Crisis Event Extraction Service (CREES) – Automatic Detection and Classification of Crisis-related Content on Social Media

Grégoire Burel
Knowledge Media Institute,
The Open University.
Milton Keynes, UK
g.burel@open.ac.uk

Harith Alani
Knowledge Media Institute,
The Open University.
Milton Keynes, UK
h.alani@open.ac.uk

ABSTRACT

Social media posts tend to provide valuable reports during crises. However, this information can be hidden in large amounts of unrelated documents. Providing tools that automatically identify relevant posts, event types (e.g., *hurricane*, *floods*, etc.) and information categories (e.g., *reports on affected individuals*, *donations and volunteering*, etc.) in social media posts is vital for their efficient handling and consumption. We introduce the Crisis Event Extraction Service (CREES), an open-source web API that automatically classifies posts during crisis situations. The API provides annotations for crisis-related documents, event types and information categories through an easily deployable and accessible web API that can be integrated into multiple platform and tools. The annotation service is backed by Convolutional Neural Networks (CNNs) and validated against traditional machine learning models. Results show that the CNN-based API results can be relied upon when dealing with specific crises with the benefits associated with the usage word embeddings.

Keywords

Event Detection, Word Embeddings, Deep Learning, Convolutional Neural Networks, API.

INTRODUCTION

Social media has emerged as a dominant channel for communities to gather and spread information during crises. Such media has proven itself as an invaluable information source in several recent natural and social crisis situations, such as floods (Starbird et al. 2010), earthquakes (Qu et al. 2011), wildfires (Vieweg et al. 2010), nuclear disasters (Thomson et al. 2012), and civil wars (Bercovici 2012).

A survey by the American Red Cross showed that 40% of the population would use social media during a crisis, and 76% of them expect their help requests to be answered within three hours. Doing this through manual analysis, however, is far from trivial, due to the sheer data volume generated during crisis events. For example, in a single day during the 2011 Japan earthquake, 177 million tweets related to the crisis were sent (Campanella 2006).

Although information is paramount during major crises, it is almost impossible for organisations and communities to manually absorb, process, and turn the volume of social media data during crisis into sensible, actionable information (Gao et al. 2011). Tools to automatically identify the type of emergency events reported by citizens (e.g., need shelter, trapped in building) are largely unavailable. Genuine help requests are often difficult to spot, group and validate and many urgent aid requests by individual citizens can go unnoticed.

Current works for event identification from social media data make use of supervised and unsupervised Machine Learning (ML) methods, such as classifiers, clustering and language models (Atefeh and Khreich 2015). Lately, deep learning has emerged as a promising ML technique able to capture high level abstractions in the data, providing significant improvement for various tasks over more traditional ML methods, such as text classification (Kim 2014),

machine translation (Bahdanau et al. 2014; Cho et al. 2014) or sentiment analysis (Tang et al. 2015; Dos Santos and Gatti 2014). Deep learning, in particular, Convolutional Neural Networks (CNNs) have been recently applied with success to social media during crises situations (Caragea et al. 2016; D. T. Nguyen et al. 2017; Burel, Saif, Fernandez, et al. 2017; Burel, Saif, and Alani 2017). However, these approaches have been mostly pursued in academic contexts and have not been made available to the public and social organisations through easily accessible and integrable tools.

Even though creating models for identifying relevant documents, event types and information categories is a major step in the right direction and show the potential of automatic approaches for dealing with a large amount of social media posts during crisis events, its adoption by practitioners is conditioned on their availability and accessibility to the general public.

In this paper, we create three different CNN-based classifiers that allow the identification of crisis-related documents (i.e., *related* vs *unrelated*), event types (e.g., *hurricane*, *floods*, etc.) and information categories (e.g., *reports on affected individuals*, *donations and volunteers*, etc.) using data from the CrisisLexT26 dataset (Olteanu, Castillo, et al. 2014) and make those models available through an easily accessible web service called CREES (Crisis Event Extraction Service).[1] The API and codebase is designed to be easily deployed and integrated into existing tools and has been already integrated into the Ushahidi platform[2] and Google Sheets.[3]

The contribution of this paper can be summarised as follows: 1) The generation of three CNN models that target the problem of event identification in crisis situations (event relatedness, event types and information categories), and; 2) The creation of an easily deployable web API that exposes the previously trained CNN classifiers and its integration into the Ushahidi platform and Google Sheets.

RELATED WORK

Recently, several works have introduced the use of deep learning for event detection (Chen et al. 2015; Feng et al. 2016; T. H. Nguyen and Grishman 2015; Ghaeini et al. 2016; Zeng et al. 2016). Unlike traditional ML feature-based methods, deep learning models do not generally require heavy feature engineering and are therefore less prone to error propagation, caused by using external NLP and text processing tools. Also, deep learning models are more generic and tolerant to domain and context variations than feature-based models, as the former use word embeddings as a more general and richer representation of words (T. H. Nguyen and Grishman 2015).

Pioneer works in this vein include (Chen et al. 2015; Feng et al. 2016; T. H. Nguyen and Grishman 2015). These works address the problem of event detection at the sentence and/or phrase level by first identifying the event triggers in a given sentence (which could be a verb or nominalisation) and classifying them into specific types. For example, the word *"release"* in *"The European Unit will release 20 million euros to Iraq"* is a trigger for the event *"Transfer-Money"*.

