Noname manuscript No.
(will be inserted by the editor)

CrisMap: A Big Data Crisis Mapping System based on Damage Detection and Geoparsing

Marco Avvenuti · Stefano Cresci · Fabio Del Vigna · Tiziano Fagni · Maurizio Tesconi

the date of receipt and acceptance should be inserted later

Abstract Natural disasters, as well as human-made disasters, can have a deep impact on wide geographic areas, and emergency responders can benefit from the early estimation of emergency consequences. This work presents `CrisMap`, a Big Data crisis mapping system capable of quickly collecting and analyzing social media data. `CrisMap` extracts potential crisis-related actionable information from tweets by adopting a classification technique based on word embeddings and by exploiting a combination of readily-available semantic annotators to geoparse tweets. The enriched tweets are then visualized in customizable, Web-based dashboards, also leveraging ad-hoc quantitative visualizations like choropleth maps. The maps produced by our system help to estimate the impact of the emergency in its early phases, to identify areas that have been severely struck, and to acquire a greater situational awareness. We extensively benchmark the performance of our system on two Italian natural disasters by validating our maps against authoritative data. Finally, we perform a qualitative case-study on a recent devastating earthquake occurred in Central Italy.

Marco Avvenuti
University of Pisa, Department of Information Engineering
E-mail: marco.avvenuti@unipi.it

Stefano Cresci (✉)
National Research Council (CNR), Institute of Informatics and Telematics (IIT) – via G. Moruzzi 1, 56124 Italy
Tel.: +39 050 315 8272, Fax: +39 050 315 2593
E-mail: stefano.cresci@iit.cnr.it

Fabio Del Vigna · Tiziano Fagni · Maurizio Tesconi
National Research Council (CNR), Institute of Informatics and Telematics (IIT)
E-mail: [fabio.delvigna,tiziano.fagni,maurizio.tesconi]@iit.cnr.it

Keywords Crisis mapping · word embeddings · geoparsing · online social networks · social media · big data

1 Introduction

Computational solutions capable of overcoming societal sustainability challenges have always been looked at with great interest from both Academia and practitioners [**Wang2014**]. In recent years, one of the fields that has attracted more attention is the one related to the exploitation of user-generated information for disaster management [**Avvenuti2016framework**]. Within this context, Social Media (SM) data revealed to be particularly valuable in the aftermath of those events, typically natural and human-made disasters, which trigger massive participation of affected communities in sharing time-sensitive and actionable information [**Gao2011**, **Avvenuti2016predictability**]. Both types of disasters require a timely intervention by emergency responders, who are in charge of providing support and relief to the affected population. In many practical situations, the scarcity of key resources – temporal, economic, and human resources above all – imposes dire limitations to the extent and the effectiveness of the emergency management process. For this reason, tools capable of supporting resource allocation and prioritization can have a significant impact towards the effectiveness of emergency management operations. Among these tools, there are the SM-based crisis mapping systems, which increase situational awareness by enabling the real-time gathering and visualization of data contributed by many SM users. Such type of system is a platform able to collect text and multimedia content from a variety of sources, such as Twitter and Facebook, to analyze and aggregate collected data, and to visualize relevant facts on a map. Notably, during many recent disasters, civil protection agencies developed and maintained live Web-based crisis maps to help visualize and track stricken locations, assess damage, and coordinate rescue efforts [**Middleton2014**].

Indeed, recent work demonstrated the possibility to create crisis maps solely using geolocated data from SM, to understand better and monitor the unfolding consequences of disasters [**Goolsby2010**, **Middleton2014**, **Avvenuti2016impromptu**]. All these SM-based crisis mapping systems face the fundamental challenge of *geoparsing* the textual content of emergency reports to extract mentions of places/locations, thus increasing the number of messages to exploit. Geoparsing involves binding a textual document to a likely geographic location which is mentioned in the document itself. State-of-the-art systems, such as [**Middleton2014**], perform the geoparsing task by resorting to a number of preloaded geographic resources containing all the possible matches between a set of place names (toponyms) and their geographic coordinates. This approach requires an offline phase where the system is specifically set to work in a geographically-limited region. Indeed, it would be practically infeasible to load associations between toponyms and coordinates for a vast area or a whole country. Moreover, not all geolocated data is useful towards understanding the severity of the emergency and, indeed, only a small fraction of messages convey information about the consequences of the emergency on communities and infrastructures. Current crisis mapping systems typically detect the most stricken areas by considering the number of messages shared and by following the assumption that more emergency reports equal to more damage [**Weber2014**, **Middleton2014**]. Although this relation exists when considering densely and uniformly populated areas [**Liang2013**], it becomes gradually weaker when considering broader regions or rural areas. These challenges, related to the detection of damage and geoparsing, are reflected by the current limitations of state-of-the-art SM-based crisis mapping systems [**Avvenuti2016impromptu**].

Here, we propose solutions to overcome the main drawbacks of current state-of-the-art crisis mapping systems. Our proposed system exploits both situational assessments and position information contained in textual data produced during an emergency. One interesting novelty of our approach is the analysis of emergency-related tweets from a twofold perspective: (i) a damage detection component exploits word embeddings and a SVM classifier to detect messages reporting damage to infrastructures or injuries to the population; (ii) a message geolocation component performs the geoparsing task by exploiting online semantic annotation tools and collaborative knowledge-bases such as Wikipedia and DBpedia. Information extracted by the damage detection and the message geolocation components are combined to produce interactive, Web-based crisis maps.

Contributions. We describe `CrisMap`: a system capable of producing crisis maps in the aftermath of mass emergencies by simultaneously adopting word embeddings for tweet filtering and classification, and by exploiting semantic annotators for tweet geoparsing. In particular:

– we propose an architectural solution for crisis mapping, based on scalable and resilient Big Data technologies;
– we address the problem of damage detection in SM messages;
– we compare a damage detection approach based on natural language processing (NLP) versus one based on word embeddings (WE), highlighting the advantages of WE towards language independence and fast processing;
– we propose and benchmark a geoparsing technique based on readily-available semantic annotators;
– we validate the reliability of the crisis maps generated by our system against authoritative data for two past emergencies;
– we perform a qualitative case-study on a recent severe earthquake in Italy.

The deployment of our proposed `CrisMap` system can help to quickly map damage scenarios to concentrate rescue efforts and organize a prompt emergency response.

2 Related Works

The possibility to exploit social media data for crisis mapping has been first envisioned in a few trailblazing works [**Goolsby2010**, **Gao2011**, **Meier2012**] and further backed up by recent research [**Avvenuti2016predictability**]. Since the early works, there has been a growing interest by both practitioners and scholars in all areas related to crisis mapping: from data acquisition and management to analysis and visualization [**Avvenuti2016impromptu**].

2.1 Practical experiences

The great interest of practitioners and emergency responders towards the exploitation of SM data is testified by the efforts of the Federal Emergency Management Agency (FEMA) and the United States Geological Survey (USGS), which already led to interesting results with practical applications to earthquake emergency management[1] [**Burks2014**, **Earle2012**]. Regarding already deployed applications, well-known crisis mapping platforms are Ushahidi[2], Mapbox[3], Google's Crisis Map[4], ESRI ArcGIS[5], and CrisisCommons[6] [**Bauduy2010**]. The main features of these platforms are related to data acquisition, data fusion, and data visualization. Such platforms represent hybrid crowdsensing systems where users can voluntarily load data onto the system in a participatory way, or the system can be configured to automatically perform data acquisition opportunistically. The same hybrid data collection strategy has also been employed in a fully automatic system recently benchmarked in the earthquake emergency management field [**Avvenuti2017**]. Another already-deployed application that exploits crowdsourced data is USGS's "Did You Feel It?" (DYFI) system[7]. This system, although not relying on SM data, exploits citizen reports and responses to earthquakes to automatically assess potential damage. It is foreseeable that in the near future such system could instead be fed with SM data. Indeed, there is already interesting research – from both USGS itself and other laboratories – moving towards this direction [**Guy2014**, **Avvenuti2016nowcasting**, **Kryvasheyeu2016**, **Kropivnitskaya2017**].

2.2 Academic works

Recent scientific literature has instead switched the focus from data acquisition and data fusion to in-depth data analysis. This is typically done by leveraging powerful machine learning techniques and resulted in novel solutions being proposed to overcome critical crisis mapping challenges such as geoparsing and extracting situational awareness from microtexts [**Cresci2015wise**].

