent work (Dabiri & Heaslip, 2019), the authors proposed an approach that receives a stream of Twitter publications, identifies traffic-related publications by using deep learning techniques (particularly convolutional neural networks), and geocodes the detected traffic incidents. However, the proposed approach is not completely addressed. Particularly, the authors addressed the identification of traffic-related publications but not the detection and geocodification of traffic incidents.

Besides the semantic interpretation of the incident, our approach clusters all posts referring to the same incident in a single event, plots it in an online interactive map, and broadcasts it in a dedicated Twitter account. The information retrieved by our approach constitutes a valuable source of information that could be used by recommender engines that needs to know the traffic conditions that could affect the user mobility, such as those presented in (Bothorel, Lathia, Picot-Clemente, & Noulas, 2018).

3 The Proposed Approach

Many users have adopted social networks as a communication media to report traffic incidents. Professional TV and radio traffic reporters inform about the traffic status in their cities by using social networks to reach a bigger audience. In addition, ordinary citizens share their experience while travelling across the city or when they witness a traffic incident (such as a car crash or a road closure). These reports are extremely helpful for people that must drive through the areas affected by incidents. For instance, if a driver knows in advance that there are incidents affecting the path that he/she will take, he/she may plan a new route to avoid those areas. However, checking for traffic incidents reported on a social network is impractical and time consuming due to the high number and variety of publications. Furthermore, although there exists dedicated accounts for reporting traffic incidents, ordinary citizens driving in the city are usually aware of those incidents before traffic reporters. Moreover, it is uncomfortable and dangerous to read the publications or to report the incidents while driving. In addition, since traffic incident reports contain a textual location of the traffic incidents, drivers must invest time in analyzing whether the incident will affect their travel.

In this context, we propose an approach for detecting, analyzing, diffusing, and visualizing traffic incidents shared in Twitter publications. Particularly, we selected Twitter as the data source because it is a social network with millions of active users and it is well-known for the speed of the real-time propagation of topics and news. In addition, unlike other social networks, publications posted in Twitter are public and easy to obtain programmatically by using its streaming API. However it is important to note that our approach could use any other social network as the data source (as long as it offers a streaming API or a similar feature) The approach spreads the interpreted traffic incidents via two main mediums: (a) an interactive map where recent traffic incidents can be observed in an area of interest; and (b) a Twitter channel that summarizes all the traffic incidents gathered by the approach. Both mediums can be considered as valuable sources of information in which interested people or third party applications can query about current traffic incidents.

Figure 1 shows the information flow of our approach. Posts collected from Twitter are first preprocessed to remove useless information that might hinder the analysis in the remaining of the process. Preprocessed posts are then classified in order to identify traffic-related posts and discard non traffic-related posts. This first stage avoids performing unnecessary time consuming computations on non traffic-related posts in the next stage. In the second stage, the traffic-related posts are analyzed for identifying relevant information about the reported traffic incidents by using NLP techniques. Particularly, information such as the type of incident and the affected area are identified. Notice that if any non traffic-related post is classified as traffic-related in the first stage, it will probably be discarded in the second stage since the approach would not be able to detect an incident type or a location from the post. Finally, all posts reporting the same incident are grouped together and published online. As mentioned before, the information retrieved could be used as a source of information for a GPS navigator or a smartphone app for helping people to move through a city.

3.1 Text Preprocessing

Since users' posts are not always syntactically correct, the approach starts with a text preprocessing step that normalizes each publication. This normalization process includes: splitting CamelCase strings in the text, e.g., 'GardenStateParkway' is splitted into 'Garden State Parkway'; lowercasing to avoid case-sensitive variations; deleting letters repeated more than twice,[3] e.g., 'streeet' is replaced by 'street' and 'nooooo' is replaced by 'noo'; removing accents marks (common in many languages such as Spanish, French or Italian); removing hashtags symbols (#); removing references to other users and removing links/URLs. At the end of this stage, the approach has the preprocessed text that serves as input for the following stages. Notice that although this stage facilitates the analysis of publications in the following stages, it does not address other types of writing errors such as the existence of spelling mistakes (which are addressed later).

3.2 Identification of Traffic-Related Publications

After the text preprocessing stage, the approach uses ML classifiers to predict whether each publication has to be considered as traffic-related or not. Prior to the classification, the proposed approach makes some extra modifications to the text of each publication in order to enhance the classification. First, it removes stopwords[4] and some punctuation marks such as commas and periods, since they negatively affect the prediction precision of the ML techniques (Mandal & Sen, 2014; Schneider, 2005). Then, the approach replaces numbers in the text by the tag [number-L] where L is the length of the number, e.g., '420' is replaced by '[number-3]'. The approach transforms the resulting text into the Vector Space Model (VSM) representation (C. D. Manning, Raghavan, & Schütze, 2008). This is necessary because most ML classifiers do not handle plain text. It is important to note that, the original text of the publication is kept for the second stage because the removed stopwords, punctuation marks, and numbers are a valuable source of information for detecting traffic incidents in the publication.

Once the preprocessed text is represented using VSM, the approach uses different supervised ML classifiers to simultaneously predict a category ('traffic-related' or 'non traffic-related') for each given publication. Then, the approach classifies a publication as traffic-related if at least one of the classifiers predicted it as relevant; otherwise, the publication is classified as non traffic-related and it is discarded. In this way, the approach tries to increase the recall of traffic-related reports, that is, it aims to discard the smallest possible number of traffic-related publications. At this point, misclassifying and discarding a traffic-related publication means 'losing traffic information'. On the contrary, misclassifying and analyzing a non traffic-related tweet does not mean 'false traffic information'. The second stage discards the non traffic-related publication after unsuccessfully trying to extract information about traffic incidents from its text. In this sense, we consider that it is preferable to analyze a few number of

[3] We decided to keep two repeated letters at maximum because some words and names have repetitions of letters (e.g. 'Saavedra' and 'calle' in Spanish or 'street' and 'traffic' in English).

[4] Stopwords are irrelevant words such as articles or prepositions that have a high probability of occurrence in the publications, regardless of the topic discussed.

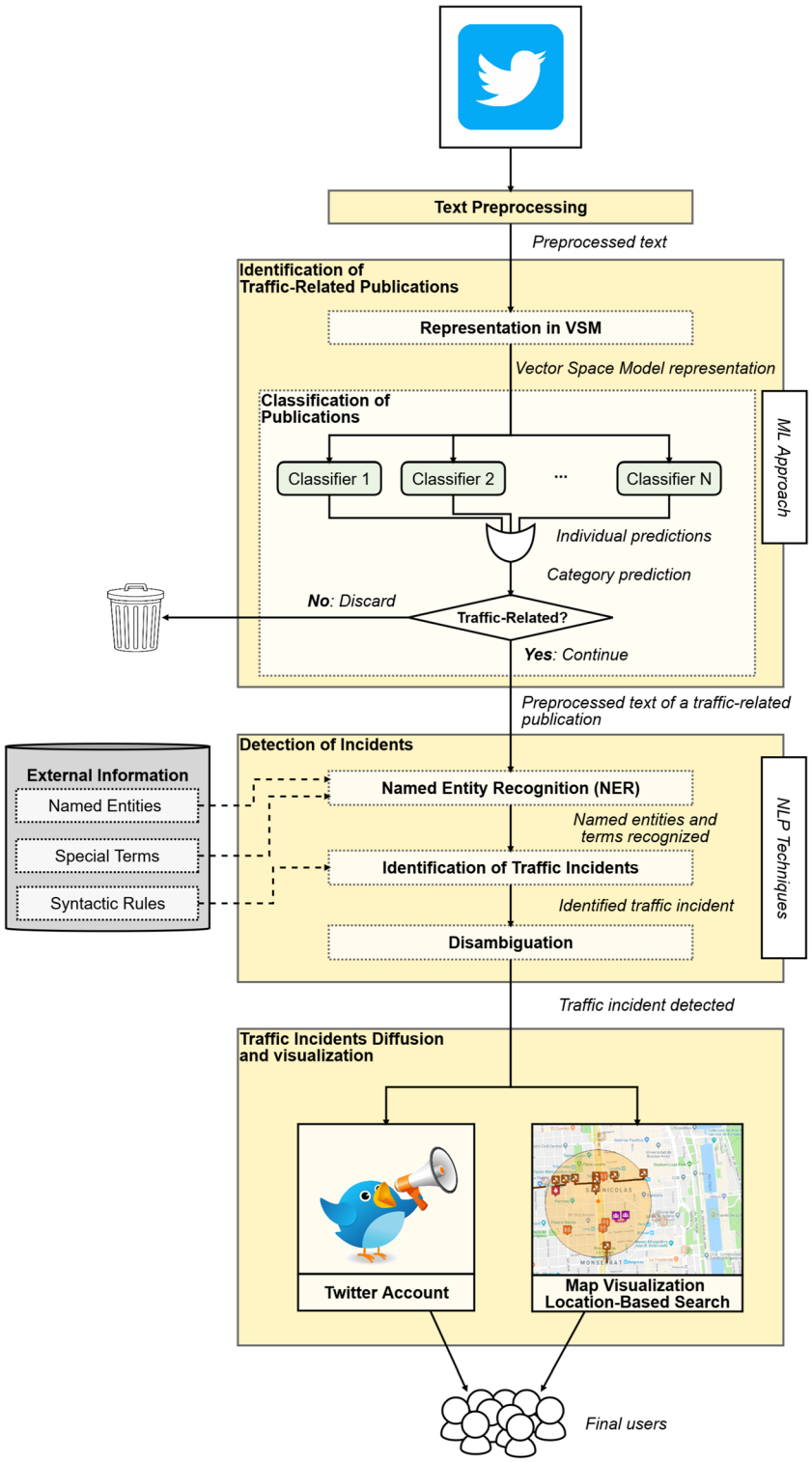

Fig. 1 The proposed approach for detecting traffic incidents

non traffic-related publications due to a misclassification than discarding publications that contain traffic-related information. The publications classified as traffic-related in this stage serve as input for the next stage.