Multiple deep learning models have been proposed. For example, Nguyen and Grishman (T. H. Nguyen and Grishman 2015) use a Convolutional Neural Network (CNN) (LeCun et al. 1998) with three input channels, corresponding to word embeddings, word position embeddings and entity type embeddings, to learn a word representation and use it to infer whether a word is an event trigger or not. Chen et al. (Chen et al. 2015) argue that a sentence may contain two or more events and that using a traditional CNN model with a max-pooling layer[4] often leads to the capture of clues of one event in a sentence but to miss the rest. To address this issue, the authors propose using a CNN with a dynamic multi-pooling layer to obtain a maximum value for each part of a sentence and therefore cover more valuable clues of the events within it.

Feng et al. (Feng et al. 2016) use a hybrid neural network model for cross-language event detection. The proposed model incorporates both, a bidirectional LSTM (Bi-LSTM) (Schuster and Paliwal 1997) and CNN component. Bi-LSTM captures the contextual semantics of a given word by means of the preceding and the following information in the text, while CNN is used to capture structure information from the local contexts (e.g., sentence chunks). Results show that the proposed model achieves relatively high and robust performance when applied to data of multiple languages including English, Chinese and Spanish, in comparison with traditional feature-based approaches.

Other works (Caragea et al. 2016; D. T. Nguyen et al. 2017; Burel, Saif, Fernandez, et al. 2017; Burel, Saif, and Alani 2017) have explored directly the application of CNN models to the identification of crisis-related content. Although Caragea et al. (Caragea et al. 2016) explored the dataset used in this paper, they focused on floods event

[1] CREES, http://evhart.github.io/crees.
[2] Ushahidi, http://www.ushahidi.com.
[3] CREES Add-on, https://goo.gl/t73SNY.
[4] In a CNN, a max-pooling layer applies a max operation over the representation of an entire sentence to capture the most useful information.

types and on the informativeness classification rather than event relatedness, event types and information categories. D. T. Nguyen et al.'s work (D. T. Nguyen et al. 2017) investigated information types, they focused only on one of the three tasks investigated in this paper. Nevertheless, their results showed that CNN models can be used successfully for social media classification in crisis situations. Burel, Saif, Fernandez, et al. (Burel, Saif, Fernandez, et al. 2017; Burel, Saif, and Alani 2017) investigated the same dataset as the one investigated in this paper and proposed multiple extension to CNN using semantics. Their results showed some small improvement over conventional CNN. Unfortunately, the proposed approach requires concept extraction tools which tend to not scale well making this approach unsuitable for the development of a web API.

Few works have explicitly targeted the development of automatic machine learning tools and APIs that can be used for automatically labelling social media documents during crises. In general, most tools do not have any automatic classification abilities and rely on simple algorithms or manual work. AIDR (Imran, Castillo, et al. 2014), is one of the very few tools that is explicitly designed for automatically classifying social media documents by providing a platform for training and serving annotation models during crisis situations. Although AIDR shares many similarities to the CREES API presented in this paper and even provide advanced features for retraining machine learning models, contrary to CREES, AIDR relies on traditional machine learning models and requires more effort to be integrated into existing tools and workflows. Finally. it appears that AIDR does not provide any default models whereas CREES is designed to be deployed without any re-training requirements. CrisisNET[5] is another tool and API designed for extracting information in social media document. However, the project has not been updated since 2014 and does not give much information concerning its set of features and the models used.

APPLICATION SCENARIO

During crises, a very large number, sometimes in the millions, of messages are often posted on various social media platforms by using the hashtags dedicated to the crises at hand. However, a good percentage of those messages are irrelevant or uninformative.

Olteanu, Vieweg, et al. observed that crises reports could be classified into three main categories of informativeness; related and informative, related but not informative, and not related (Olteanu, Vieweg, et al. 2015). The percentage of relevant and informative social reports during crises varies a great deal, ranging from 10% in some cases[6] to 65% in others (Sinnappan et al. 2010). However, buried under very many mundane and irrelevant tweets, sometimes one post emerges that needs an urgent response.

In this paper, our goal is to create an automatic tool and the corresponding API that efficiently identifies crisis-related social media messages of sufficient relevance and value. For this purpose, and based on the event types identified by Olteanu, Vieweg, et al. (Olteanu, Vieweg, et al. 2015) and later on investigated by Burel, Saif, Fernandez, et al. (Burel, Saif, Fernandez, et al. 2017), we consider the following three tasks when developing our API:

- *Task1 - Crisis vs. non-crisis related messages*: The goal of this task is to differentiate those posts that are *related to a crisis situation vs. those posts that are not*.
- *Task2 - Type of crisis*: The goal of this task is to identify the different types of crises the message is related to and the potential sub-event linked to a particular crisis. Following the work of (Olteanu, Vieweg, et al. 2015) we consider the following types of natural and human-induced types of crises: *shooting, explosion, building collapse, fires, floods, meteorite fall, haze, bombing, typhoon, crash, earthquake and derailment*.
- *Task3 - Type of information*: the goal of this task is to provide a fine-grained information detection in crises situations. Following the work of (Olteanu, Vieweg, et al. 2015) we consider the following categories of crisis-related information: *affected individuals, infrastructures and utilities, donations and volunteer, caution and advice, sympathy and emotional support, useful information, other*.