Specifically, [**Middleton2014**] presents a state-of-the-art system that matches preloaded location data for areas at risk to geoparse real-time tweet data streams. The system has been tested with data collected in the aftermath of New York's flooding (US – 2012) and Oklahoma's tornado (US – 2013) and achieved promising results. Among the key features of [**Middleton2014**] is the possibility to match toponyms at region-, street-, or place-level. This step is achieved by preloading already existing geographic databases (e.g., the Geonames and GEOnet Names global gazetteers) for areas at risk, into the system. Crisis maps are then generated by comparing the volume of tweets that mention specific locations with a statistical baseline. Although presenting state-of-the-art solutions, [**Middleton2014**] still has several drawbacks. The system can only work on a specific geographical area at a time since it has to load and manage external data for that area. Tweets mentioning locations outside the predefined area cannot be geolocated and consequently disasters cannot be monitored outside the area's boundaries. Moreover, the width of the area covered by the system has direct implications on the amount of data to load and manage. This impacts on system's performances thus resulting in limitations on the maximum geographical area that can be monitored with [**Middleton2014**]. Furthermore, the system in [**Middleton2014**] does not take into account the problem of toponymic polysemy [**Cresci2015wise**, **Avvenuti2016impromptu**]. In addition, crisis maps generated by [**Middleton2014**] only consider tweet volumes and may result less accurate than those obtained by analyzing the content of tweets. For example in the case of severe earthquakes, where the shaking is also perceived hundreds of kilometers far from the epicenter, the majority of tweets comes from densely populated areas, such as big cities. Regardless, locations that have suffered most of the damage might be small villages in rural areas around the epicenter, which risk remaining unnoticed if the analysis only considers tweet volumes [**Cresci2015wise**, **Avvenuti2016impromptu**].

In addition to the fully functional crisis mapping system described above, other solutions for the geoparsing task have been recently proposed in [**Gelernter2011**, **Gelernter2013**, **Deoliveira2015**] where authors experimented with heuristics, open-source named entity recognition software, and machine learning techniques. Furthermore, other works emphasized the extraction of actionable and time-sensitive information from messages. For instance, authors of [**Verma2011**] apply natural language processing techniques to detect messages carrying relevant information for situational awareness during emergencies. In [**Imran2013**] is described a technique to extract "information nuggets" from tweets – that is, self-contained information items relevant to disaster response. While these works present fully

[1] `https://blog.twitter.com/2014/using-twitter-to-measure-earthquake-impact-in-almost-real-time`
[2] `https://www.ushahidi.com/`
[3] `https://www.mapbox.com/`
[4] `https://www.google.org/crisismap/`
[5] `http://www.esri.com/arcgis/`
[6] `https://crisiscommons.org/`
[7] `http://earthquake.usgs.gov/research/dyfi/`

automatic means to extract knowledge from texts, in [**Vieweg2014**] is proposed a hybrid approach exploiting both human and machine computation to classify messages. All these linguistic analysis techniques for the extraction of relevant information from disaster-related messages have however never been employed in a crisis mapping system.

Although these works are limited in scope to analyses of textual data, other efforts were recently devoted to the exploitation of multimedia content for emergency response. Aerial photographs and imagery have been widely adopted in monitoring tasks of areas involved in emergencies. This operation can be performed using vehicles, drones, and satellites [**Lewis2007**] and when such content is coupled with geographic and temporal information, it provides actionable information [**Dashti2014**]. Also images coming from SM can be exploited to train classifiers to recognize potential critical situations and get more knowledge about emergencies, as shown in [**Liang2013**, **Lagerstrom2016**]. However, messages related to emergencies could potentially contain false information and rumors, which may alter analyses [**Cheong2011Social**]. Similarly to textual data, also multimedia data can be faked, and such content can spread rapidly through Twitter [**Gupta2013**]. Fortunately, fake multimedia content is represents a minority of all multimedia content published in the aftermath of emergencies, and machine learning algorithms have been shown to be capable of characterizing bogus messages[**dewan2017towards**] in order to filter them out [**Gupta2013b**].

To conclude, we highlight that a recent survey presented an extensive review of current literature in the broad field of SM emergency management and can be considered for additional references [**Imran2015**].

3 System architecture

The software architecture of our crisis mapping system is presented in Figure **??**. As with any system that needs to cope with the massive amount of data collected from social networks, special requirements are imposed in the design by both the real-time constraints and the heterogeneity of data. For these reasons, our `CrisMap` system deploys several Big Data technologies to process incoming SM data efficiently, without sacrificing scalability and fault-tolerance.

The functional workflow is divided into three logical steps: (i) Data Ingestion and Enrichment (labelled with a blue circle), where SM data is collected and processed in order to select and geoparse messages containing information about damages; (ii) Data Indexing (labelled with a red circle), in which useful data is conditioned and saved into the internal storage; (iii) Data Visualization (labelled with a green circle), in which stored data is retrieved and used to create maps or, more generally, to provide results to the end users through a Web dashboard.

In the following, we give an overall description of the functionalities of each of these steps. A detailed technical description and evaluation of the solutions we adopted and implemented to address the main issues related to the design of a crisis mapping system, namely *mining messages to search for damage*, *geoparsing* and *visualize data*, are given in Sections **??**, **??** and **??**, respectively.

3.1 Data Ingestion and Enrichment

The first logical step of the system consists into ingesting data coming from available SM data sources, possibly enriching it with additional information not directly available from the data source and which can provide useful information exploitable subsequently during the visualization phase. Data ingestion occurs using platform-specific crawling/scraping software. For the sake of simplicity, in this work we focused our attention on Twitter solely. However, nothing prevents the adopter to deploy the system using a different data source.

Our system is able to process both real-time data fetched from the Twitter stream and historical data acquired from data resellers. In a practical application scenario, the system is fed with real-time data, while historical data can be used to run simulations on past emergency events. The *Crawler* component exploits Twitter's Streaming API[8] to perform data acquisition from the social network. The Streaming API gives low latency access to Twitter's global stream of messages. Collected messages can be optionally filtered using search keywords.

Collected messages are then forwarded to the Processing layer through a specific queue (named "Processing") placed at the Broker layer. Our implementation choice fell on Kafka[9], a distributed publish-subscribe messaging platform. As described on the website, Kafka "is used for building real-time data pipelines and streaming apps. It is horizontally scalable, fault-tolerant, wicked fast, and runs in production in thousands of companies". Its characteristics help the system to sustain very high traffic rates by mitigating the back-pressure data problem [**Backpressure**] and to process

[8] `https://developer.twitter.com/en/docs/tweets/filter-realtime/overview`

[9] `https://kafka.apache.org`

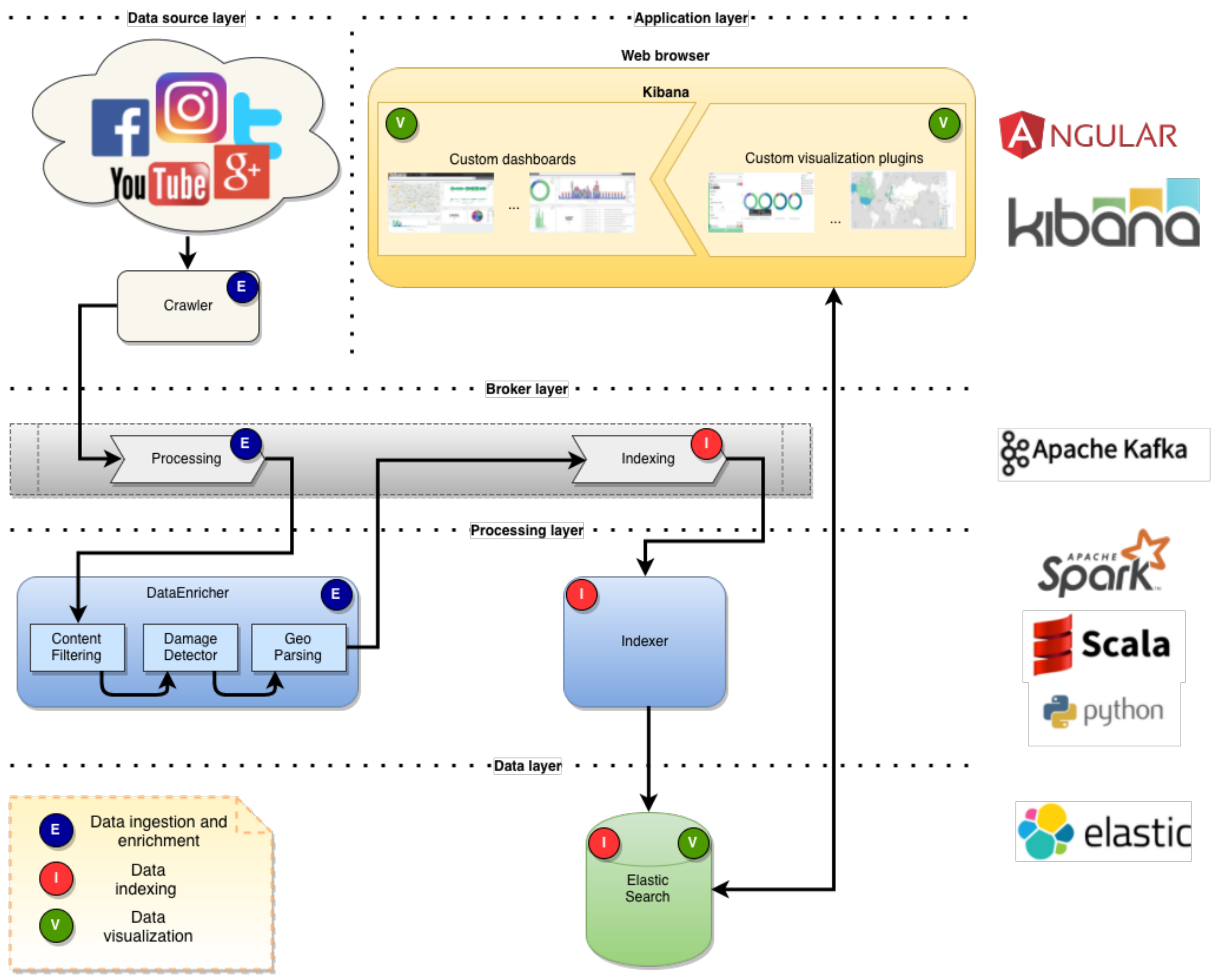

Fig. 1 The logical architecture of our crisis mapping system. On the left-hand side, the architecture is organized in a stack of layers: from the topmost data source and application layers of our system to the broker, processing, and data layers. On the right-hand side, for each layer are reported the technologies adopted to design and implement the components of the system.

data resiliently, with no loss in case of system failures (e.g., hardware faults).