3.3 Detection of Incidents

In this stage, the approach analyzes the preprocessed text of the traffic-related publications to extract relevant information

about traffic incidents. This stage consists in three steps: (a) *Named Entity Recognition*; (b) *Identification of Traffic Incidents*; and (c) *Disambiguation*.

3.3.1 Named Entity Recognition

The Named Entity Recognition step takes as input the preprocessed text of the publications (Section 3.1) and analyzes it looking for named entities and special terms. A named entity is a word or a sequence of words that can be classified into predefined categories (Kozareva, Bonev, & Montoyo, 2005) (e.g. the phrase 'car crash' belong to the category 'traffic incident'), while special terms refer to words such as connectors, pronouns, prepositions, and terms related to the traffic jargon (e.g. 'car', 'highway', 'intersection').

The recognition of named entities and special terms in social networks publications is a difficult task, because this kind of publications often presents an 'informal' writing style that reduces the precision of traditional NER approaches. For example, the traditional NER approach usually relies on in word capitalization for recognizing named entities. However, social network posts commonly present capitalization errors. Thus, the traditional NER approach falls on an unreliable and unpredictable solution (Ritter et al., 2011). Moreover, social networks publications may exhibit spelling mistakes, abbreviations and an incorrect use of accent marks. Furthermore, different users might use different alternatives to refer to a particular entity in natural language. For instance, the entity 'Highway' can be referred to as 'highway', 'hwy', 'HW', among others.

Another factor that negatively affects the use of traditional NER approaches in this context is that a text fragment can reference different named entities. For instance, the same name can be simultaneously used for referring to both a street and an avenue in the same city. In this sense, the approach needs to univocally identify which entity is being referred to in a post for correct geolocation of the reported incident. However, publications usually do not contain the context required to identify which particular entity is being referred to in the reported incident. For this reason, if a text fragment can refer to several named entities, it is important to consider all alternatives instead of selecting a single one as traditional approaches do.

To address the aforementioned problems, our approach uses an ad hoc NER. This NER combines two techniques: regular expressions and Approximate String Matching (ASM) (Yang, Yu, & Kitsuregawa, 2010). The regular expressions technique is useful to recognize named entities that follow a pattern (e.g. dates, times, numbers). For example, the regular expression '[0–9]+' recognizes entities belonging to the category 'Number'. In this way, this regular expression may recognize the house number part of an address. Nevertheless, this technique is not the best alternative to deal with spelling mistakes and incorrectly written named entities. In contrast, ASM allows the recognition of named entities and special terms even if they are misspelled. The process of ASM involves two tasks. Firstly, it splits the preprocessed text of a publication using white spaces and punctuation marks as delimiters, and generates a list of all the possible text fragments or n-grams. Secondly, this technique looks for similarities between the text fragments and the entries of a customizable catalogue. Each entry of the catalogue may define either a special term or a traffic-related named entity (e.g. road names or types of incidents) indicated by a list of possible alternative texts and its corresponding category. For example, the named entity 'Highway 92' belongs to the 'Road' category and it has the following alternative texts: 'highway 92', 'hway 92', 'hw 92' and 'hwy 92'.

To detect similarities during the second step, the ASM technique calculates the proximity score between each text fragment and each mention of the entries of the catalogue. The proximity score between a text fragment T_f and a catalogue entry C_e is calculated using Eq. 1. In this equation, $Lev(C_e,T_f)$ is the Levenshtein distance (Levenshtein, 1996) between a catalogue entry and a text fragment. The Levenshtein distance between two texts is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to convert one text into another. The term $length(C_e)$ refers to the length of the special term or the entity mention that are being compared. The proximity score represents the percentage of matching characters between two texts. A proximity score value equal to 1 indicates a perfect match (i.e., identical texts). Proximity score values less than 1 describe partial matches between the texts. For instance, Eq. 2 shows that the resultant proximity score considering the text fragment T_f = 'higway' and the corresponding catalogue entry C_e = 'highway' is 0.857.

$$ProximityScore(Ce, Tf) = 1 - \frac{Lev(Ce; Tf)}{length(Ce)} \tag{1}$$

$$ProximityScore\left('highway', 'higway'\right) = 1 - \frac{Lev('highway', 'higway')}{length('highway)} = 1 - \frac{1}{7} = 0,857 \tag{2}$$

The approach compares every proximity score with a predefined customizable threshold. If the proximity score is greater than or equal to this threshold, the approach accepts the recognized entity; otherwise, the approach disregards the entity. For example, assuming a threshold value of 0.8, the ASM technique would accept the recognition of the named entity 'Highway' from the text 'higway' since 'highway' is similar enough to 'higway'.

On the one hand, each named entity recognition consists of the named entity detected, the text that matched the entity, the category of the entity (for example 'Road' in the example

above), and the proximity score with which they were recognized in the preprocessed text of the publication. On the other hand, each special term recognition consists of the special term detected, the text that matched the term, and the proximity score.

3.3.2 Identification of Traffic Incidents

In this step, the approach identifies traffic incidents by performing a pattern matching process. This process consists in making associations (with a set of syntactic rules) between the named entities and the special terms recognized in each publication to identify traffic incidents.

A syntactic rule looks for a sequence of named entity categories and special terms that satisfy a predefined pattern. If a sequence of named entities categories and special terms matches the antecedent of a syntactic rule, then that sequence of elements is replaced by the consequent of the rule. For example, a rule R1 "<Road> 'and' <Road > → [Intersection]" looks for two named entities with the road category and the special term 'and' between them. If this sequence exists, the rule recognizes a 'location' component with the category 'Intersection'. In this way, a syntactic rule can identify traffic incident components (i.e., incident type and location). Following the example, a rule R2 "<Incident Type> 'at' [Intersection] → [Traffic Incident]" associates an incident type and the special term 'at' with the traffic incident component derived by R1, thus, detecting a traffic incident with all the required elements.

In the pattern matching process, the approach iteratively attempts to apply each syntactic rule until no rule can be applied. In addition, if a text fragment has assigned several alternatives of named entities or special terms, the approach considers every possible alternative in the pattern matching process. When the approach is not able to recognize the named entities and special terms required to identify a traffic incident (i.e. incident type and location) in a publication, the publication is discarded.

3.3.3 Disambiguation

At this point, the approach has identified the traffic incidents reported in a publication. However, a traffic incident may have several semantic meanings due to the recognition of different named entities or special terms in a same text fragment. For instance, if there are two different roads recognized in a same text fragment (e.g. when an avenue and a street have the same name), the detected incident will have two different interpretations, each one referring to a different location. Therefore, the approach assesses the different interpretations to identify the correct one.

In the disambiguation step, the approach gathers the traffic incident information from each interpretation and structures this information on a template. A template has the fields to represent a traffic incident such as the incident type, the location, date and time of detection, and the geographic coordinates. The incident type and the location are the result of the previous step of the approach and the date and time is obtained from the tweet metadata. Then, the approach uses the location field of the template to obtain the corresponding geographic coordinates. To do this, the approach uses a geocoding service that accepts a textual location as input and returns the corresponding geographic coordinates as output. Currently, the prototype of the approach relies on the Google Geocoding API[5] as the geocoding service. Nonetheless, any other geocoding service can be used instead. The request to the geocoding service is generated by combining the textual location described in the respective template with the name of the city and the country that is being monitored. If a request succeeds, the approach adds the geographic coordinates into the corresponding field of the template. If the request fails, the location may not exist, so the approach disregards the respective template. Notice that it is possible that more than one template is generated from a single incident since a given name can refer to different places.

The approach generates the templates for each incident reported in the publication. This allows us to detect every traffic incident referred on a publication that reports several traffic incidents. At the end of this step, it is expected to have one single template for each detected traffic incident. If all the templates of a traffic incident were discarded, we assume that the reported location does not exist. Otherwise, if two or more templates can be applied to the same traffic incident, the report is considered ambiguous. In both cases, the traffic incident is disregarded. The result of this stage consists of a list of templates completed with all the required information.

3.4 Traffic Incident Diffusion and Visualization

In the last stage, the approach groups all the publications that refer to the same incident and discloses the detected incidents using two information media. The first medium for disclosing incidents is a Twitter account where the approach reports every traffic incident detected. In this context, traffic reports consist of the original text of a publication informing a traffic incident, the original Twitter account that warned about the traffic incident, and a link to the traffic incident on the map. Thus, the Twitter account keeps track of all the traffic incidents detected by the proposed approach.