The API resulting from training the models on the three previous tasks provides means for: 1) filtering crisis-related content on social media; 2) the identification of event types, and; 3) the identification of information categories. In a real-world use case, such services can be used sequentially in order to gain deep insights concerning a crisis situation.

[5]CrisisNet, http://crisis.net.

[6]Behavioral & Linguistic Analysis of Disaster Tweets, http://irevolution.net/2012/07/18/disaster-tweets-for-situational-awareness.

AUTOMATIC CLASSIFICATION OF CRISIS-RELATED CONTENT ON SOCIAL MEDIA

The first step required for creating the crisis annotation API is to train document classifiers that automatically identify event-related posts, crisis types and information categories. The following sections discuss the CNN model used for training the API models and evaluate the models against multiple baselines in order to validate the accuracy of the API.

Classification of Crisis-related Content using Convolutional Neural Networks (CNN)

Event detection in the context of Twitter is a text classification task where the aim is to identify if a given document (post) describes or is related to an event. In this section, we describe the Convolutional Neural Networks (CNN) model used for performing event detection on Twitter. Similarly to previous work (D. T. Nguyen et al. 2017; Burel, Saif, Fernandez, et al. 2017), the model relies on word embeddings in order to better represent the implicit semantics of the documents and consequently better identify crisis-related documents.

The pipeline of the model consists of three main phases as depicted in Figure 1:

1. *Text Processing*: A collection of input tweets are cleaned and tokenised for later stages;
2. *Word Vector Initialisation*: Given the word tokens produced in the previous stage and pre-trained word embeddings, a matrix of word embedding is constructed to be used for model training;
3. *CNN Training*: In this phase, the CNN model is trained using the word embeddings matrix.

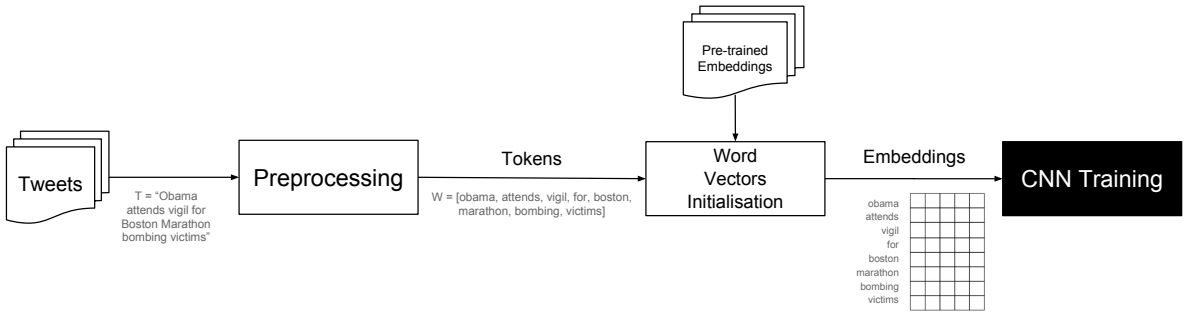

Figure 1. Pipeline of the CNN deep learning model event detection model.

In the following subsections, we describe each of the phases of the pipeline in more detail.

Text Preprocessing

Tweets are usually composed of incomplete, noisy and poorly structured sentences due to the frequent presence of abbreviations, irregular expressions, ill-formed words and non-dictionary terms. This phase, therefore, applies a series of preprocessing steps to reduce the amount of noise in tweets including, for example, the removal of URLs, and all non-ASCII and non-English characters. After that, the processed tweets are tokenized into words that are consequently passed as input to the *word embeddings* phase. Although different methods can be used for preprocessing the input data, in this paper we follow the same approach as Kim (Kim 2014) since it is scalable and does not require any complex algorithms.

Word Vector Initialisation

An important part of applying deep neural networks to text classification is to use word embeddings. As such, this phase aims to initialise a matrix of word embeddings for training the event classification model.

Word embeddings is a general name that refers to a vectorised representation of words, where words are mapped to vectors instead of a one dimension space (Bengio et al. 2003). The main idea is that semantically close words should have a similar vector representation instead of a distinct representation. Different methods have been proposed for generating embeddings such has Word2Vec (Mikolov et al. 2013) and GloVe (Pennington et al. 2014) and they have shown to improve the performance of multiple NLP tasks. Hence, in this work, we choose to bootstrap our model with Google's pre-trained Word2Vec model (Mikolov et al. 2013) to construct our word embeddings matrix, where rows in the matrix represent embeddings vectors of the words in the Twitter dataset. We decide to use this model since it is a commonly used embedding model. In future work, we are planning to experiment with Twitter and crisis-specific embedding models such as the model proposed by Imran, Mitra, et al. (Imran, Mitra, et al. 2016).

CNN Model for Text Classification

This phase aims to train the CNN model from the word embeddings matrix. Below we describe the CNN model along with the proposed training procedure.

As previously discussed, CNN can be used for classifying sentences or documents (Kim 2014). The main idea is to use word embeddings coupled with multiple convolutions of varying sizes that extract important information from a set of words in a given sentence, or a document, and then apply a softmax function that predicts its class.