SM messages are fetched from the *Processing* queue and processed by an enrichment component called *DataEnricher*. As depicted in the figure, the *DataEnricher* is organized internally into a pipeline of three sub-modules that process, filter, and enrich incoming data. The *ContentFiltering* sub-module analyzes the tweets to select those that are relevant to the problem (e.g., whose contents relate to natural disasters). Relevant tweets are then analyzed by the *DamageDetector* sub-module, which in turn discriminates between data carrying or not carrying damage information. The tweets containing damage information are finally forwarded to the *GeoParsing* sub-module, which possibly enriches each tweet with information about the discovered geolocation. Upon completion of the pipeline, the enriched SM messages are pushed into another queue, the *Indexing* queue, waiting to be indexed and stored into the system. The *DataEnricher* is implemented using Spark[10] and uses a streaming approach to process incoming data from the Processing queue quickly.

3.2 Data Indexing

In the second step, enriched data is moved from the *DataEnricher* to a permanent storage through the *Indexing* queue. Data fetched from the queue is instantaneously parsed and pushed into a search and analytics engine. The choice of Elasticsearch[11] (ES) was driven by its capability to scale horizontally, as well as to perform fast search and aggregation on textual data leveraging its internal Apache Lucene[12] engine. In fact, the integration of ES with other software like Kibana

[10] `http://spark.apache.org`
[11] `https://www.elastic.co/products/elasticsearch`
[12] `https://lucene.apache.org/`

and Logstash makes it a valid solution for the storage system.

The role of the *Indexer* module is to map the data structure of a Twitter message into an optimized ES index. Each field type is treated differently to exploit the features of the engine to provide fast search and real-time analytics on stored data (e.g., ES tokenizes textual fields and provides an inverted terms list to search for in the related fields efficiently).

3.3 Data Visualization

In the third step, data stored in the index can be browsed and queried through the Kibana[13] software. Kibana is an open source visualization tool, part of the ELK stack [14] provided by Elasticsearch company. Kibana software provided a Web GUI to access Elasticsearch data without manually writing queries. With Kibana, it is possible to build real-time visualizations with handy insights, like real-time volumes, maps, bar charts and word clouds. Those visualizations can also be joined to build complex dashboards that allow the user to track changes over time for the most important metrics of the dataset. Moreover, Kibana supports the possibility to easily extend the internal visualization types to customize graphical views using the most appropriate visual analytics for data. For our purposes, we developed a plugin to replace the native visualization maps. The new plugin[15] supports multi-resolution choropleth maps and the possibility of normalizing data according to population, offering different options for what concerns the scales and their customization.

3.4 Real-time and scalability features

Considering the practical purpose of our system, we designed and developed `CrisMap` with performance in mind, keeping the architecture flexible and capable of scaling up horizontally. To maximize our throughput, we optimized all system's components, starting from the *Crawler* that is our data entry point. Specifically, the *Crawler* does not impose any limitation on the number of tweets delivered per unit of time. In fact, the only limitation is represented by rate limits of the Streaming API, which are estimated to be around 60 tweets per second (i.e., 1% of total Twitter traffic). In our experience with Italian emergencies, we just witnessed to peaks of a few hundreds of tweets per minute, which is far below the threshold.

In the sub-modules *ContentFiltering* and *DamageDetector*, the solution adopted, based on Embeddings representation[16], has a tangible positive effect on system performance. Indeed, the classification time of a single tweet, considering both encoding time of tweet's text into a vector and its prediction, is on average 0.12 milliseconds, guaranteeing the remarkable throughput of more than 8,000 tweets per second.

With regards to the *GeoParsing* sub-module, the time taken to extract geographic information from a tweet depends on the actual implementation – that is, on the specific semantic annotator leveraged to geoparse the tweet. For the sake of experimentation, we benchmarked several well-known semantic annotators to assess their throughput.

The fastest annotator, `TagMe`, specifically designed to operate in a streaming fashion thus performing on-the-fly annotations [**Ferragina2010**], is able to geoparse around 9 tweets per second. It is also worth noting that our prototypical implementation of the `CrisMap` system performs the geoparsing operation by querying the RESTful Web APIs of the semantic annotators. However, many semantic annotators are open source and can be installed and configured to run locally (e.g., `Dexter`[17] and `DBpedia Spotlight`[18]). Thus, in a production environment it would be possible – and highly advisable – to avoid Web API queries in favor of much faster local computations.

Next in the pipeline, `Elasticsearch` (`ES`) provides high-performance indexing operations. In fact, according to the official documentation, it is capable of indexing around 2,000 tweets per second [19].

Given these considerations regarding the throughput, the bottleneck is represented by the *GeoParsing* sub-module. However, these results are related to a system deployed with the lowest possible degree of parallelization. In fact, since `CrisMap` is based on scalable Big Data technologies, it is possible to replicate system components (e.g., *GeoParsing*) over a cluster of machines in order to achieve overall better performances.

Regarding the latency, we extensively tested the *Crawler* performance in collecting a real-time stream of $\sim$30,000 tweets. Results showed an average latency of about 0.17 seconds. The amount of time taken by the

[13] `https://www.elastic.co/products/kibana`

[14] `https://www.elastic.co/products`

[15] The plugin is publicly available at `https://github.com/marghe943/kibanaChoroplethMap.git` .

[16] See Section **??** for more details about the proposed approach.

[17] `https://github.com/dexter/dexter`

[18] `https://github.com/dbpedia-spotlight/model-quickstarter`

[19] `https://www.elastic.co/blog/elasticsearch-performance-indexing-2-0`

ContentFiltering and *DamageDetector* sub-modules to classify a single tweet is in the order of milliseconds, thus representing a negligible delay. `TagMe` instead introduces a delay of 0.12 seconds per tweet, on average. Finally, Elasticsearch takes ∼1 second to index a document and to make it available for search and visualization purposes[20]. Thus, the total delay of the `CrisMap` analysis pipeline is about 1.3 seconds, which perfectly fits our need to produce real-time crisis maps.

Finally, some of the APIs exploited by `CrisMap` impose usage limits on the number of calls. The main limitation is related to the APIs needed for the geoparsing operation. Such limitation depends on the specific semantic annotator used for geoparsing. `TagMe`, hosted within the distributed and highly scalable `SoBigData` European research infrastructure[21], does not impose any limitation on the number of API calls, which in turn, opens up the possibility to perform large-scale and parallel analyses. In addition, many semantic annotators are open source and can be installed and configured to run locally (e.g., `Dexter`[22] and `DBpedia Spotlight`[23]). Indeed, local installations avoid limitations in the number of API calls.

4 Datasets

The datasets used for this work are composed of tweets in the Italian language, collected in the aftermath of 5 major natural disasters. For our experiments, we considered different kinds of disasters, both recent and historical: 3 earthquakes, a flood, and a power outage. Specifically, the `L'Aquila` and the `Emilia` datasets are related to severe earthquakes that struck rural areas of Italy in 2009[24] and 2012 respectively[25]. The `Amatrice` dataset is related to a recent earthquake that struck central Italy in 2016[26]. The `Sardinia` dataset has been collected in the aftermath of a flash flood occurred in the Sardinia island in 2013[27]. Finally, the `Milan` dataset describes a power outage occurred in the metropolitan city of Milan (northern Italy) in 2013. To investigate a wide range of situations, we picked disasters having variable degrees of severity: some caused only moderate damage, while other produced widespread damage and casualties.

The datasets were created by using the Twitter's Streaming API[28] for recent disasters, and the Twitter resellers' Historical APIs[29] (GNIP) for past disasters. The APIs give access to a global set of tweets, optionally filtered by search keywords. We exploited a different set of search keywords for every different disaster to collect the most relevant tweets about it. Whenever possible, we resorted to hashtags specifically created to share reports of a particular disaster, such as the *#allertameteoSAR* hashtag for the `Sardinia` dataset. In this way, we were able to select only tweets related to that disaster. However, for historical disasters, we could not rely on specific hashtags and had to exploit generic search keywords already proposed in literature, see [**Sakaki2013**, **Avvenuti2014earthquake**, **Avvenuti2014ears**]. This is the case of the `L'Aquila` dataset, for which we exploited the "terremoto" (*earthquake*) and "scossa" (*tremor*) Italian keywords. Also, we only used "fresh" data shared in the aftermath of the disasters under investigation. For instance, all the 3,170 tweets in the `Emilia` dataset were posted in less than 24 hours since the earthquake occurred.