The second medium is an online interactive map. This map includes all the incidents detected in a recent time frame and shows the information stored in the template of every incident. For example, the incident type field is used to determine the marker icon used in the map. Moreover, the corresponding

5 https://developers.google.com/maps/documentation/geocoding/

marker is located on the map using the geographical coordinates' field. By looking at this map, anybody can have a birds-eye view of the real-time traffic status for a given area of interest.

Furthermore, the traffic incidents detected may be useful for third-party applications. For example, a location-based recommender system that needs to know in advance the current traffic status that might affect a user's mobility. In this sense, if a user is using a location-enabled device (e.g. a smartphone), then its current GPS position can be used to discover which traffic incidents are within a given distance radius as shown in Fig. 2 or which traffic incidents may affect his/her commute to a specific destination. Thus, a user has a simplified vision of nearby traffic incidents according to his/her current location and destination.

4 Evaluation

We carried out different experiments to evaluate the effectiveness of our approach. In Section 4.1 we evaluated the first stage of the approach, i.e. the ability of the approach to identify traffic-related publications. Moreover, we compared the results achieved by our approach with the results achieved by current state-of-art techniques (Dabiri & Heaslip, 2019; Pereira et al., 2017; Carvalho, 2010; D'Andrea et al., 2015; Gu et al., 2016). Then, in Section 4.2, we evaluate the second stage of the approach, i.e. ability of the approach to detect traffic incidents reported in Twitter publications.

4.1 Identification of Traffic-Related Publications

The identification of traffic-related publications was evaluated by using two datasets of different languages, Spanish and English. First (Section 4.1.1), we evaluated the use of multiple classifiers with respect to the use of a single classifier for each dataset. From the results of this experiment, we selected the best classifier combination for both datasets. Then (Section 4.1.2), we compared the selected combination of classifiers with five classification approaches present in the literature (Dabiri & Heaslip, 2019; Pereira et al., 2017; Carvalho, 2010; D'Andrea et al., 2015; Gu et al., 2016).

The Spanish dataset (SpD) contains 6060 tweets written in Spanish. These tweets were collected via the Twitter Streaming API using query parameters to fetch tweets reporting traffic incidents in Buenos Aires city (CABA), Argentina. The query parameters used involved a list of Twitter users or keywords that CABA's citizen usually mention when report traffic incidents or complain about traffic (such as the government official accounts, traffic reporters, journalists accounts, the civil defense account, and the hashtag #TransitoBue). We manually labeled each collected tweet as: 'traffic-related' if it contained the required two traffic incident components (i.e. the *incident type* and *location*); 'non traffic-related' otherwise. The collected dataset was balanced with respect to the number of posts belonging to each category by using down sampling, since standard ML techniques yield better prediction performance when they are trained using a balanced dataset (Ertekin, Huang, Bottou, & Lee, 2007). Thus, the balanced dataset consisted of 2979 traffic-related

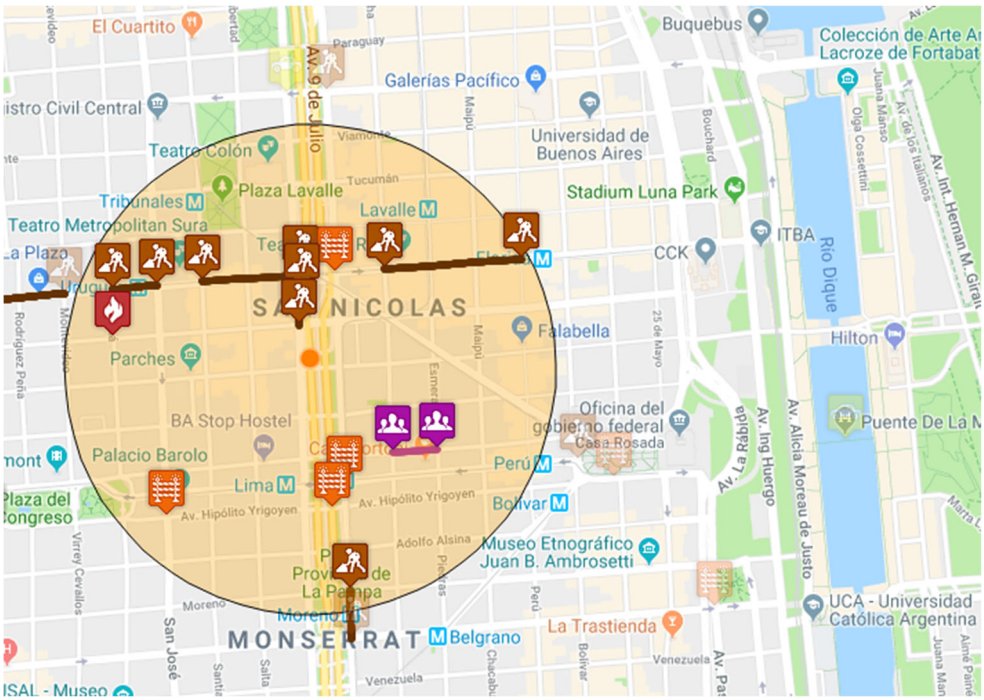

Fig. 2 A user shares his/her current GPS position to discover traffic incidents within a given distance radius

publications (49.15%) and 3081 non traffic-related publications (50.85%).

The English dataset (EnD) was published in (Dabiri & Heaslip, 2019). It contains 16,203 tweets written in English. These tweets were collected via Twitter Search API with query parameters to fetch tweets reporting traffic incidents in US. In particular, authors did multiple queries combining different keywords from a dictionary of traffic-related keywords and specifying within the boundaries of US. Then, the authors manually labelled each collected tweet as: 'traffic incident' if the tweet reports a non-recurring event that generate an abnormal increase in traffic demand, e.g. a car crash; 'traffic conditions and information' if the tweet reports traffic flow conditions, e.g. a traffic congestion; 'non-traffic' otherwise. For our experiment, we grouped the first two categories as 'traffic-related' and leave the third category as 'non traffic-related' in order to match our two categories scheme. Thus, the final dataset is balanced, containing 8116 traffic-related publications (50.08%) and 8087 non traffic-related publications (49.91%).

4.1.1 Evaluating Classifier Combinations

First, we evaluate the use of multiple classifiers with respect to the use of a single classifier for both datasets. It should be noted that, adapting our classification approach for a particular language is as simple as setting the list of stopwords corresponding to that language. During this experiment, we considered five different classification techniques: Support Vector Machines (SVM) with Linear Kernel, Discriminative Multinomial Naïve Bayes (DMNBText), VotedPerceptron (VP), C4.5 Decision Tree (J48), and Instance-based Learning (IBk). This gives a total of five single classifiers and 26 possible combinations.

The effectiveness of the individual and combined classifiers was evaluated in terms of precision, recall and f-measure metrics. In order to calculate these metrics it is necessary to define them. Precision (Eq. 3) measures the ratio between the number of traffic-related publications correctly classified (TP) and the total number of publication classified as traffic-related ($TP + FP$). In this work, a low precision causes a high number of non traffic-related publications to reach the second stage of the approach. Recall (Eq. 4) measures the ratio between the number of traffic-related publications correctly classified (TP) and the total number of existing traffic-related publications ($TP + FN$). A high recall indicates that a low number of relevant publications were discarded. F-measure (Eq. 5) is a retrieval measure that focuses on precision and recall simultaneously by weighting them with a real value called β. Generally, the f-measure with $\beta = 1$ is used to represent the harmonic mean between precision and recall. When $\beta < 1$, the metric weights precision heavier than recall. Instead, when $\beta > 1$, the metric weights recall heavier than precision. As mentioned before, misclassifying a non traffic-related tweet (FP) does not mean 'false traffic information' since the second stage will discard the publication after further analysis. On the contrary, misclassifying and discarding a traffic-related publication that contains valuable information (FN) means 'losing traffic information'. Therefore, as the cost of FNs is higher than the cost of FPs, using a $\beta = 2$ is preferred (Köknar-Tezel & Latecki, 2009).

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$F_{measure}(\beta = 2) = \left(1 + \beta^2\right) \cdot \frac{Precision \cdot Recall}{\left(\beta^2 \cdot Precision\right) + Recall}$$

$$= 5 \cdot \frac{Precision \cdot Recall}{(4 \cdot Precision) + Recall} \tag{5}$$

We employed a 10-fold cross-validation in order to test each single classifier and each possible combination of classifiers with the two mentioned datasets. Table 1 and Table 2 details the results obtained by the classifiers for the SpD and the EnD respectively. The first column of each table indicates the number of combined classifiers, while the second column specifies the classifiers combined. The remaining columns show the values of the precision, recall and f-measure ($\beta = 2$) reached by each combination. The tables first present the five single classifiers. Then, since the number of possible classifier combination raises to 26, the tables present a reduced set of classifier combinations. In particular, the tables present the combinations of classifiers with highest values of each metric for each N-subset. A N-subset if the subset of classifier combinations that combine N classifiers. For example, the best classifier of the 2-subset (i.e. the best combination of two classifiers) in terms of recall and f-measure for SpD (Table 1) is SVM + DMNB. The highest values of each N-subset are in bold, while the highest values of the entire experiment are also marked with an asterisk.