The proposed CNN model is based on Kim's model (Kim 2014), a CNN model widely used for text classification. It consists of a convolution layer (with three region sizes and multiple filters per region) followed by a max-pooling phase and a fully connected layer where the softmax function is applied for predicting the document classes.

For example, the document $D =$ '*Obama attends vigil for Boston Marathon bombing victims.*' may be tokenised as $T_w = [$'$obama$', '$attends$', '$vigil$', 'for', '$boston$', '$marathon$', '$bombing$', '$victims$'] by a word tokeniser. Then, the embedding D_w is created using the previous document representation T_w and convolutional layers are used followed by a max-pooling step before applying the softmax step that classifies individual documents as depicted in Figure 2.

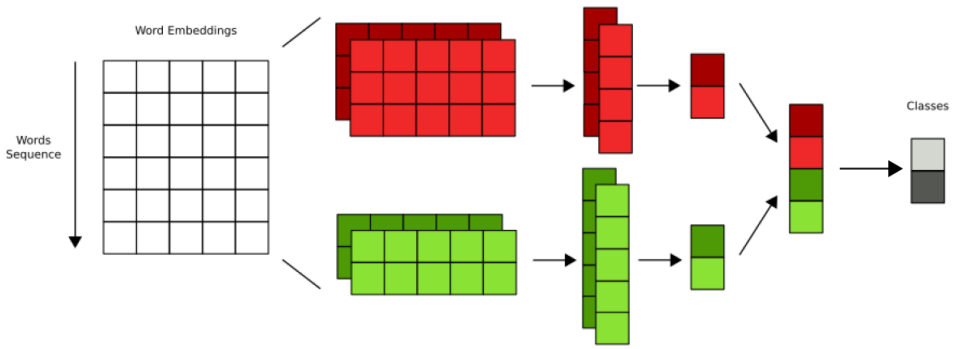

Figure 2. Convolutional Neural Network (CNN) for text classification using word embeddings.

Experimental Setup

Here we present the experimental setup used to assess the event detection model used in the classification API. As previously mentioned, we aim to design an API that supports three different tasks. As such, our evaluation setup requires the selection of (i) Twitter datasets, (ii) the semantic extraction tool, and (iii) baseline models for cross-comparison.

Dataset

To assess the performance of the event detection model we require the use of datasets where each tweet is annotated with: whether or not it relates to a crisis event, the type of crisis (e.g., *earthquake*, *flood*, etc.) and the type of information (e.g., *affected individuals*, *infrastructures*, etc.). For the purpose of this work, we use the CrisisLexT26 dataset (Olteanu, Castillo, et al. 2014).

CrisisLexT26 includes tweets collected during 26 crisis events in 2012 and 2013. Each crisis contains around 1,000 annotated tweets for a total of around 28,000 tweets with labels that indicate if a tweet is related or unrelated to a crisis event (i.e. *related/unrelated*, Task 1)

For the second task, we need a list of crisis types. In order to obtain such information, we consider that the annotated tweets that are from the same sub-collection belong to the same type of event. Using this approach we obtain 12 different crisis types (*shooting, explosion, building collapse, fires, floods, meteorite fall, haze, bombing, typhoon, crash, earthquake and derailment*) (Task 2).

The CrisisLextT26 tweets are also annotated with additional labels indicating the type of information present in the tweet (*affected individuals, infrastructures and utilities, donations and volunteer, caution and advice, sympathy and emotional support, and useful information and unknown*, Task 3). More information about the CrisisLexT26 dataset can be found on the CrisisLex website.[7]

[7]CrisisLex T26 Dataset, http://www.crisislex.org/data-collections.html#CrisisLexT26.

Table 1. Event detection performance of baselines and the CNN model under the three evaluation tasks on the full and undersampled datasets (PT-Embed: Pre-trained word embeddings).

			Related/Unrelated			Event Types			Information Types		
Model	Data	Features	*P*	*R*	*F*1	*P*	*R*	*F*1	*P*	*R*	*F*1
Naive Bayes	*Full*	TF-IDF	0.846	0.684	0.733	0.941	0.927	0.933	0.600	0.570	0.579
CART	*Full*	TF-IDF	0.742	0.707	0.723	0.992	0.992	0.992	0.506	0.491	0.497
SVM	*Full*	TF-IDF	0.870	0.738	0.785	0.997	0.996	0.997	0.642	0.604	0.616
CNN	*Full*	PT-Embed	0.861	0.744	0.797	0.991	0.986	0.988	0.634	0.590	0.609
Naive Bayes	*Sample*	TF-IDF	0.795	0.787	0.785	0.929	0.928	0.928	0.558	0.563	0.556
CART	*Sample*	TF-IDF	0.770	0.769	0.769	0.988	0.988	0.988	0.471	0.464	0.464
SVM	*Sample*	TF-IDF	0.833	0.830	0.829	0.995	0.995	0.995	0.606	0.609	0.605
CNN	*Sample*	PT-Embed	0.839	0.838	0.838	0.983	0.983	0.983	0.610	0.610	0.610

Since the annotations tend to be unbalanced, we also create a balanced version of the dataset for each task by performing biased random undersampling using tweets from each sub-collection. As a result, the first task dataset is reduced to 6703 tweets (24%), the second task to 12997 tweets (46.5%) and the final task to 9105 tweets (32.6%). The balanced datasets are created in order to avoid an overfitted model that does not generalise well in future event predictions where the data distribution can differ from the datasets used for training the models.