Tweets in the `L'Aquila`, `Emilia`, and `Sardinia` datasets have been manually annotated for mentions of damage according to the 3 following classes: (i) tweets related to the disaster and carrying information about damage to infrastructures/communities (*damage*); (ii) tweets related to the disaster but not carrying relevant information for the assessment of damage (*no damage*); (iii) tweets not related to the disaster (*not relevant*). The inclusion of a class for tweets that are not related to a disaster (*not relevant*) is necessary because the automatic data collection strategy we adopted does not guarantee that all the tweets collected are related to the disaster under investigation. This aspect is especially true for the datasets collected with generic search keywords and represents a further challenge for our classification task. The manual annotation of damage mentions among tweets is exploited to train and validate our damage detection classifier, as thoroughly explained in Section **??**. Furthermore, following the same approach adopted in [**Middleton2014**, **Gelernter2013**], we carried out an additional manual annotation of 1,900 random tweets of the aforementioned datasets with regards to mentions of places/locations. This further annotation is exploited to validate our geoparsing results, as described in Section **??**. The `Milan` dataset is also used for a comparison of geoparsing techniques in

[20] `https://www.elastic.co/guide/en/elasticsearch/reference/6.0/tune-for-indexing-speed.html`
[21] `http://www.sobigdata.eu/`
[22] `https://github.com/dexter/dexter`
[23] `https://github.com/dbpedia-spotlight/model-quickstarter`
[24] `https://en.wikipedia.org/wiki/2009_L'Aquila_earthquake`
[25] `https://en.wikipedia.org/wiki/2012_Northern_Italy_earthquakes`
[26] `https://en.wikipedia.org/wiki/August_2016_Central_Italy_earthquake`
[27] `https://en.wikipedia.org/wiki/2013_Sardinia_floods`
[28] `https://dev.twitter.com/docs/api/streaming`
[29] `http://gnip.com/sources/twitter/historical`

Section **??**, since it was already exploited in previous work [**Middleton2014**]. The `Emilia` and `Sardinia` datasets are also used in Section **??** to quantitatively validate our crisis maps against authoritative data. Finally, the `Amatrice` dataset is exploited in Section **??** as a case study of our system in the aftermath of the recent central Italy earthquake.

Notably, the total number of 15,825 tweets in our datasets, shown in Table **??** along with other details, is greater than those used in other related works, such as [**Middleton2014**] (6,392 tweets across 4 datasets), and [**Gelernter2011**] (2,000 tweets for a single dataset).

To better understand the importance of geoparsing in a crisis mapping task, in Table **??** we also reported the number of tweets natively geolocated (GPS column). Geolocation of these tweets is performed directly by Twitter whenever a user enables GPS or WiFi geolocation. Statistics on our datasets confirm previous findings reporting that only a small percentage ($1\% \div 4\%$) of all tweets are natively geolocated [**Cheng2010**, **Cresci2015wise**]. As introduced in Section **??**, the low number of natively geolocated tweets drastically impairs crisis mapping, hence the need for a geoparsing operation. Noticeably, none of the 1,062 tweets of the `L'Aquila` dataset, dating back to 2009, are natively geolocated.

5 Mining text to search for damage

The detection of damage in SM messages is a challenging task due to the almost completely unstructured nature of the data to be analyzed [**Cresci2015**]. The *DataEnricher* component of Figure **??** analyzes the content of tweets with the twofold goal of discarding irrelevant tweets and labeling the relevant ones according to the presence (or lack thereof) of damage mentions. In our system, “damage” refers both to damage to buildings and other structures and to injuries, casualties, and missing people. In other words, damage encompasses all harmful consequences of an emergency on infrastructures and communities.

In this work, we approach the damage detection problem as a two-levels binary classification task. For our purposes, we are interested in identifying four different classes of tweets:

– *Not relevant*: tweets not related to a natural disaster.
– *Relevant*: tweets related to a natural disaster.
– *Without damage*: tweets related to a natural disaster but which do not convey information relevant to damage assessment.
– *With damage*: tweets related to a natural disaster which convey information relevant to damage assessment.

At the first level, the binary classifier (sub-module *ContentFiltering* in Figure **??**) acts as a filter to discriminate between not relevant and relevant tweets, allowing only the latter ones to pass over to the second level. The classifier at the second level (sub-module *DamageDetector* in Figure **??**) discriminates between tweets containing relevant information about damage and those not containing information relevant for damage assessment.

We built the two binary classifiers using the Support Vector Machines (SVM) algorithm [**Cortes1995**] with a simple linear kernel as machine learning method[30]. The set of features used by the single classifier was obtained from tweets by analyzing the textual content of a tweet using an approach based on word embeddings [**Bengio2003**].

The NLP (Natural Language Processing) research field has gained much attention in the last years, due to the renewed interest in neural networks technologies (e.g., deep learning), the continuous growth of the computational power of the CPUs and GPUs, and the explosion of available data that can be used to train these neural networks in an unsupervised way. In this work, we specifically used the approach proposed by Mikolov *et al.* [**NIPS2013˙5021**] describing the `word2vec` software, which is currently the most popular model for embeddings used in NLP-related tasks. Word embeddings techniques are an elegant solution to the problem of the features sparseness in document vectors created by using classic approaches like bag-of-words, char-N-grams, or word-N-grams [**Sebastiani2002**]. From one side, they aim to create a vector representation with a much lower dense dimensional space. On the other side, they are useful for extracting semantic meaning from text, to enable natural language understanding by learning the latent context associated with every specific word extracted from training data. More formally, distribute word representations (word embeddings) learn a function $W : (word) \rightarrow R^n$ that maps a word into a vector where each dimension models the relation of that specific word with a specific latent aspect of the natural language, both syntactically or semantically. The vectors are learned through the training of neural language models together with the parameters of the network from a set of unannotated texts and according to an objective function (e.g., distributional hypothesis[31]) [**Bengio2013**]. The

[30] As software implementation we used the SVC class available in the `scikit-learn` Python package.
[31] The meaning of this hypothesis is that words appearing in similar contexts often have a similar meaning.

Table 1 Characteristics of the Datasets.

				tweets					used in
dataset	**type**	**year**	**users**	*damage*	*no damage*	*not relevant*	*GPS*	*total*	**sections**
`L'Aquila`	Earthquake	2009	563	312 (29.4%)	480 (45.2%)	270 (25.4%)	0 (0%)	1,062	??, ??, ??
`Emilia`	Earthquake	2012	2,761	507 (16.0%)	2,141 (67.5%)	522 (16.5%)	205 (6.5%)	3,170	??, ??
`Milan`	Power outage	2013	163	-	-	-	15 (3.8%)	391	??
`Sardinia`	Flood	2013	597	717 (73.5%)	194 (19.9%)	65 (6.6%)	51 (5.2%)	976	??, ??, ??
`Amatrice`	Earthquake	2016	7,079	-	-	-	21 (0.2%)	10,226	??

resulting word vectors are computed to maintain the semantic/syntactic relationship existent between words. They allow to (i) visualize the vectors of similar words very close into a given metric space (e.g., visualize the word embeddings space on 2-D space through techniques like t-SNE); (ii) compute algebraic operations on vectors to point out some specific characteristic of the data (e.g., $W(\text{“}queen\text{”}) \cong W(\text{“}king\text{”}) - W(\text{“}man\text{”}) + W(\text{“}woman\text{”}))$.

The approach we used in our system to build the damage-detection component is entirely different from the one presented in our recent work on the same research topic [**Avvenuti2016impromptu**]. In our previous work, we built this component using a multiclass classifier based on SVM with a linear kernel but operating with features extracted from training data using classic NLP techniques [**Cresci2015**]. The classifier based on classic NLP guarantees a good accuracy at classification time, but it has some significant drawbacks that we aim to mitigate in this work. In particular, in the past work, we used five different classes of features (e.g., lexical text features, morphosyntactic features, sentiment-analysis features, etc.) that are almost language-dependent and extracted with a quality level strongly dependent from the set of NLP tools and resources available to analyze the textual data. The choice of which features to extract (often referred to as the “feature engineering” problem) is a non-trivial task that must be solved to provide the classifier a set of features sufficiently informative for the resolution of the specific problem, given the input domain. Moreover, the total number of features extracted in this way is very high (in the order of several hundred-thousands features), and it has a severe impact on the system in terms of performance both at training and classification time. This performance degradation depends both on the complexity of the SVM algorithm (which growths linearly with the number of features) and the time spent to use external NLP tools to enrich data. Conversely, using our new approach based on word embeddings helps to handle this type of issues because this technique completely avoids the feature engineering process and makes the system almost independent of any specific language. The only requirement affecting this model is the language coherence on a set of unannotated textual data used for the training of the embeddings, a condition often satisfiable very easily and with no human effort on many application domains (such as the Twitter domain). In particular, the modules we developed in this work are focused just on the Italian language, but the proposed methodology is very easily adaptable to other different languages, being Twitter a very popular multilanguage service. Another useful implication of word embeddings is the reduced number of features used by the classifier, being often in the order of few hundreds, and therefore contributing to speed-up learning new models and classifying new documents.