As Table 1 and Table 2 show, as the number of combined classifiers increases, the maximum values of each metric vary in different ways. On the one hand, the precision of the classifiers tends to decrease when they are combined, while their recall increases. This occurs because the approach categorizes a publication as traffic-related if at least one classifier categorized it as traffic-related (increasing the TP and the FP). The maximum recall for an individual classifier (0.9722 for SpD and

Table 1 Classification results of different classifiers combination with the SpD

Quantity	Classifiers	Precision	Recall	F-Measure ($\beta = 2$)
1	DMNB	0.9136	**0.9722**	**0.9599**
	SVM	0.9587	0.9468	0.9492
	VP	0.9504	0.9468	0.9475
	J48	0.9292	0.9050	0.9097
	IBk	**0.9674***	0.8542	0.8747
2	SVM + DMNB	0.9052	**0.9825**	**0.9660**
	SVM + IBk	**0.9507**	0.9595	0.9577
3	**SVM + DMNB + IBk**	0.9009	**0.9852**	**0.9671***
	SVM + VP + IBk	**0.9340**	0.9712	0.9635
4	SVM + DMNB + VP + IBk	0.8940	**0.9863**	**0.9663**
	SVM + VP + J48 + IBk	**0.9067**	0.9770	0.9621
5	SVM + DMNB + VP + IBk + J48	**0.8743**	**0.9880***	**0.9630**

0.9638 for EnD) was improved with the use of five classifiers to 0.9880 for SpD and 0.9849 for EnD. On the other hand, the maximum precision for individual classifiers (0.9674 for SpD and 0.9806 for EnD) was reduced with the use of five classifiers to 0.8743 for SpD and 0.8798 for EnD.

The interesting thing about this experiment arises when analyzing the f-measure metric. The value of the f-measure ($\beta = 2$) tends to increase when classifiers are combined. This occurs because the f-measure metric (with $\beta > 1$) weights recall more than precision. However, while the recall increases as more classifiers are combined, the f-measure finds a maximum point when combining 3 classifiers. From this point, the f-measure begins to decrease. This occurs with both datasets, indicating that combining more than 3 classifiers increases the number of FP much more than it reduces the number of FN. In this sense, the classifier that maximized f-measure ($\beta = 2$) metric for both datasets is the combination of SVM, DMNB and IBk. This combination of classifiers reaches a high value of recall (0.9852 for SpD and 0.9789 for EnD) without losing much precision (0.9009 for SpD and 0.9089 for EnD).

4.1.2 Comparing Classification Approaches

Once we determined the combination of classifiers that maximizes the f-measure ($\beta = 2$), we compared the results of our approach with other classification approaches present in the literature (Dabiri & Heaslip, 2019; Pereira et al., 2017; Carvalho, 2010; D'Andrea et al., 2015; Gu et al., 2016). The

Table 2 Classification results of different classifiers combination with the EnD

Quantity	Classifiers	Precision	Recall	F-Measure ($\beta = 2$)
1	DMNB	0.9298	**0.9638**	**0.9568**
	VP	0.9504	0.9486	0.9489
	SVM	0.9508	0.9486	0.9486
	J48	0.9282	0.9212	0.9226
	IBk	**0.9806***	0.7579	0.7940
2	SVM + DMNB	0.9121	0.9776	**0.9638**
	DMNB + J48	0.8948	**0.9787**	0.9607
	SVM + IBk	**0.9452**	0.9551	0.9531
3	**SVM + DMNB + IBk**	0.9089	0.9789	**0.9640***
	SVM + DMNB + J48	0.8860	**0.9837**	0.9624
	SVM + VP + IBk	**0.9322**	0.9657	0.9588
4	SVM + DMNB + VP + IBk	0.9023	0.9805	**0.9638**
	SVM + DMNB + VP + J48	0.8819	**0.9845**	0.9622
	SVM + VP + J48 + IBk	**0.9025**	0.9755	0.9600
5	SVM + DMNB + VP + IBk + J48	**0.8798**	**0.9849***	**0.9619**

Table 3 Classification results for SpD and EnD

Classifiers	Precision		Recall		F-Measure ($\beta = 2$)	
	SpD	EnD	SpD	EnD	SpD	EnD
Our approach	0.9009	0.9089	**0.9852**	**0.9789**	**0.9671**	**0.9640**
(Dabiri & Heaslip, 2019)	0.9567	0.9546	0.9494	0.9534	0.9509	0.9537
(Pereira et al., 2017)	**0.9639**	0.9502	0.9593	0.9466	0.9602	0.9473
(Carvalho, 2010)	0.9631	0.9493	0.9625	0.9460	0.9626	0.9467
(Gu et al., 2016)	0.9015	0.9272	0.9434	0.9443	0.9347	0.9408
(D'Andrea et al., 2015)	0.9512	**0.9619**	0.8741	0.8276	0.8885	0.8514

implementations of these approaches were published in (Dabiri & Heaslip, 2019). We evaluated these five approaches with both datasets (EnD and SpD). Since these approaches were prepared to work with publications written in English, we had to adapt them in order to work with the SpD. In particular, we used the same list of Spanish stopwords for those approaches that require a list of stopwords as parameter (Dabiri & Heaslip, 2019; Carvalho, 2010; D'Andrea et al., 2015). Moreover, (Dabiri & Heaslip, 2019) and (Pereira et al., 2017) use word embeddings. In this case, we used a Spanish model pre-trained from Wikipedia.[6] Table 3 resumes the results obtained from this experiment for both datasets (SpD and EnD). The first column indicates the classification approach, while the remaining columns show the values of the precision, recall and f-measure ($\beta = 2$) reached by each approach for each dataset. The highest values for each metric and each dataset are in bold.

Our approach reached the highest recall and the highest f-measure ($\beta = 2$) with both datasets. This was at the cost of reaching a lower precision than the rest of the approaches. However, as we previously mentioned, this is part of the desired behavior of reducing the number of publications to be analyzed in the second stage losing the least amount of traffic-related publications as possible. This is because the second stage of the approach will discard the misclassified non traffic-related publications after unsuccessfully trying to extract information about traffic incidents from them. It is worth noting that, our approach only needs a list of stopwords as configuration parameter. This is an advantage compared to approaches that use word embeddings, since not all languages have word embedding models as well trained as English and Spanish.

4.2 Detection of Traffic Incidents

The detection of traffic-related publications was evaluated by using the SpD. For this experiment, the EnD was not considered since it contains tweets reporting incidents in multiple cities in the US and we are not aware to which city refers each tweet. In this sense, it is impossible for our approach to geolocate the traffic incidents detected, since a single address may be valid for multiple cities. Even in the article in which the EnD dataset is presented (Dabiri & Heaslip, 2019), authors do not address the geocoding of traffic incidents, recognizing the lack of information needed to carry out this task. With this in mind, we evaluated the detection of traffic incidents with the SpD, which only contains tweets from Buenos Aires city. First (Section 4.2.1), we evaluated the NER step and selected the optimal threshold for the Approximate String Matching technique. Then (Section 4.2.2), we evaluated the success rate of the approach in the detection and geolocalization of traffic incidents.

4.2.1 Evaluating the NER Step

As it was stated in previous sections, the approach needs a catalogue of named entities and regular expressions to perform this step. Therefore, we defined a catalogue in order to evaluate the effectiveness of the NER step. Particularly, this catalogue was composed of 4335 named entities related to Buenos Aires city. Most of the named entities were obtained from the official local government web site.[7] Furthermore, four regular expressions were specified in order to detect named entities that follow a particular pattern. These regular expressions are described in Table 4.

After the setup, the NER step was performed with a random fold of the cross-validation of the first experiment. During the experiment we considered different threshold values for the Approximate String Matching technique, varying it from 0.7 to 1.0, with steps of 0.025. The NER step was evaluated with respect to the f-measure ($\beta = 1$) metric since this is an appropriate metric to discover which threshold value maximizes both precision and recall at the same time.

Figure 3 illustrates the experimental results used for assessing the NER step by changing the threshold value of the ASM technique. The horizontal axis indicates the values

[6] https://fasttext.cc/docs/en/pretrained-vectors.html

[7] http://data.buenosaires.gob.ar/dataset/calles

Table 4 Regular expressions for detecting named entities

Regular Expression	Named Entity
[0–9]+	Number
\d +?(([:.]?\d +?(AM|PM|HS)?)(\d)*?(am|pm|hs|horas))	Time Expression
[Rr][(Pp)|(Nn)]?\d+	Provincial or National Route
[!"#$%&'()*+,-./:;<=>?@[\]^_'{|}~]	Punctuation

of the threshold, while the vertical axis shows the result values for the metrics employed. The graph shows that the f-measure reaches a peak at the threshold 0.875 and then starts to slightly decline when the threshold value is higher than 0.9. From this experiment, we concluded that the optimal threshold value for the NER is 0.875. By setting this threshold value, the NER will achieve high precision without decreasing recall for identifying traffic incidents in Buenos Aires city.