Evaluation

In this section, we report the results obtained from using the CNN model for crisis event detection of tweets under three evaluation tasks: (Task1) Crisis vs. non-crisis related tweets, (Task2) type of crisis, and (Task3) information categories. Our baselines of comparison are three traditional machine learning classifiers: Naive Bayes, Classification and Regression Trees (CART), and SVM with RBF kernels trained from words unigrams. We initialise the CNN models with the Google News 3 million words and phrases pre-trained word embeddings data.[8] Results for all experiments are computed using 5-fold cross-validation. For each task, we perform the evaluation on the full and undersampled versions of the dataset.

We train the CNN model using 300 long word embeddings vectors with $F_n = 128$ convolutional filter of sizes $F_s = [3, 4, 5]$. For avoiding over-fitting, we use a dropout of 0.5 during training and use the ADAM gradient descent algorithm (Kingma and Ba 2014). We perform 400 iterations with a batch size of 256.

Table 1 shows the results of our event detection classifiers for the three evaluation tasks on the full and undersampled versions of the dataset. In particular, the table reports the precision (P), recall (R), and F1-measure (F1) for each evaluation task and model. The table also reports the types of features and embeddings used to train the different classifiers.

Baselines Results

As seen in Table 1, the results for each task and each baseline show that the first two tasks are relatively easy to predict whereas predicting information types is much more complex. In general, we also observe that SVM is the best performing algorithm followed by CART and Naive Bayes. For the first two tasks with the full data, each method achieve *precision*, *recall* and $F_1 > 0.72$ and SVM appears to be the best model with $F_1 = 0.785$ for identifying crisis related tweets and $F_1 = 0.997$ for identifying event types. The task of identifying information types show much lower F_1 across the board. This is probably due to the fact that compared to the previous tasks, information types probably contain much more general terms in each class. Similarly to the previous tasks, SVM performs the best with $F_1 = 0.616$.

With the balanced datasets, the results are similar. However, the predictions for the first task increase by around +4.8%. This result is likely due to the fact that the first task was the most imbalanced task and benefits the most from the undersampling process.

The high precision and recall results observed for the second task ($F_1 = 0.997$) suggests that the different models overfit the data. The issue is not resolved by undersampling the data with an F_1 of 0.995. Looking at the data in more details, we observe that each category contains very clear category indicators. For instance, 77% of the tweets about *meteorite falls* contain the word *meteor*, whereas 76.2% of the tweets about *explosions* contains the word *Boston*. In order to reduce such issue, we could, for instance, remove some of these words from the dataset so the models become less tied to practical event instances (e.g the Boston bombings).

[8]Google Word2Vec, https://code.google.com/archive/p/word2vec

CNN Results

In general, applying CNNs with pre-trained word embeddings (*PT-Embed*) for both the full and undersampled data does not improve significantly over SVM. Using the full dataset, we obtain an F_1 of 0.797 for the crises related tweets and full dataset, 0.988 for event types detection and 0.616 for information type identification.

When using the undersampled datasets, the results are similar to the previous observations with an increase of +3.7% in F_1 for the first task. There is also a slight improvement for the last task with +0.6% in F_1.

Even though CNN models marginally improve over the baselines, word embeddings have the advantage to be less strict than the bag of words model. This means that when new vocabulary is added, the CNN model may be less affected by out of vocabulary words (e.g., by using pre-trained embeddings).

Discussion

Similar to previous work (D. T. Nguyen et al. 2017; Burel, Saif, Fernandez, et al. 2017; Burel, Saif, and Alani 2017), we used deep learning CNN models for detecting events on Twitter. This section discusses the limitations of the presented work as well as different areas of future investigations.

We experimented with three event detection tasks and observed that identifying crisis-related events and event types in tweets (i.e., Task 1 and Task 2) with high accuracy appears to be a relatively easy task that can be fulfilled well with both traditional models such as SVM and CNN models. Identifying the types of information provided in crisis related tweets (Task 3) is much more challenging as tweets mentioning event information types tend to contain much more general terms in each class than the tweets that are related or unrelated to crises or are discussing different types of events.

Looking into the details of the second task, we observed that for this task, the models were generally overfitted even after balancing the data. The reason seems to be associated with the presence of very clear category indicators (e.g., place names). In order to reduce such an issue, we could remove place names from training instances or try to collect additional data so that the associations between event types and locations is reduced.For instance we could consider the CrisisNLP datasets [9].

Although comparing our results with existing works is not straightforward due to differences in the used datasets and experimental setup, our results appear to be similar to previous observations (D. T. Nguyen et al. 2017; Burel, Saif, Fernandez, et al. 2017; Burel, Saif, and Alani 2017). As future work, we are considering using more complex preprocessing methods as well as augmenting the CrisisLexT26 dataset with the CrisisNLP datasets similarly to the work done by D. T. Nguyen et al. (D. T. Nguyen et al. 2017). We also plan to improve the CNN model by adding additional convolutional layers and performing parameter optimisation. For instance, we could try to improve the results by modifying the size of the model filters as well as the number of filters. We could also increase and optimise the number of training steps in order to obtain better results. Besides CNN, other more suitable models such as recurrent neural networks (RNN) (Graves 2012) may be also considered as future work due to their ability to capture textual relations more accurately.