Table 2 NLP classifier vs. Embeddings classifier in terms of $F1$ effectiveness compared over the three available labeled datasets.

	dataset	**damage**	**no damage**	**not relevant**
NLP	`L'Aquila`	0.89	0.87	0.73
	`Emilia`	0.90	0.87	0.49
	`Sardinia`	0.89	0.46	0.29
EMB	`L'Aquila`	0.85	0.82	0.72
	`Emilia`	0.85	0.87	0.52
	`Sardinia`	0.82	0.43	0.33

To validate the goodness of our new proposal based on embeddings, we compared the $F1$ effectiveness [**Sebastiani2002**] obtained with a 10-fold cross evaluation of both approaches using the multiclass single-label configuration proposed in our previous work, as shown in Table **??**. The NLP classifier (reported as NLP) has been tuned as described in [**Avvenuti2016impromptu**] while the embeddings classifier (reported as EMB) has been tuned as in the following. The word embeddings vectors have been obtained by training the system on a dataset composed of the union of the tweets coming from `L'Aquila`, `Emilia`, `Sardinia`, and `Amatrice` using the CBOW method [**NIPS2013˙5021**], and the size of the embeddings was set to 100. Given a tweet, to obtain a single vector representing the tweet content, we computed the tweet vector as the averaged sum of the vectors of the words contained in the text of the tweet[32]. The SVM classifier working over embeddings has been set to use the linear kernel with the parameter $C = 1$, and we have handled unbalanced data distribution among labels by assigning to each label a weight proportional to its popularity[33]. As reported in the Table **??**, the embeddings classifier obtains very similar results to the NLP classifier in all tested datasets, confirming that our new approach provides all previously discussed advantages without sacrificing too much the accuracy of the damage detection system.

This comparison also suggests that to improve the accuracy of the embeddings classifier we can operate at two different levels. Firstly, we can use more training data to train the classifier: a simple and effective solution is to merge the four separated datasets (`L'Aquila`, `Emilia`, `Sardinia`, and `Amatrice`) into one bigger training dataset. Secondly, to simplify the task of classification model's learning, we can change the target problem from a multiclass single-label classification problem into a pair of separated binary classification problems, as described at the beginning of this section. This last modification also has the advantage to separate the filter part that identifies relevant tweets from the damage detection phase. This separation enables parallel execution of the damage classification task and the geoparsing task, decreasing the total time required to process a single tweet entirely.

We tested the system with the proposed improvements using a 10-fold cross evaluation and by optimizing the model parameters to build a reasonably good set of classifiers. The measure used to drive the choice in the best configuration values is the micro-averaged $F1$ [**Sebastiani2002**]. The set of parameters subject to optimization were the following:

- *Embeddings dataset*: the dataset used to learn word embeddings. We have used 5 possible different datasets: `Amatrice`, `L'Aquila`, `Emilia`, `Sardinia`, and the union of all previous ones (`All`).
- *Embeddings size*: the size used to represent word embeddings vectors and consequently the vector size of a single tweet. The possible set of values are 50, 100, 200, and 400. The default value is 100.
- *Class weight*: the weight of the errors associated with the positive class used in the two binary classifiers. The positive classes were "Not relevant" for filter classifier and "With damage" for damage classifier. The possible set of values were 1.0, 1.5, 2.0, 2.5 and "Balanced" (same meaning as explained in footnote **??**). The default value is 'Balanced'.
- C: indicates the cost penalty associated with a misclassification. The possible set of values are $\in$ [0, 12] with a step increment of 0.1. The default value is 1.0.

To avoid testing all possible combinations of the above parameters, and to choose a reasonably good set of parameter values, we followed a simplified procedure as illustrated in the following. Using the order of the parameters as described above, we optimize one parameter at a time testing the full set of values for that specific parameter and using the default values for the other parameters. In case one of the other parameters has been optimized already, we use its best-found value instead. The dataset used to evaluate the system, composed by the union of the `L'Aquila`, `Emilia` and `Sardinia` datasets, has been generated in two different versions, one for filtering and one for damage detection. In the case of filtering, every tweet in the dataset originally labeled with "With damage" or "Without damage" labels has been relabeled with "Relevant" label, resulting in a final dataset containing 4,351 relevant tweets and 857 not relevant tweets. In the case instead of damage detection, every tweet in the dataset originally labeled with "Not relevant" has been relabeled with "Without damage" label, resulting in a final dataset containing 3,672 tweets marked with "Without damage" label and 1,536 tweets marked with "With damage" label.

In Table **??** and Table **??** we report the experimental results obtained from the optimization process, respectively while building the filter classifier and the damage classifier. Except for the C parameter, which we report only as the best value found, we have marked with ** the best configuration found among all tested ones. In the case of parameters "Embeddings dataset" and "Embeddings size" in both type of classifiers the best results are obtained with the full dataset (`All`) and 100 as the dimension of embeddings[34]. The optimal weight class values are different for each classifier reflecting the fact that the data distribution is very different between the two used datasets. In particular, the dataset used for filter classifier is more unbalanced towards relevant tweets, resulting in more variance in the results obtained for each tested configuration and in

[32] We did not use more sophisticated methods like "Paragraph Vector" [**DBLP:conf/icml/LeM14**] because these statistical methods do not work well for small texts like tweets.

[33] We used the 'balanced' value for class weight, see `scikit-learn` documentation at `http://bit.ly/2g5QSqk`. In this way we indicate to SVM to treat the various labels in different ways during the training phase, giving more importance to class errors (measured with used loss function) made for skewed classes.

[34] In case of configurations with equal results in terms of F1 we prefer to choose those having more balanced values between precision and recall measures.

Table 3 Choice of Optimal Parameters for Embeddings Classifier Filtering Relevant/not Relevant Tweets.

		Not relevant			Relevant			Micro avg results		
	Configuration	**Pr**	**Re**	**F1**	**Pr**	**Re**	**F1**	**Pr**	**Re**	**F1**
Emb. dataset	`Amatrice`	0.28	0.73	0.41	0.92	0.63	0.75	0.82	0.65	0.69
	`L'Aquila`	0.26	0.89	0.40	0.96	0.50	0.65	0.84	0.56	0.61
	`Emilia`	0.28	0.82	0.41	0.94	0.58	0.72	0.83	0.62	0.67
	`Sardinia`	0.21	0.85	0.34	0.93	0.37	0.53	0.81	0.45	0.50
	`All` **	0.36	0.79	0.49	0.95	0.72	0.82	0.85	0.73	0.76
Emb. size	50	0.35	0.80	0.48	0.95	0.71	0.81	0.85	0.72	0.76
	100 **	0.36	0.79	0.49	0.95	0.72	0.82	0.85	0.73	0.76
	200	0.35	0.79	0.49	0.94	0.72	0.81	0.85	0.73	0.76
	400	0.35	0.79	0.49	0.95	0.71	0.81	0.85	0.73	0.76
Class weight	1.0	1.00	0.00	0.00	0.84	1.00	0.91	0.86	0.84	0.76
	1.5	0.42	0.23	0.30	0.86	0.94	0.90	0.79	0.82	0.80
	2.0 **	0.44	0.54	0.48	0.91	0.86	0.88	0.83	0.81	0.82
	2.5	0.42	0.59	0.49	0.91	0.84	0.88	0.83	0.80	0.81
	Balanced	0.36	0.79	0.49	0.95	0.72	0.82	0.85	0.73	0.76
Best C	9.1	0.43	0.56	**0.49**	0.91	0.86	**0.88**	0.83	0.81	**0.82**

Table 4 Choice of Optimal Parameters for Embeddings Classifier Identifying With Damage/Without Damage Tweets.

		Without damage			With damage			Micro avg results		
	Configuration	**Pr**	**Re**	**F1**	**Pr**	**Re**	**F1**	**Pr**	**Re**	**F1**
Emb. dataset	`Amatrice`	0.92	0.75	0.82	0.58	0.85	0.69	0.82	0.77	0.78
	`L'Aquila`	0.87	0.89	0.88	0.72	0.69	0.70	0.83	0.83	0.83
	`Emilia`	0.95	0.88	0.91	0.76	0.89	0.82	0.89	0.88	0.89
	`Sardinia`	0.91	0.89	0.90	0.74	0.79	0.76	0.86	0.86	0.86
	`All` **	0.96	0.87	0.91	0.75	0.92	0.83	0.90	0.89	0.89
Emb. size	50	0.96	0.87	0.91	0.75	0.92	0.82	0.90	0.88	0.89
	100 **	0.96	0.87	0.91	0.75	0.92	0.83	0.90	0.89	0.89
	200	0.96	0.87	0.91	0.75	0.92	0.82	0.90	0.88	0.89
	400	0.96	0.87	0.91	0.75	0.92	0.82	0.90	0.88	0.89
Class weight	1.0	0.93	0.92	0.92	0.81	0.84	0.82	0.89	0.89	0.89
	1.5 **	0.95	0.89	0.92	0.78	0.88	0.83	0.90	0.89	0.89
	2.0	0.96	0.88	0.92	0.76	0.91	0.83	0.90	0.89	0.89
	2.5	0.96	0.87	0.91	0.75	0.92	0.82	0.90	0.88	0.89
	Balanced	0.96	0.87	0.91	0.75	0.92	0.83	0.90	0.89	0.89
Best C	0.4	0.95	0.90	**0.92**	0.78	0.88	**0.83**	0.90	0.89	**0.89**

general providing quite low F1 values for class "Not relevant" (~ 0.5). Anyway, the results also show that the errors made for this last class by the filter classifier are partially recovered by the damage classifier, considering that the F1 accuracy on both damage classes is remarkably high (> 0.8 in both cases), resulting into an effective and useful implementation of the damage detection component.