4.2.2 Evaluating the Detection of Traffic Incidents

With the best combination of classifiers for the classification stage and the best threshold value for the NER step, we proceeded to assess the ability of the approach to identify traffic incidents. The approach was tested with a sample of Twitter publications to determine its success rate in the detection of traffic incidents. Besides, the experiment was aimed to verify whether the approach detected false incidents (i.e. traffic incidents detected in non traffic-related publications).

The NER step as described previously was enhanced with a catalogue of special terms (such as the commonest articles, prepositions, pronouns, and connectors) extracted from the RAE Spanish term list.[8] Particularly, 345 syntactic rules were defined to be used in the pattern matching process. These syntactic rules were abstracted from the manual analysis of the traffic-related publications from the random fold used in the second experiment.

The input for the approach in this experiment was another random fold different to the one used for the NER setup. The selected fold contains considerably more non traffic-related publications (355 publications, representing the 58.58%) than traffic-related publications (251 publications, representing the 41.42%). The identification of traffic-related publications stage resulted in 287 publications classified as traffic-related (245 TP and 42 FP). Those 287 publications were the input for the second stage of the approach. Once these publications were analyzed in the second stage, the approach detected traffic incidents in 207 publications and discarded the remaining 80 publications. Among these 80 publications, there were 42 non traffic-related publications that have reached this stage and 38 traffic-related publications that were wrongly analyzed. At this point, the approach correctly analyzed 207 out of the 245 traffic-related publications. This means that the approach could interpret the 84.48% of the publications received in the second stage.

To summarize, the whole approach correctly identified traffic incidents from 207 traffic-related publications of the total of 251 (82.47%). The remaining 44 publications were incorrectly identified for several reasons: (a) the classification stage incorrectly discarded 6 publications; (b) the approach was not able to detect the traffic incident in 2 publications that presented an incorrect syntactic structure; (c) the NER step could not recognize some named entities in 7 publications, so the approach could not detect the necessary traffic incident components in these publications; and (d) the pattern matching process could not interpret the remaining 29 publications given the absence of necessary syntactic rules. Regarding the 355 non traffic-related publications, the approach discarded 313 in the classification stage and it analyzed the remaining 42. However, the approach discarded these 42 publications since it was not able to identify the necessary traffic incident components.

The 29 publications that could not be interpreted due to the absence of necessary syntactic rules were analyzed to

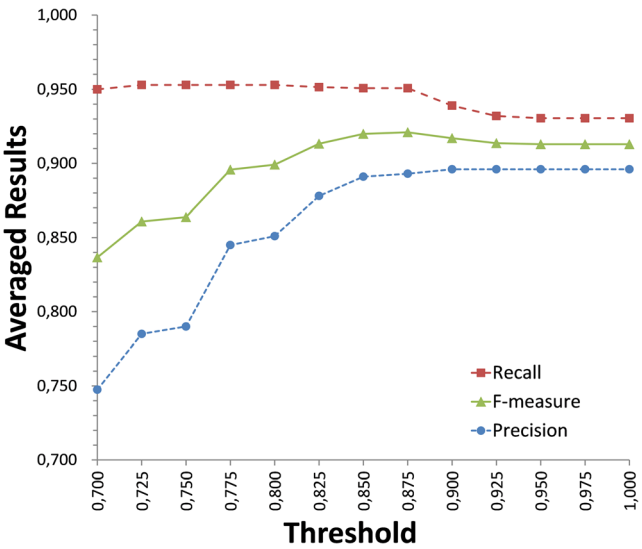

Fig. 3 Experimental results for the assessment of the best threshold value to be used in the Approximate String Matching technique

8 http://www.rae.es/diccionario-panhispanico-de-dudas/terminos-linguisticos

incorporate new rules to the approach. With this set of new rules, the second stage improved the identification of traffic incidents from 84.48% to 96.33%. By considering the full process of the approach, the identification of traffic incidents improved from 82.47% to 94.02% (from the 251 traffic-related publications, the approach correctly classify and identified 236). Adding new rules allow the approach to increase the approach to increase the amount and quality of knowledge available to detect and interpret traffic incidents, gradually approaching the optimal knowledge.

5 Approach in Action

All the previous experiments measured the effectiveness of the approach. The results achieved showed that the approach provides a valuable source of real-time traffic information. To present our approach as a whole materialized solution, we developed a tool named Manwë (Section 5.1). This tool filters, analyzes and reports in real time traffic-related incidents detected in Twitter's public stream restricted to Buenos Aires city. Then, we compared Manwë with Waze (Section 5.2) in order to validate our approach with respect to a popular traffic-specific social network.

5.1 Manwë

Manwë was developed in Java and configured to detect traffic incidents in Buenos Aires city. Figure 4 shows a diagram with the main components of Manwë, the external libraries used, and the initial configuration that has to be provided by an analyst or administrator of the system. From top to bottom, the first component of Manwë is the *TwitterStreamer* component. This component is responsible for establishing a connection with the streaming API of Twitter and receiving recent publications. This component relies on Twitter4J,[9] a Java library for the Twitter API. In order to connect with Twitter API, the *TwitterStreamer* needs the required access tokens. This information is defined by the analyst in the twitter.cfg configuration file. Additionally, in this file the analyst also defines the query parameters accepted by the Twitter API[10] to filter which publications will be part of the streaming: a list of Twitter users, a list of keywords, geographical boundaries, and the language of interest.

When the *TwitterStreamer* receives a new publication, Manwë starts processing it. As we mentioned in Section 3.1, the first step is normalize the text in the publication. This task is carried out by the *PreProcessor* component. This component splits CamelCase strings, lowercases the text, deletes repeated letters, and removes accent marks, hashtags symbols (#), references to other users, links/URLs and emails. As we explained in Section 3, this preprocessed text is used as input for the two main stages of the approach: *the identification of traffic-related publications* and the *detection of traffic incidents*.

The first stage uses supervised machine learning techniques to filter non traffic-related publications. First, the *TextCleaner* component deletes some elements of the text that may negatively affects the behavior of the ML techniques, as we mentioned in Section 3. Particularly, this component deletes stop-words, punctuation marks, and replaces numbers with the tag [number-L] where L is the length of the number (e.g. the number 1030 is replaced by [number-4]). Since different languages have different stopwords, the analyst can define the list of stopwords to be removed in the *stopword.dic* file according to the language in which the tweets that Manwë will analyze are written. The resultant text is used as input to the different ML classifiers. In this implementation, Manwë uses the three classifiers selected in Section 4.1.1 (SVM + DMNB + IBk). The classification models are generated by the *Trainer* component during the initial configuration of the system. This component relies on Weka,[11] which provides a Java library to train classification models and also to use them. The *Trainer* component trains the classification models from the *tweets.arff* file, which contains a set of publications earlier labeled by the analyst in two categories: traffic-related and non traffic-related. Note that the analyst can incorporate new labeled publications to the *tweets.arff* in any moment, in order to generate more accurate classification models.

For each received publication, each of the three classification models predicts a category. The predicted categories are taken by the *DecisionMaker* component to determine whether or not to continue with the analysis of the given publication. If none of the classifiers categorized the publication as traffic-related, then the publication is considered as non traffic-related and it does not reach the second stage. Although these publications could simply be discarded, our current implementation stores them in a repository (a PostgreSQL[12] database). Thus, the analyst may gradually review these publications looking for misclassifications. Misclassified publications can then be added by the analyst to the *tweet.arff* file in order to enrich the classification models, as was mentioned before. The rest of the publications, which were classified as traffic-related by at least one classifier, reach the second stage of the approach for further analysis.

9 http://twitter4j.org/en/index.html

10 https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/basic-stream-parameters