THE CRISIS EVENT EXTRACTION SERVICE (CREES)

In order to be able to use the previous models, it is necessary to create an easily deployable API that can be integrated into multiple tools and software. In this section we introduce the Crisis Event Extraction Service (CREES)[10] as a simple REST API for automatically identifying crisis-related documents.

The CREES API Server

The CREES server provides a simple web API that can be queried using standard HTTP POST and GET queries. The API accepts textual documents as input and returns standardised JSON objects that contain the annotations returned by the CNN model presented in the previous section. The following subsections describe the API in more details and show how the API design choices help its integration into third-party tools.

[9]CrisisNLP, http://crisisnlp.qcri.org/.
[10]CREES API, http://evhart.github.io/crees.

API Description and Services

The CREES API exposes three different services that follow the tasks described in the previous sections. Each method can be accessed using a GET query with a text parameter that contains the document that needs to be labelled. CREES also provide POST queries that can be used for annotating more than one document by submitting a JSON array that contains multiple textual documents to the endpoint. The methods are the following:

- */events/eventRelated*: Determines if a document is related to a crisis situation. The following labels are returned: *non-related*, *related* (Task 1).
- */events/eventType*: Determines the type of crisis discussed in a document. The following labels are returned: *bombings*, *collapse*, *crash*, *derailment*, *earthquake*, *explosion*, *fire*, *floods*, *haze*, *meteorite*, *none*, *shootings*, *typhoon* and *wildfire* (Task 2).
- */events/infoType*: Determines the type of information discussed in a document. The following labels are returned: *affected_individuals*, *caution_and_advice*, *donations_and_volunteering*, *infrastructure_and_utilities*, *not_applicable*, *not_labeled*, *other_useful_information* and *sympathy_and_support* (Task 3).

Each method returns a similar JSON object. For example the following CURL[11] query:

```
curl -G http://127.0.0.1/events/infoType  \
--data-urlencode 'text=If you are evacuating please dont wait, \
take your pets when you evacuate'
```

Returns the following JSON object:

```
{
    "classifier": "CNN",
    "input": "if you are evacuating please dont wait, take your pets when you evacuate ",
    "label": "caution_and_advice",
    "version": 0.3
}
```

Although the GET method only accepts one document as input, the API also allows POST queries that can be used in order to annotate more than one document by submitting a JSON array containing a list of documents to annotate. Each method returns a similar JSON object. For example:

```
curl -X POST http://127.0.0.1/events/eventRelated  --header \
'Content-Type: application/json' -d '["If you are evacuating, \
"take your pets when you evacuate", "AAPL, NBA playoffs 2013, \
New York Post, West Texas, ..."]'
```

Returns the following JSON object:

```
{
    "labels": [
    {
        "input": "If you are evacuating, take your pets when you evacuate",
        "label": "related"
    },
    {
        "input": "AAPL, NBA playoffs 2013, New York Post, West Texas, ...",
        "label": "non-related"
    }
    ],
    "classifier": "CNN",
    "version": 0.3
}
```

[11] CURL, http://curl.haxx.se.

CREES is built on top of Flask[12] and Tensorflow[13] and exposes an OpenAPI v2[14] compliant endpoint that can be used for helping the automatic consumption of the CREES services. The API serves the aforementioned CNN models and integrates Swagger UI[15] so the API can be used directly in a web browser.

Installation and Integration

Besides the ability to handle multiple classification tasks concurrently, the development of the CREES server was driven by two key requirements. First, the API needed to be easy to understand and integrate into existing tools. Second, the CREES server needed to be easy to download install and deploy.

CREES has been integrated into the Ushahidi platform[16] as part of the COMRADES[17] project using the Ushahidi webhooks API. Ushahidi is a crowd-sourcing platform designed for collecting data from multiple resources such as social media, SMS and email. The platform helps the visualisation and classification of posts in order to better understand crisis situations by allowing users to manually map, annotate and classify documents. The CREES integration allows the automatic categorisation of Ushahidi posts as soon as they are added to the platform and as result reduces the human resources required for classifying incoming documents.

We also created a Google Sheets add-on[18] that exposes the CREES classifiers to Google Spreadsheet users since spreadsheet software is often used by volunteers during crises. The add-on exposes each of the three classifiers as a spreadsheet function (i.e., `CREES_RELATED`, `CREES_EVENT` and `CREES_INFO`) that can be used for annotating individual table cells or columns.

In order to allow the fast deployment of the CREES server during crises, the CNN models and code are both available online and can be deployed in a few minutes using Docker and a step-by-step documentation. Another specificity of CREES is that it is not deeply integrated into a specific ecosystem. This simplifies the integration of the annotation services into multiple products as the Ushahidi and Google Sheets integrations demonstrate.

Load Testing

Although the current version of CREES is not explicitly optimised for scalability, the API is built on top of Flask so multiple servers can be used for distributing user load when a large amount of data needs to be analysed concurrently. Nevertheless, we load test the CREES API in order to better understand the current strengths and limitations of the deployed server implementation. We conduct a traditional load balancing test by progressively increasing the number of API users and observing how the server handles the requests.