The results discussed above are obtained exploiting ideal conditions for data availability – i.e., each event typology is well represented in the training data. In Table **??** we report transfer learning results obtained by training the system (specifically, the damage classifier) on data related to a given event and by reusing the learned model on data related to a different event [**pan2010survey**]. This scenario better resembles the real-world operative conditions in the aftermath of an emergency. We performed the experiments using the optimal parameters reported in Table **??**, this time using each available dataset separately, with a split of 70/30 between training and test data. Each row corresponds to the set of training data used, while each column corresponds to the set of data tested. The table shows that the best results are obtained, for each tested dataset, when the corresponding training data is used (e.g., `Emilia` vs. `Emilia`). The loss in classifier's effectiveness is acceptable when considering events of the same type (e.g., `L'Aquila` and `Emilia`, which are both earthquakes) with a deterioration in performance comprised between 6.2% and 19.1%. Instead, for events of a different type, the results are remarkably worse (e.g., in `Sardinia` vs. `Emilia` the loss for `Emilia` is 81.2% with respect to the best obtainable effectiveness). These experiments confirm the conclusions reported by other similar works in the literature on this research topic, such as those reported in [**Cresci2015**]. In conclusion,

Table 5 Transfer-learning results obtained with SVM classifiers using embeddings representation across `L'Aquila`, `Emilia` and `Sardinia` datasets.

Training/Test	`L'Aquila`			`Emilia`			`Sardinia`		
	Without damage F1	With damage F1	Micro F1	Without damage F1	With damage F1	Micro F1	Without damage F1	With damage F1	Micro F1
`L'Aquila`	0.92	0.81	**0.89**	0.93	0.69	0.90 (-6.2%)	0.63	0.85	0.80 (-0.0%)
`Emilia`	0.75	0.66	0.72 (-19.1%)	0.98	0.86	**0.96**	0.10	0.84	0.64 (-20.0%)
`Sardinia`	0.19	0.50	0.29 (-67.4%)	0.17	0.27	0.18 (-81.2%)	0.59	0.88	**0.80**

to obtain a reasonable effectiveness, we suggest to possibly train the system using datasets that cover all the possible types of events that the system should process.

6 Geoparsing

Geoparsing – namely, the resolution of toponyms in a textual document to a set of geographic coordinates – is typically considered the focal point of crisis mapping and has been faced since the diffusion of the Web. This task is typically solved extracting toponyms from a text and looking up for matches in gazetteers containing all the possible matches between a set of place names and their geographic coordinates [**Middleton2014**]. This approach requires an offline phase where the geoparsing system is specifically set to work in a geographically-limited region. Although this solution is effective for limited areas and offers a fast response, it is practically infeasible to load associations between toponyms and coordinates for a wide region or a whole country [**Avvenuti2016impromptu**]. Another challenge related to geoparsing is that of toponymic polysemy – that is, the situation in which a toponym might have different meanings, thus possibly referring to different places, according to the context in which it is used (e.g., the word "Washington" may refer to the first US president, to the US capital, to the US state, etc.)[35]. This last problem is particularly relevant for geoparsing systems based on gazetteers lookups since with this approach there is no way to perform a disambiguating operation of the toponyms to understand their actual meanings [**Cresci2015wise**, **Avvenuti2016impromptu**].

To overcome these limitations, the *GeoParsing* submodule of the *DataEnricher*, shown in Figure **??**, adopts semantic annotators in the geoparsing process. Semantic annotation is a process aimed at augmenting a plain-text with relevant references to resources contained in knowledge-bases such as Wikipedia and DBpedia. The result of this process is an enriched (annotated) text where mentions of knowledge-bases entities have been linked to the corresponding Wikipedia/DBpedia resources. This annotation process is highly informative since it enables the exploitation of the rich information associated with the Wikipedia/DBpedia resources that have been linked to the annotated text. Here, we exploit semantic annotations for our geoparsing task by checking whether knowledge-bases entities, which have been linked to our tweets, are actually places or locations. Semantic annotation also has the side effect of alleviating geoparsing mistakes caused by toponymic polysemy. In fact, some terms of a plain-text can potentially be linked to multiple knowledge-bases entities. Semantic annotators automatically perform a disambiguating operation and only return the most likely reference to a knowledge-base entity for every annotated term. Overall, our proposed geoparsing technique overcomes 2 major problems affecting current state-of-the-art crisis mapping systems: (i) it avoids the need to preload geographic data about a specific region by drawing upon the millions of resources of collaborative knowledge-bases such as Wikipedia and DBpedia, (ii) it reduces the geoparsing mistakes caused by toponymic polysemy that are typical of those systems that perform the geoparsing task via lookups in preloaded toponyms tables. Another additional useful characteristic of our geoparsing technique is that it is unsupervised, unlike the one presented in [**Gelernter2011**].

Because of these reasons, our geoparsing technique is particularly suitable for being employed in a system aimed at producing crisis maps *impromptu*, such as the one that we are proposing. The possibility to link entities mentioned in emergency-related messages to their pages also allows exploiting the content of their Wikipedia/DBpedia pages to extract other useful information about the unfolding emergency. Most commonly used semantic annotators also provide a confidence score for every annotation. Thus, it is possible to leverage this information and only retain the most reliable annotations, discarding the remaining ones.

Among all currently available semantic annotators, `CrisMap` is currently based on `TagMe` [**Ferragina2010**], `DBpedia Spotlight` [**Mendes2011**], and `Dexter` [**Trani2014**], three well-known, state-of-the-art systems [**Usbeck2015**]. `TagMe` is a service of text annotation and disambiguation developed at the University of Pisa. This tool provides a Web appli-

[35] `http://en.wikipedia.org/wiki/Washington`

	GPS		DBpedia Spotlight		Dexter		TagMe	
L'Aquila	0	(0%)	271	(25.5%)	578	(54.4%)	721	(67.9%)
Emilia	205	(6.5%)	975	(30.8%)	1,037	(32.7%)	1,671	(52.7%)
Milan	15	(3.8%)	99	(25.3%)	320	(81.8%)	364	(93.1%)
Sardinia	51	(5.2%)	530	(54.3%)	784	(80.3%)	582	(59.6%)

Table 6 Contribution of our *GeoParsing* sub-module, used in conjunction with the different semantic annotators, on the number of geolocated tweets.

cation[36] as well as a RESTful API for programmatic access and can be specifically set to work with tweets. Language-wise, TagMe supports analyses on English, Italian, and German texts. Since TagMe is based on the Wikipedia knowledge-base, the annotated portions of the original plain-text are complemented with the ID and the name of the linked Wikipedia page. TagMe also returns a confidence score *rho* for every annotation. Higher *rho* values mean annotations that are more likely to be correct. After annotating a tweet with TagMe, we resort to a Wikipedia crawler to fetch information about all the Wikipedia entities associated to the annotated tweet. In our implementation, we sort all the annotations returned by TagMe on a tweet in descending order according to their *rho* value, so that annotations that are more likely to be correct are processed first. We then fetch information from Wikipedia for every annotation and check whether it is a place or location. The check for places/locations can be simply achieved by checking for the *coordinates* field among Wikipedia entity metadata. We stop processing annotations when we find the first Wikipedia entity that is related to a place or location, and we geolocate the tweet with the coordinates of that entity. The very same algorithmic approach is employed for the exploitation of the other semantic annotators: DBpedia Spotlight and Dexter. Indeed, it is worth noting that our proposed geoparsing technique does not depend on a specific semantic annotator, and can be implemented with any annotator currently available, or with a combination of them. Regarding language support, DBpedia Spotlight is capable of performing analyses for 16 different languages, including English and Italian, and also allowing users to train models for additional languages. Instead, Dexter natively supports only the English language at the time of writing, but can be extended to work with any other language, similarly to DBpedia Spotlight.

We used our *GeoParsing* sub-module to geocode all the tweets of our datasets. Then, following the same approach used in [**Middleton2014**] and [**Gelernter2013**], we manually annotated a random subsample of 1,900 tweets to validate the geoparsing operation. Noticeably, our system achieves results comparable to those of the best-of-breed geoparsers with an $F1 = 0.84$, whether the systems described in [**Middleton2014**] and [**Gelernter2013**] scored in the region of $F1 \sim 0.80$. Furthermore, to better quantify the contribution of our *GeoParsing* sub-module, we report in Table **??** the number of natively geolocated tweets (GPS column) and the number of tweets geolocated by our *GeoParsing* sub-module via TagMe, DBpedia Spotlight, and Dexter. As shown, our system geoparses the highest number of tweets based on the annotations of TagMe, for all datasets, except for the Sardinia one, for which the best results are achieved with Dexter's annotations. In any case, our *GeoParsing* sub-module managed to geoparse from a minimum of 25.3% tweets to a maximum of 93.1% tweets of the Milan dataset, meaning that almost all tweets of that dataset were associated to geographic coordinates, allowing to use such tweets in our crisis maps.