11 https://www.cs.waikato.ac.nz/ml/weka/

12 https://www.postgresql.org/

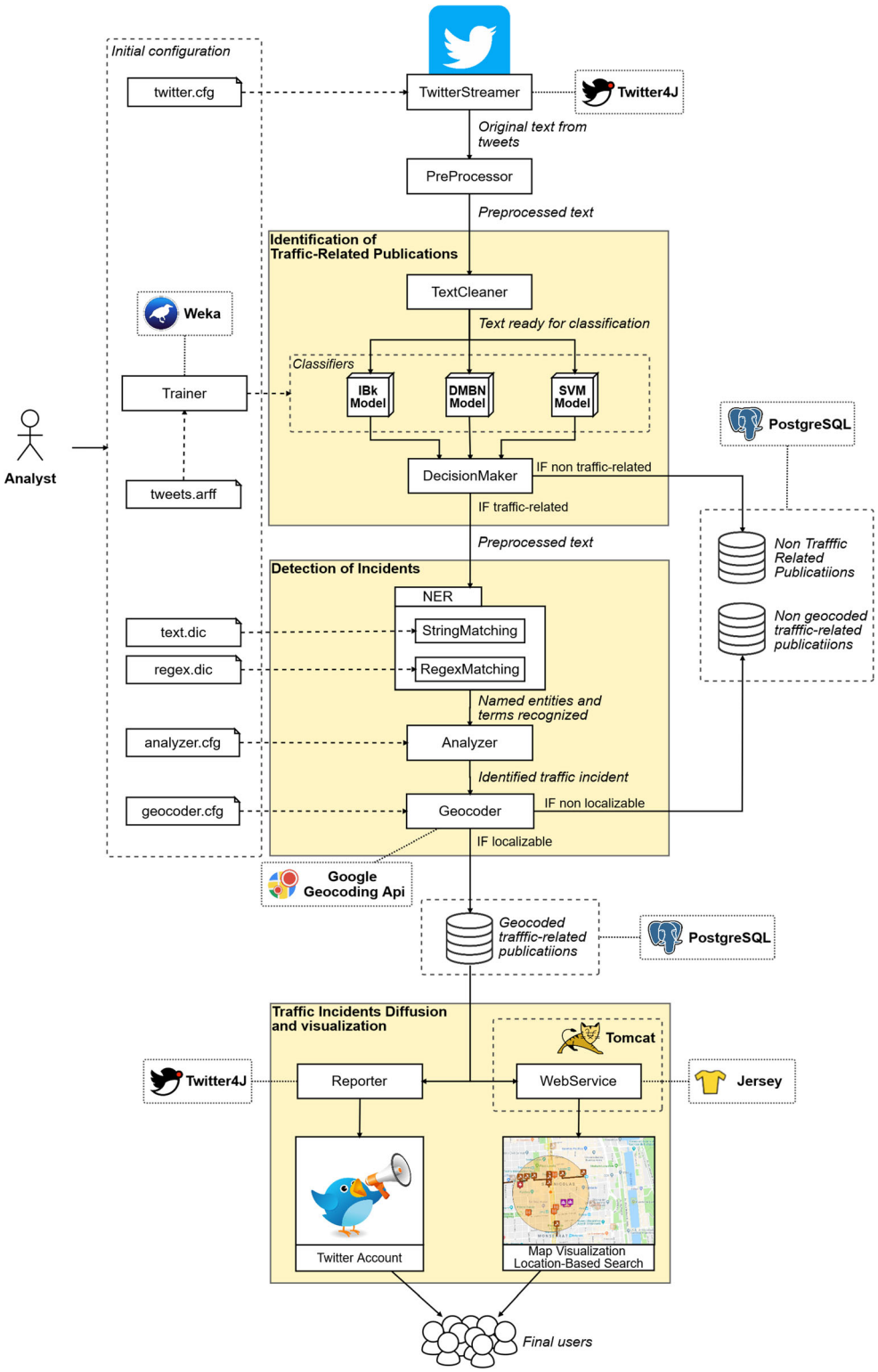

Fig. 4 The implementation diagram of Manwë

The second stage uses natural language processing techniques to detect any traffic incident reported in the publication. First, in the NER step, the approach recognizes named entities and special terms by using regular expressions and Approximate String Matching (as was mentioned in Section 3.1.1). On the one hand, the *RegexMatching* component uses regular expressions defined by the analyst in the regex.dic in order to detect entities that follow a pattern (such

as a date). To achieve this goal, this component relies on the implementation of regular expressions already provided by Java.[13] On the other hand, the *StringMatching* component looks for text fragments in the publication text that match some entity or special term defined by the analyst in the text.dic. To carry out this task, firstly Manwë splits the preprocessed text of a publication using white spaces and punctuation marks as delimiters, and generates a list of all the possible text fragments or n-grams. Secondly, for each text fragment Manwë uses a q-gram algorithm (similar to the one proposed in (Yang et al., 2010)) to retrieve the known entities and special terms that may potentially match that text fragment. Then, Manwë calculates the Levenshtein distance between the entities and special terms selected in the previous step and the given piece of text. Finally, the Levenshtein distance is used to calculate the similarity score between the text fragment and the entity or special term (as was mentioned in Section 3.3.1). Note that the analyst may gradually add new entities or special terms just by modifying the regex.dic or the text.dic.

The entities and special terms recognized in the text are used by the *Analyzer* component to detect traffic incidents. This component iteratively applies the syntactic rules defined by the analyst in the analyzer.cfg configuration file. By applying these rules, Manwë incrementally compound the entities and terms recognized into locations, and then into traffic incidents (as was mentioned in Section 3.3.2). The capability of Manwë to identify traffic incidents will depend on the set of syntactic rules defined by the analyst. If the analyst identifies that a traffic incident was not detected due to the lack of the necessary syntactic rule, she/he can add the new rules needed just by modifying the analyzer.cfg configuration file. This allows Manwë to analyze tweets written in different languages and with different phrase structures. For example, an address in Spanish starts with the street name followed by the address number (for example Av. Avellaneda 1727); instead, in English address are referred to with a number followed by the street name(for example 1253 Jackson St). These differences can be easily addressed by defining the correct syntactic rule according to the language spoken on the region where Manwë is planned to be used.

In the last step of the second stage, the *Geocoder* component tries to obtain the geographic coordinates where the incident took place. This component uses the location identified by the *Analyzer* component (for example, Av. Avellaneda 1727), and sends a geocoding request to the Google Geocoding API.[14] The geocoding request is enriched with information about the region that is being monitored by Manwë. This information is defined by the analyst in the geocoder.cfg configuration file. In this file the analyst can define the API key uses to access the Geocoding API, and the names of the city and country being monitored by Manwë. For example, if the city being monitored is Buenos Aires, then the geocoding request asks for 'Av. Avellaneda 1727, Buenos Aires, Argentina'. When the Geocoding API correctly returns the geographic coordinates of the location, the publication and the geocoded traffic incident are stored in a repository for later diffusion. When the Geocoding API gives no results or when the location identified by the Analyzer is no geolocalizable (for example, "Delays in Av. Avellaneda" provides only the name of a street and cannot be translated to precise geographic coordinates), then the publication is stored in another auxiliary repository. The analyst can manually review these publications later in order to improve Manwë. For example, if the incident was not detected due to the lack of a named entity, the analyst can add the missing entity to the regex.dic or the text.dic. Instead, if the incident was incorrectly identified due to an error in the syntactic rules, the analyst can just fix the error by modifying the analyzer.cfg configuration file. Thus, Manwë will gradually improve the named entity recognition and contemplate new different phrase structures.

In order to disseminate the information about the detected traffic incidents, we created a Twitter account[15] and an interactive map.[16] Each time a new traffic incident is detected, the *Publisher* component posts about the incident in the dedicated Twitter account. It only tweets the first time that an incident is detected and the corresponding source (i.e. the Twitter account who published the original tweet). Furthermore, a *WebService* component (developed with Jersey[17] and deployed in a Tomcat[18] container) provides the information about traffic incidents to any third party application that wishes to access this information. For example, we developed an interactive map that consumes this web service and shows the detected incidents with different marker types containing useful information about the incident: (a) the time of the first tweet reporting the incident; (b) the estimated finishing time (every traffic incident type has associated an estimated duration time according to its incident type, for example, the time assigned to a car crash is lower than the time assigned to a road closure for maintenance); (c) the list of different tweets that reported that incident. From our point of view, the interactive map is a powerful tool to help users planning their movements across the city. Moreover, an interactive map it would enable the possibility of collecting feedback from users. For example, users could report a traffic incident incorrectly detected or incorrectly geocoded. This feedback could be used by the

13 https://docs.oracle.com/javase/7/docs/api/index.html

14 https://developers.google.com/maps/documentation/geocoding/start

15 https://twitter.com/Manwebsas

16 http://si.isistan.unicen.edu.ar/Manwe/research-demo/

17 https://eclipse-ee4j.github.io/jersey/

18 http://tomcat.apache.org/

analyst to make the needed corrections in Manwë (as those mentioned in the previous paragraphs).

5.2 Comparing Manwë with Waze

To evaluate the utility of our approach in a real case study, we compared Manwë with Waze. Waze is a traffic-specific social network with a community-driven GPS navigation app owned by Google. Waze users, known as *wazers*, can report different types of traffic incidents (such as car crashes, traffic jams, speed cameras, and police traps). Furthermore, the users' speed and location, along with other information, is sent to Waze's database to improve the service as a whole. There are two main differences between Manwë and Waze. First, Waze is fed by the closed community of users of the application, while Manwë is fed by the traffic-related reports collected from a general-purpose social network. Second, wazers report an incident at the moment in which they are witnesses of it, therefore, the GPS position of the user is used as the location of the reported incident. On the contrary, Manwë does not require that the user reports the incident in situ, since it geolocates the incident from the information that can be derived from the text of the publication.

In this section, we compare the incidents detected by Manwë with respect to the incidents reported in Waze. To do this, we recorded the traffic incidents detected by Manwë and Waze during a period of 5 days (including working and non-working days). The area of the experiment was limited to incidents within the City of Buenos Aires, Argentina (CABA) (specifically, between the maximum latitude of −34.5328; minimum latitude of −34.7075; maximum length of −58.3031; minimum length of −58.5324). We limited our analysis to two specific types of incidents: car crashes and road closures. During the period of the experiment, Manwë detected 303 unique incidents, while Waze detected 706 (second column of Table 5). After analyzing the detected incidents, we noticed that there were several issues that worth being highlighted. The first issue regards the reputation or confidence of the reports generated by the users. For Manwë, we decided to discard all the incidents with a single report (a single tweet), unless the tweet has been published by a trusted account (such as an official government account). Similarly, in Waze each incident has a 'confidence' level according to the number of reports and the confidence on the reporters. Those reports with a level greater than 0 were kept for the experiment (third column of Table 5).