Although in principle, more than 6000 tweets are created every second, the standard (*free*) Twitter API only allows for the access of at most 20 tweets per second (180 queries that returns 100 tweets per 15 minutes windows). As a consequence, a realistic usage of the CREES API is to consider that each user will submit at most 12 GET query per second to each CREES annotation service every second (i.e., 12 queries per second to CREES). In this context, we decide to load test the CREES API by considering that each user performs 12 queries per second.

We load test CREES using Locust[19] by increasing the number of users every second by 10 until we have 1000 users that perform simultaneous server queries using the previously mentioned 12 GET queries per second. We report the number of average requests performed per second on the endpoint as well as the number of failed requests. We also report the server latency in order to evaluate the responsiveness of the API. In order to have a fairer evaluation, we perform the evaluation outside the network domain where the server is hosted.

Results

The evaluation results are listed in Table 2. In general, we can observe that each model performs roughly similarly and is able to perform between 77 and 82 queries per second with a median latency of 700 milliseconds. Out of the 1698 requests submitted to the CREES API, less than 5% of the calls fail. When performing the same test with at most 500 users, we observe no failing requests.

In general, the results show that the CREES API is able to serve 500 users concurrently without any issues and to some extent 1000 users successfully when each user perform more than 12 queries every second to the CREES API.

[12]Flask, http://flask.pocoo.org.
[13]Tensorflow, http://www.tensorflow.org.
[14]Open API, http://www.openapis.org.
[15]Swagger UI, https://swagger.io/swagger-ui/.
[16]Ushahidi, http://www.ushahidi.com.
[17]COMRADES, http://www.comrades-project.eu.
[18]CREES Add-on, https://goo.gl/t73SNY.
[19]Locust, http://locust.io.

Table 2. Load test of the CREES API using 1000 users performing 12 API calls per second with an hatching rate of 20 users per second.

Query	Nb Reqs	Nb Fails	Avg	Min	Max	Median	Req/S
GET /events/eventRelated	573	32 (5.29%)	1159	361	63845	700	82.60
GET /events/eventType	559	22 (3.79%)	1041	372	63210	690	79.40
GET /events/infoType	566	22 (3.74%)	1024	392	58950	700	77.80
Total	1698	76 (4.48%)					239.80

It is important to note that it is relatively unlikely that in a real use case more than 500 users or organisation will be accessing CREES concurrently. Moreover, if the users use the batch CREES API (POST), they need to only make 0.6 query per seconds to the API since they can submit multiple posts to the API using only one call. In this context, the CREES API can potentially handle much more connections if the clients use less aggressive methods for querying the API.

Discussion

Although the current CREES API requirements are principally focused on providing a simple API that can be integrated into existing tools and workflows, we showed that the CREES API can already support realistic query loads (i.e. 500 users performing 12 query per second on the API endpoint concurrently). Moreover, if necessary, the API efficiency can be easily improved by using multiple CREES servers together. Finally, we also showed that CREES can be easily integrated into third-party tools such as Ushahidi and Google Sheets.

Compared to other tools that use machine learning methods for helping the analysis of social media documents during crises, CREES is designed to be lightweight in order to easily integrate to already existing platforms and workflows such as the Ushahidi platform or Spreadsheet software. This approach can be directly contrasted to more integrated tools such as AIDR (Imran, Castillo, et al. 2014) that provide end-to-end annotation heavyweight features. Compared to AIDR, CREES models are also backed by CNN models making them potentially more accurate compared to the classifiers used by AIDR.

Even though improving the underlying CNN models can lead to better predictions, the CREES API can be improved in multiple ways in order to make it more robust and scalable. At the moment, CREES does not support rate-limiting and user authentication as well as advance caching features. Adding those features would allow for a more robust API. Another line of research could be the integration of methods for updating the trained CNN models through the API. However, adding such feature may increase the integration complexity of CREES.

CONCLUSIONS AND FUTURE WORK

In this paper, we introduced CREES (Crisis Event Extraction Service), a web API that allows the automatic classification of crisis-related social media documents. The API provides services for identifying crisis-related documents, event types and information categories. Compared to existing tools, CREES is a lightweight API that can be easily integrated into existing tools and workflows. In order to demonstrate the versatility of CREES, we integrated the API into the Ushahidi platform and exposed its features as a Google Sheet add-on.

CREES annotation features are backed-up by CNN models trained on a Twitter dataset consisting of 26 different crisis events. The models were tested evaluated. Results show that CNNs are able to successfully identify the existence of events, and event types with > 79% F-measure, but the performance of CNN significantly drops (61% F-measure) when identifying fine-grained event-related information. These results are competitive with more traditional Machine Learning models, such as SVM.

Although multiple improvements can be done in order to improve the CNN models used by CREES such as using additional training datasets (e.g., CrisisNLP) or alternative deep learning models (e.g., RNN). We plan to focus future work on adding more features to the CREES API such as user authentication mechanisms, rate-limiting features and better caching support. We are also working on evaluating the Ushahidi and Google Sheets integration with users as part of a crisis response exercise.

ACKNOWLEDGMENT

This work has received support from the European Union's Horizon 2020 research and innovation programme under grant agreement No 687847 (COMRADES).

REFERENCES

Atefeh, F. and Khreich, W. (2015). "A survey of techniques for event detection in Twitter". In: *Computational Intelligence* 31.1, pp. 132–164.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473*.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). "A neural probabilistic language model". In: *Journal of machine learning research* 3.Feb, pp. 1137–1155.