7 Mapping data

Among the diverse data visualization techniques, one that is commonly employed to represent the geographic distribution of a statistical variable is the *choropleth map*. A choropleth map is a thematic representation in which subareas of the map are filled with different shades of color, in proportion to the measurement of the given variable being displayed[37]. This visualization technique is usually exploited to depict the spatial distribution of demographic features such as population, land use, crime diffusion, etc. In CrisMap we exploit the same visualization technique to show the spatial distribution of damage in the aftermath of an emergency. A clear advantage of exploiting choropleth maps instead of the typical on/off maps used in previous works [**Middleton2014**], lies in the possibility to apply different shades of color to the different areas of the map, according to the estimated extent of damage suffered by that area. This complements well with

[36] https://tagme.d4science.org/tagme/

[37] https://en.wikipedia.org/wiki/Choropleth_map

the prioritization needs that arise in the first phases of an emergency response. Most notably, our system can be easily extended to produce different end-results. In other words, we can choose to produce a choropleth crisis map or any other visualization of the analyzed tweets exploiting the high flexibility of the Kibana interface. Notably, `CrisMap` is capable of producing choropleth crisis maps with a spatial resolution at the level of *municipalities*. In any case, when tweets are accurate enough, it is also possible to precisely identify objects (e.g., a specific building) that suffered damage. It is also worth noting that the choice to produce crisis maps showing the estimated degree of damage among the different municipalities is not due to a region-level only geoparsing. Indeed, the exploitation of semantic annotators potentially allows geocoding every entity that has an associated Wikipedia/DBpedia page. So, in those cases when a tweet contains detailed geographic information, it is possible to geoparse it to building- or even street-level. Our choice to produce crisis maps at the level of municipalities is instead motivated by an effort to rigorously compare our crisis maps to data officially released by the Italian Civil Protection agency, which reports damages at the municipality level.

Here, we first show the accuracy of our visualizations referring to two case studies, the `Emilia` earthquake and the `Sardinia` flood. We compare the maps realized with SM data against those produced using authoritative data provided by Italian civil protection agency. Finally, we present results obtained by applying `CrisMap` to study the `Amatrice` earthquake. Unfortunately, there is no fine economic loss estimation for the `Amatrice` earthquake, since the same area suffered a second severe shake just a few months after the first one when official damage surveys still had to be completed. Nonetheless, in the case of the `Amatrice` earthquake, we are still able to provide a qualitative case-study of our crisis maps.

7.1 Quantitative validation

The authoritative data that we used for the comparison is the economic loss/damage (quantified in millions of euros) suffered by the different municipalities, as assessed by the Italian Civil Protection agencies of Emilia Romagna and Sardinia. Specifically, for the `Emilia` earthquake, authoritative data has been collected via *in situ* damage surveys carried in out in the months after the earthquake. Results of such surveys are published in the `http://www.openricostruzione.it` Web site, maintained by the regional administration of Emilia Romagna, and comprise detailed information on the economic losses suffered as a consequence of the earthquake, as well as on the overall status of the rebuilding process. With regards to the `Sardinia` flood of 2013, the Italian Civil Protection Agency surveyed all damaged municipalities and reported the estimated economic losses suffered by private properties, public infrastructures, and production (industrial and agricultural) facilities. Final results of the damage survey have been published in a public document[38] on February 24, 2014.

It is possible to perform a first quantitative evaluation of our crisis maps following the approach used in [**Middleton2014**], that is, evaluating the system as a classification task. Under this hypothesis, the goal of the system is to detect damaged municipalities disregarding of those requiring prioritized intervention – namely, those that suffered the most damage. Thus, we can exploit well-known machine learning evaluation metrics to compare crisis maps generated by our system with those obtained from official data. The comparison is performed by checking whether a municipality with associated damage to authoritative data also appears as damaged in our crisis maps.

Table **??** reports the results of this comparison for the `Emilia` earthquake. We first consider all the municipalities of the affected region, namely the Emilia Romagna region, and then we repeat the comparison considering only those municipalities that suffered a significant degree of damage (more than 10% of the damage suffered by the Ferrara municipality, which is the maximum value for the `Emilia` earthquake). As clearly highlighted by Table **??**, the proposed crisis mapping system can accurately identify the areas where the damage occurred. However, not all the damaged municipalities are identified by the system, as represented by the low Recall value in the first row of the table. Regardless, when we remove from the comparison those municipalities that suffered the lowest damage, the Recall metric reaches a value of 0.813, showing the system's ability in detecting areas that suffered a significant amount of damage. In other words, the majority of the mistakes of our system occurred in municipalities that suffered relatively low damage, and not on those requiring immediate attention. The same also applies for the `Sardinia` flood, as reported in Table **??**, with an improvement of the Recall metric from 0.128 to 1 when considering municipalities that suffered more than 2% of the maximum damage (i.e., the damage suffered by the municipality of Olbia).

[38] `http://www.regione.sardegna.it/documenti/1_231_20140403083152.pdf` - Italian Civil Protection report on damage to private properties, public infrastructures, and production facilities.

Table 7 Binary detection of damaged municipalities for the `Emilia` earthquake.

	evaluation metrics					
task	*Precision*	*Recall*	*Specificity*	*Accuracy*	*F-Measure*	*MCC*
Detection of all damaged areas	1.000	0.178	1.000	0.797	0.303	0.375
Detection of areas that suffered significant damage	1.000	0.813	1.000	0.992	0.897	0.898

Table 8 Binary detection of damaged municipalities for the `Sardinia` flood.

	evaluation metrics					
task	*Precision*	*Recall*	*Specificity*	*Accuracy*	*F-Measure*	*MCC*
Detection of all damaged areas	0.833	0.128	0.993	0.814	0.222	0.280
Detection of areas that suffered significant damage	0.500	1.000	0.995	0.995	0.667	0.705

Overall, the results obtained by our system concerning the detection of damaged areas are comparable to those reported in [**Middleton2014**]. However, our system operated with a fine geographic resolution on 2 case studies of natural disasters that affected extensive, rural, and sparsely populated areas. Conversely, the system presented in [**Middleton2014**] has a fine resolution only for an emergency affecting a densely and uniformly populated area (Manhattan, New York) while it shows coarse resolution results for a disaster striking a wide area (the state of Oklahoma).

In addition to detecting damaged areas, `CrisMap` also visually sorts municipalities using a color hue that is proportional to the intensity of the damage suffered. In other words, it order ranks municipalities based on the (normalized) number of tweets conveying damage information. This original feature opens up the possibility to perform a finer evaluation of our crisis maps than that carried out in previous works. Indeed, it is possible to compare the ranking of damaged municipalities as obtained from tweets, with a ranking derived from authoritative sources, such as those provided by civil protection agencies. A crisis mapping system that can rapidly identify the most damaged areas would become a valuable tool in the first phases of emergency response when resource prioritization plays a dominant role. A possible way of performing such evaluation is by employing metrics that are typically used to assess the performance of ranking systems, such as search engines. Search engines are designed to return the most relevant set of results to a given user-submitted query. In our scenario, we can consider `CrisMap` as a basic "search engine" that returns a list of areas and that is specifically designed to answer a single, complex query: "which areas suffered the most damage?". Search engines are evaluated with several metrics and indices, aimed at capturing a system's ability to return desired resources (e.g., Web pages, text documents, etc.) among the first results. We can then evaluate the ability of `CrisMap` to correctly identify areas that suffered a high degree of damage by employing evaluation metrics of search engines. Specifically, among such well-known metrics are the *normalized Discounted Cumulative Gain* (*nDCG*) [**Jarvelin2002**] and the *Spearman's Rho coefficient*. The *nDCG* measures the performance of a "recommendation" (or ranking) system based on the graded relevance of the recommended entities. It is the normalized version of the *Discounted Cumulative Gain* and ranges from 0 to 1, with 1 representing the ideal ranking of the entities. This metric is commonly used in information retrieval to evaluate the performance of web search engines:

$$DCG_k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

$$nDCG_k = \frac{DCG_k}{IDCG_k}$$

where rel_i is the graded relevance of the result at position i and $IDCG_k$ is the maximum possible (ideal) DCG for a given set of entities.

Spearman's Rho instead measures the correlation between two variables described using a monotonic function, and it is evaluated as:

$$\rho = 1 - \frac{6\sum_i D_i^2}{N(N^2-1)}$$

where $D_i = r_i - s_i$ is the difference between actual position (given by the system) and expected position (given by the reports). For instance, it measures the correlation between the ideal output of the system (Civil Protection ordering) with the result of the system and describes how likely one variable is going to change (tweets with damage) given the other (damage amount). Being

Table 9 Ranking evaluation of most damaged municipalities.

dataset	*nDCG*	*Spearman's Rho*
`Emilia`	0.770	0.698
`Sardinia`	0.647	0.408

a correlation coefficient, it ranges from -1 to 1, with values in the region of 0 indicating no correlation. Using these metrics, we assessed the ability of our system in detecting the most stricken areas against authoritative data based on the economic damage suffered by the affected municipalities. Our experiments confirm that there is a considerable agreement between tweet-derived rankings and those based on authoritative data, as reported in Table **??**.