The second issue regards redundant traffic incidents. We noticed some redundancy due to the geospatial proximity of the detected incidents. We believe that this redundancy is generated by the fact that the same incident is detected/reported multiple times by users and pinned in the map as different incidents. While this situation occurred in both approaches, it became more evident in the case of Waze, since different users reported incidents in nearby locations (each using their GPS position). It is worth noticing that wazers receive a score for reporting incidents, and the score received by a user for reporting a new incident in Waze is greater than the score received by confirming an existing incident. Thus, wazers may prefer to report a new incident rather than confirming an incident reported by other wazers. Figure 5 shows an example of the redundancy found during the experiment. The example is a snapshot of a region of the map for a specific day in a period of 4 h between the first and the last reported incidents.

To remove this redundancy, we clustered all the incidents within a given radius R and within a time interval T, as a single incident. The radius R was set at 100 mts since that is the approximate length of a block in CABA. Therefore, considering a higher radius derives in the risk of clustering different incidents as one. The time interval T varies depending on the type of incident. For car crashes, we set T to 4 h, while for road closures we set T to 24 hs. In this way, two incidents are considered the same incident if the distance between them is lower than 100 mts and their reporting time is within the time interval T. The fourth column of Table 5 shows the results achieved with a filter of 100 mts. Under these conditions, we discarded 30.03% of the incidents detected by Manwë and 59.06% of the incidents reported in Waze.

The final number of incidents detected by both approaches was of 501 (212 by Manwë and 289 by Waze). To analyze both approaches, we decided to measure the overlap between them. At this point, we identified 27 incidents that were detected by Manwë and also reported in Waze. By considering

Table 5 Incidents detected and filtered

Approach	Incidents detected	Incidents with enough confidence	Incidents clustering (100mts)	Filtered incidents (%)
Manwë	303	244	212	30,03%
Waze	706	463	289	59,06%

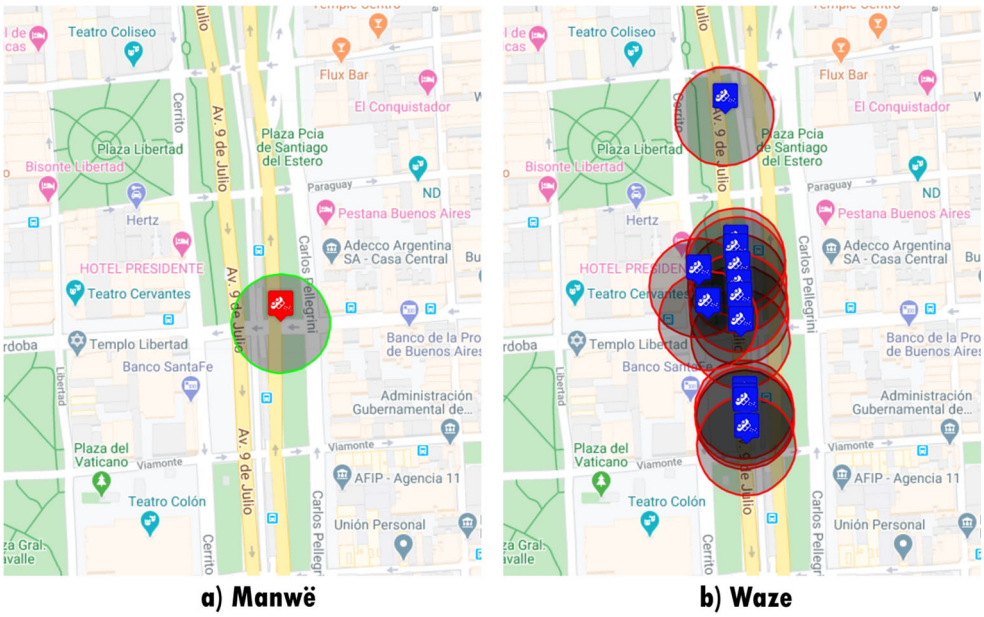

Fig. 5 Redundant traffic incidents

this overlap, the number of unique incidents detected by both approaches was of 474. Figure 6 shows the percentage of unique incidents detected by each approach.

To sum up, we observed that less than 6% of the total number of incidents is detected by both Manwë and Waze. This fact led us to conclude that Manwë and Waze are complementary approaches and that by using them together we would be able to cover a wider range of incidents. Furthermore, although Waze informed a higher number of incidents, even after the filtering process, we observed that there was still redundancy in the informed incidents, especially in highways. For example, Fig. 7 shows that Waze informs 6 different incidents in a highway, while Manwë only detected one incident in the same period of time. This led us to believe that the six reports in Waze correspond to the same incident, but reported by users in different positions of the highway (probably due to the bottleneck generated by the incident). Although the 100 m radius can be considered enough for detecting redundant incidents inside a city, it is not enough for highways. In this sense, future comparisons should differentiate between streets and highways.

6 Conclusions

This article presented and materialized a novel and complete approach for traffic incident detection from Twitter publications and its dissemination in social networks. In particular, the approach uses ML techniques to filter the publications that are relevant to traffic incidents. Then, the approach analyzes the text of each traffic-related publication to extract traffic incident components such as the incident type and the location. After that, if

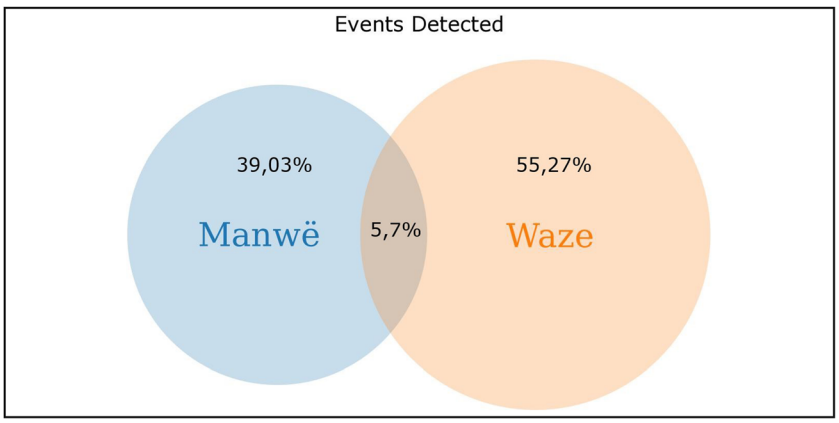

Fig. 6 Percentage of incidents detected by Manwë and Waze

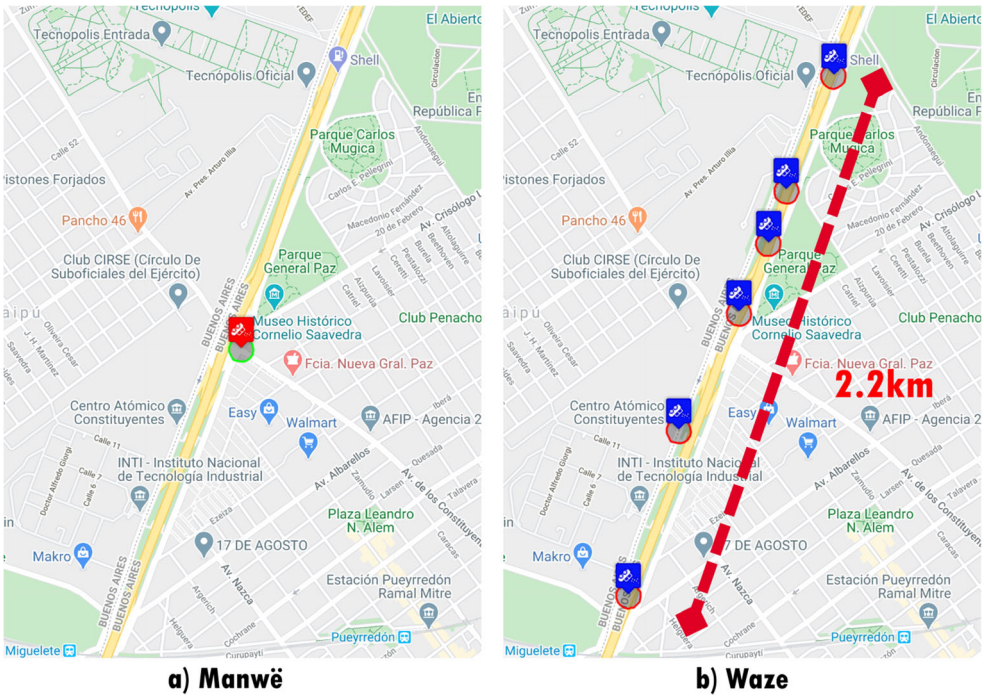

Fig. 7 Incidents detected by Manwë and Waze in the highway 'Av. General Paz'

the required components are found for a given publication, the approach groups the collected information and geolocates the traffic incident. At this point, an incident report is generated using the retrieved information and the geolocation of the incident. Finally, every incident report is conveyed via centralized mediums such as an interactive online map and a dedicated Twitter account. The map lets users know whether any traffic incident is currently affecting the area near them or a travel path from a start-point to an end-point.