A simple test for statistical significance of our ranking results with Spearman's Rho further supports our claims, achieving a confidence score $> 99\%$ for both the `Emilia` earthquake and the `Sardinia` flood. Overall, results of our system in detecting all damaged areas, as well as the most damaged ones, demonstrate the applicability and the usefulness of `CrisMap` also in extensive, rural, and sparsely populated regions.

7.2 The qualitative Amatrice case-study

Figure **??** shows an excerpt of the `CrisMap` dashboard in the aftermath of the `Amatrice` earthquake. All the visualizations shown in figure are related to the `Amatrice` dataset, collected during the first hour from the earthquake occurrence. On top of Figure **??** are two choropleth maps – visualizations **??**(a) and **??**(b) – obtained by embedding the choropleth plugin, that we specifically developed for `CrisMap`, inside the Kibana interface. Figure **??**(a) shows the map generated from all the tweets that we collected, while Figure **??**(b) is obtained only from *damage* tweets. Our choropleth maps highlight the most damaged municipalities, namely Norcia, Amatrice and Accumoli, located in the vicinity of the epicenter. The map in Figure **??**(b), related only to damage tweets, is rather sparse, since in the first hour after the earthquake only a tiny fraction of tweets conveyed damage reports. This aspect is also clearly visible from Figure **??**(d), showing the volume of tweets collected and classified by `CrisMap` every minute. As shown, tweets reporting damage (red-colored) have been shared almost only in the last minutes of the first hour. Yet, despite the very few tweets reporting damage, the word-cloud of Figure **??**(c) highlights the key consequences of the earthquake: (i) the 3 most damaged villages Norcia, Amatrice, Accumoli; (ii) and several mentions of damage, such as "crolli" (collapsed buildings) and "danni" (widespread damage).

Notably, all the visualizations of Figure **??** are interactive and are updated in real-time, as new data is collected and analyzed by `CrisMap`. Regarding user interactions, for instance, it is possible to click on a specific municipality in the choropleth maps to update all other visualizations by showing only data that is related to the selected municipality. Alternatively, one could click on a word in the word-cloud to visualize the temporal and spatial distribution of tweets containing that word. These functionalities open up the possibility to promptly perform drill-down analyses of the most relevant facts.

8 Conclusion

We presented `CrisMap`: a crisis mapping system capable of supporting emergency responders during emergency management of natural or man-made disasters. The system can process incoming SM data from Twitter to quickly produce crisis maps that are useful to prioritize the allocation of available resources, especially in the first phases of the crisis, towards the populations and territories most affected by the specific disaster. The proposed solution is designed and built using Big Data technologies, allowing the system to be scalable, fault-tolerant, and capable of processing incoming data in near-real-time. To overcome the challenges resulting from the unstructured nature of Twitter data and to identify useful information for our purposes, we analyze the data using a two-fold perspective. On the one hand, we introduced a damage detection component exploiting word embeddings and a SVM classifier to detect messages reporting damage to infrastructures or injuries to the population. On the other hand, we proposed a message geolocation component that performs the geoparsing task by exploiting online semantic annotators and collaborative knowledge-bases. The approach using word embeddings has also been compared with a traditional one based on classic NLP techniques, pointing out the potential advantages of the former concerning complexity and performance of the proposed method. The accuracy and the reliability of the system were validated analytically comparing the experimental results of `CrisMap` against the authoritative data for two past disasters. Furthermore, we also performed a qualitative evaluation of the system on a case-study of a recent severe earthquake in Italy for which authoritative data are not available.

The proposed system offers some room for further improvements, to increase the readability of the maps

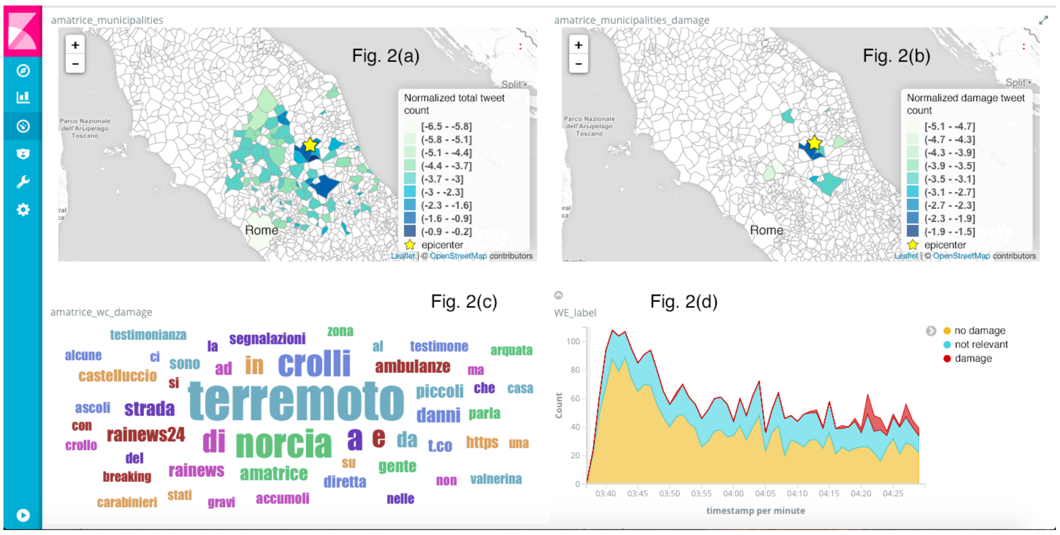

Fig. 2 Excerpt of the `CrisMap` dashboard in the aftermath of the `Amatrice` earthquake. The dashboard comprehends our ad-hoc choropleth maps showing the spatial distribution of tweets, a temporal view of collected and classified tweets per minute, and a word-cloud. The dashboard is easily extensible and customizable, allowing end-users to include more visualizations by choosing from the many ones natively provided by Kibana.

and, in general, for acquiring a better situational awareness of unfolding events. With regards to geoparsing results, recent developments of semantic annotation tools open up the possibility to provide more implementations of our proposed geoparsing technique. Therefore we envision the possibility to simultaneously exploit multiple semantic annotators in an ensemble or voting system. In the future, this approach could allow to obtain even better results and to overcome the possible limitations of a single annotator.

Moreover, to date, `CrisMap` only exploits textual data, but we believe that multimedia data, like images and live-videos, could contribute critical information for emergency responders. Thus, in the future, we aim at providing analytic and visual support for multimedia content since it appears as a promising direction for research.

Other avenues of future experimentation might be related to multi-source mining. Indeed, although Twitter is nowadays one of the preferred SM sources, given its open policies on providing data to third parties, data collection from multiple sources could mitigate the bias introduced by the analysis of a single SM.

Acknowledgements

This research is supported in part by the EU H2020 Program under the scheme `INFRAIA-1-2014-2015: Research Infrastructures` grant agreement #654024 *SoBigData: Social Mining & Big Data Ecosystem*, and by the MIUR (Ministero dell'Istruzione, dell'Università e della Ricerca) and Regione Toscana (Tuscany, Italy) funding the *SmartNews: Social sensing for Breaking News* project: `PAR-FAS 2007-2013`.

Regione Toscana | FAS Fondo Aree Sottoutilizzate 2007-2013 | REPUBBLICA ITALIANA

Marco Avvenuti is a professor of computer systems in the Department of Information Engineering at the University of Pisa. His research interests include human-centric sensing and social media analysis. Avvenuti received a PhD in information engineering from the University of Pisa. He is a member of the IEEE Computer Society.

Stefano Cresci is a PhD student in the Department of Information Engineering at the University of Pisa and a Researcher at IIT-CNR. His research interests include social media mining and knowledge discovery. Cresci received an MSc in computer engineering and an MSc in big data analytics and social mining from the University of Pisa. He is student member of the IEEE and member of the IEEE Computer Society.

Fabio Del Vigna is a PhD student in the Department of Information Engineering at the University of Pisa and an associate researcher at IIT-CNR. His research interests include social media mining and anomaly detection. Del Vigna received an MSc in computer engineering from the University of Pisa.

Tiziano Fagni is a Researcher working at IIT-CNR. He received an MSc in computer science and a PhD in information engineering from the University of Pisa. His research interests cover the areas of machine learning and data mining, specifically focusing on NLP problems applied in text analytics contexts. He is also interested in scalability issues of machine learning algorithms on Big Data application domains and lately also on deep learning technologies.

Maurizio Tesconi is a computer science Researcher at IIT-CNR. His research interests include social Web mining, social network analysis, and visual analytics within the context of open source intelligence. Tesconi received a PhD in information engineering from the University of Pisa. He is a member of the permanent team of the European Laboratory on Big Data Analytics and Social Mining.