The main contribution of this work is an approach that automatically detects, groups and geolocates traffic incidents reported in natural language on social network publications by using a mixing of both machine learning and natural language processing techniques. As social network publications generally present informal writing, such as misspelled named entities or informal syntactic structures, the proposed approach uses several NLP techniques to mitigate the informal writing and correctly interpret the traffic incidents reported. This approach processes multiple publications without human intervention, allowing a region of interest to be monitored in real time. As a result, our approach transforms information from a passive source (in which users need to search and evaluate if any incident will affect his/her path) to an active source (in which the required information can be actively provided to the user, e.g., using the GPS sensor of the user's mobile device).

The experiments in this work showed encouraging results for both stages of our approach (Section 4.1 and Section 4.2). Moreover, the comparison of our materialized approach with Waze (Section 5.2) validated the contribution of our approach by detecting traffic incidents not reported in Waze. However, we suggest that further research should be undertaken in the following research lines. The impact of processing publications that report traffic incidents should be analyzed considering different regions to study the robustness of our approach. In particular, each region has both its own entities and its own dialect, which can affect the accuracy of the text classifiers or that might require different syntactic rules. Therefore, it would be interesting to generate a knowledge abstraction to reduce the effort required to instantiate the approach in a new region. Another research line could be the development of a complementary module aiming to automatically learn unknown named entities. For instance, if a syntactic rule cannot be applied because one of the text fragments involved is not recognized as a named entity, the module could query external sources to determine if the text fragment is actually a valid named entity. If so, the module would update the named entity catalogue. It is important to note that we are currently developing an online platform to provide information about traffic incidents to third-party applications. In this way, external applications could query the traffic incidents in a specific region of interest, and then use this information to assist end users. It would also be interesting to take into account the user feedback. This feedback could be obtained from the users' interactions with the tweets broadcasted by our approach. For example, if a user likes a tweet published by Manwë, it could be interpreted that the incident was correctly detected. Moreover,

the interactive map could also offer users the possibility of confirming the existence of an incident or reporting errors in the detection. This feedback could be used by the analyst to improve the approach by retraining the classifiers, defining new syntactic rules or adding new entities to the dictionaries.

References

Albuquerque, F. C., Casanova, M. A., Lopes, H., Redlich, L. R., de Macedo, J. A. F., Lemos, M., de Carvalho, M. T. M., & Renso, C. (2016). A methodology for traffic-related twitter messages interpretation. *Computers in Industry, 78*(May), 57–69.

Bothorel, Cécile, Neal Lathia, Romain Picot-Clemente, and Anastasios Noulas. 2018. "Location recommendation with social media data." In *Lecture Notes in Computer Science*, 624–653.

Carvalho. (2010). Real-Time Sensing of Traffic Information in Twitter Messages. In *Edited by Rosaldo Rossetti*. Master: Universidade do Porto.

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*(4), 2269–2283.

Dabiri, Sina, and Kevin Heaslip. 2019. "Developing a twitter-based traffic event detection model using deep learning architectures." *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.2018.10.017.

Endarnoto, Sri Krisna, Sonny Pradipta, Anto Satriyo Nugroho, and James Purnama. 2011. "Traffic Condition Information Extraction & Visualization from social media twitter for android Mobile application." In *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, 1–4.

Ertekin, Seyda, Jian Huang, Leon Bottou, and Lee Giles. 2007. "Learning on the Border: Active Learning in Imbalanced Data Classification." In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, 127–36. CIKM '07. New York, NY, USA: ACM.

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*(June), 321–342.

Kamran, Shoaib, and Olivier Haas. 2007. "A Multilevel Traffic Incidents Detection Approach: Identifying Traffic Patterns and Vehicle Behaviours Using Real-Time GPS Data." In *Intelligent Vehicles Symposium, 2007 IEEE*, 912–17. IEEE.

Köknar-Tezel, Suzan, and Longin Jan Latecki. 2009. "Improving SVM classification on imbalanced data sets in distance spaces." In *2009 Ninth IEEE International Conference on Data Mining*, 259–267.

Kozareva, Zornitsa, Boyan Bonev, and Andres Montoyo. 2005. "Self-Training and Co-Training Applied to Spanish Named Entity Recognition." In *Lecture Notes in Computer Science*, 770–79.

Kuflik, T., Minkov, E., Nocera, S., Grant-Muller, S., Gal-Tzur, A., & Shoor, I. (2017). Automating a framework to extract and analyse transport related social media content: The potential and the challenges. *Transportation Research Part C: Emerging Technologies, 77*(April), 275–291.

Levenshtein, V. 1996. "Lower Bounds on Cross-Correlation of Codes." In *Proceedings of ISSSTA'95 International Symposium on Spread Spectrum Techniques and Applications*, 2:657–61 vol.2.

Mandal, A. K., & Sen, R. (2014). Supervised learning methods for Bangla web document categorization. *International Journal of Artificial Intelligence & Applications, 5*(5), 93–105.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Manning, C., M. Surdeanu, J. Bauer, and J. Finkel. 2014. "The Stanford CoreNLP Natural Language Processing Toolkit." *Proceedings of 52nd*. http://www.aclweb.org/anthology/P14-5010.

Pereira, João, Arian Pasquali, Pedro Saleiro, and Rosaldo Rossetti. 2017. "Transportation in social media: An automatic classifier for travel-related tweets." *Progress in Artificial Intelligence*https://doi.org/10.1007/978-3-319-65340-2_30.

Ritter, Alan, Sam Clark, Mausam, and Oren Etzioni. 2011. "Named Entity Recognition in Tweets: An Experimental Study." In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1524–34. EMNLP '11. Stroudsburg, PA, USA: Association for Computational Linguistics.

Schneider, Karl-Michael. 2005. "Techniques for improving the performance of naive Bayes for text classification." In *Lecture Notes in Computer Science*, 682–93.

Schulz, Axel, Petar Ristoski, and Heiko Paulheim. 2013. "I See a Car Crash: Real-Time Detection of Small Scale Incidents in Microblogs: 25th International Conference, CAiSE 2013, Valencia, Spain, June 17–21, 2013. Proceedings." In *Advanced Information Systems Engineering*, edited by Camille Salinesi, Moira C. Norrie, and Óscar Pastor, 7908:22–33. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg.

Sumalee, Agachai, and Hung Wai Ho. 2018. "Smarter and more connected: Future intelligent transportation system." *IATSS Research*. https://doi.org/10.1016/j.iatssr.2018.05.005.

Wanichayapong, Napong, Wasawat Pruthipunyaskul, Wasan Pattara-Atikom, and Pimwadee Chaovalit. 2011. "Social-Based Traffic Information Extraction and Classification." In *2011 11th International Conference on ITS Telecommunications*, 107–12.

Yang, Zhenglu, Jianjun Yu, and Masaru Kitsuregawa. 2010. "Fast algorithms for top-K approximate string matching." In *AAAI*. Vol. 1463. http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/viewFile/1939/2234.

Zhou, Xiaokang, and Qun Jin. 2011. "Dynamical User Networking and Profiling Based on Activity Streams for Enhanced Social Learning." *Advances in Web-Based Learning - ICWL 2011*. https://doi.org/10.1007/978-3-642-25813-8_23.

Sebastián Vallejos is a PhD student at UNICEN University, Argentina, with a research grant from CONICET (National Council for Scientific and Technological Research). He received a System Engineer degree in 2016 from the UNICEN. His research interests include natural language processing, social network analysis, and object-oriented programming. Contact him at sebastian.vallejos@isistan.unicen.edu.ar.

Diego Alonso is a PhD student at UNICEN University, Argentina, with a research grant from CONICET (National Council for Scientific and Technological Research). He received a System Engineer degree in 2016 from the UNICEN. His research interests include natural language processing, social network analysis, pattern recognition and deep learning. Contact him at diego.alonso@isistan.unicen.edu.ar.

Brian Caimmi is a PhD student at UNICEN University, Argentina. He received a System Engineer degree in 2016 from the UNICEN. His research interests include natural language processing, software design social network analysis, and object-oriented programming. Contact him at brian.caimmi@isistan.unicen.edu.ar.

Luis S. Berdun is a researcher at ISISTAN Research Institute (CONICET-UNICEN), Argentina, and a professor at UNICEN University, Argentina. He received a PhD degree in Computer Science in 2009 and a Master in Systems Engineering in 2005 from the UNICEN. His research interests include intelligent aided software engineering, planning algorithms, knowledge management. Contact him at luis.berdun@isistan.unicen.edu.ar.

Marcelo G. Armentano is a researcher at ISISTAN Research Institute (CONICET-UNICEN), Argentina, and a professor at UNICEN University, Argentina. He received a PhD degree in Computer Science in 2008 and a Master in Systems Engineering in 2006 from the UNICEN. His research interests include social network analysis, user modeling, and recommender systems. Contact him at marcelo.armentano@isistan.unicen.edu.ar.

Álvaro Soria is a researcher at ISISTAN Research Institute (CONICET-UNICEN), Argentina, and a professor at UNICEN University, Argentina. He received a PhD degree in Computer Science in 2009. His research interests include software architectures, quality-driven design, object-oriented frameworks, and fault localization. Contact him at alvaro.soria@isistan.unicen.edu.ar.