Bercovici, J. (2012). "Why Time Magazine Used Instagram To Cover Hurricane Sandy". In: *Obtenido de http://www. forbes. com/sites/jeffbercovici/2012/11/01/why-time-magazine-used-instagram-to-cover-hurricane-sandy*.

Burel, G., Saif, H., and Alani, H. (2017). "Semantic Wide and Deep Learning for Detecting Crisis-Information Categories on Social Media". In: *The Semantic Web ? ISWC 2017*.

Burel, G., Saif, H., Fernandez, M., and Alani, H. (2017). "On Semantics and Deep Learning for Event Detection in Crisis Situations". In: *Workshop on Semantic Deep Learning (SemDeep), at ESWC 2017*.

Campanella, T. J. (2006). "Urban resilience and the recovery of New Orleans". In: *Journal of the American Planning Association* 72.2, pp. 141–146.

Caragea, C., Silvescu, A., and Tapia, A. H. (2016). "Identifying Informative Messages in Disasters using Convolutional Neural Networks". In: *13th Proceedings of the International Conference on Information Systems for Crisis Response and Management, Rio de Janeiro, Brasil, May 22-25, 2016.*

Chen, Y., Xu, L., Liu, K., Zeng, D., and Zhao, J. (2015). "Event Extraction via Dynamic Multi-Pooling Convolutional Neural Networks." In: *ACL (1)*, pp. 167–176.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*.

Dos Santos, C. N. and Gatti, M. (2014). "Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts." In: *COLING*, pp. 69–78.

Feng, X., Huang, L., Tang, D., Qin, B., Ji, H., and Liu, T. (2016). "A language-independent neural network for event detection". In: *The 54th Annual Meeting of the Association for Computational Linguistics*, p. 66.

Gao, H., Barbier, G., and Goolsby, R. (2011). "Harnessing the crowdsourcing power of social media for disaster relief". In: *IEEE Intelligent Systems* 26.3, pp. 10–14.

Ghaeini, R., Fern, X. Z., Huang, L., and Tadepalli, P. (2016). "Event nugget detection with forward-backward recurrent neural networks". In: *The 54th Annual Meeting of the Association for Computational Linguistics*, p. 369.

Graves, A. (2012). "Supervised sequence labelling". In: *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, pp. 5–13.

Imran, M., Castillo, C., Lucas, J., Meier, P., and Vieweg, S. (2014). "AIDR: Artificial intelligence for disaster response". In: *Proceedings of the 23rd International Conference on World Wide Web*. ACM, pp. 159–162.

Imran, M., Mitra, P., and Castillo, C. (2016). "Twitter as a lifeline: Human-annotated twitter corpora for NLP of crisis-related messages". In: *arXiv preprint arXiv:1605.05894*.

Kim, Y. (2014). "Convolutional neural networks for sentence classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*.

Kingma, D. and Ba, J. (2014). "Adam: A method for stochastic optimization". In: *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781*.

Nguyen, D. T., Al-Mannai, K., Joty, S. R., Sajjad, H., Imran, M., and Mitra, P. (2017). "Robust Classification of Crisis-Related Data on Social Networks Using Convolutional Neural Networks." In: *ICWSM*, pp. 632–635.

Nguyen, T. H. and Grishman, R. (2015). "Event Detection and Domain Adaptation with Convolutional Neural Networks." In: *ACL (2)*, pp. 365–371.

Olteanu, A., Castillo, C., Diaz, F., and Vieweg, S. (2014). "CrisisLex: A Lexicon for Collecting and Filtering Microblogged Communications in Crises." In: *ICWSM*.

Olteanu, A., Vieweg, S., and Castillo, C. (2015). "What to expect when the unexpected happens: Social media communications across crises". In: *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM, pp. 994–1009.

Pennington, J., Socher, R., and Manning, C. D. (2014). "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543.

Qu, Y., Huang, C., Zhang, P., and Zhang, J. (2011). "Microblogging after a major disaster in China: a case study of the 2010 Yushu earthquake". In: *Proceedings of the ACM 2011 conference on Computer supported cooperative work*. ACM, pp. 25–34.

Schuster, M. and Paliwal, K. K. (1997). "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681.

Sinnappan, S., Farrell, C., and Stewart, E. (2010). "Priceless tweets! A study on Twitter messages posted during crisis: Black Saturday". In: *ACIS 2010 Proceedings* 39.

Starbird, K., Palen, L., Hughes, A. L., and Vieweg, S. (2010). "Chatter on the red: what hazards threat reveals about the social life of microblogged information". In: *Proceedings of the 2010 ACM conference on Computer supported cooperative work*. ACM, pp. 241–250.

Tang, D., Qin, B., and Liu, T. (2015). "Document Modeling with Gated Recurrent Neural Network for Sentiment Classification." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pp. 1422–1432.

Thomson, R., Ito, N., Suda, H., Lin, F., Liu, Y., Hayasaka, R., Isochi, R., and Wang, Z. (2012). "Trusting tweets: The Fukushima disaster and information source credibility on Twitter". In: *Proceedings of the 9th International ISCRAM Conference*, pp. 1–10.

Vieweg, S., Hughes, A. L., Starbird, K., and Palen, L. (2010). "Microblogging during two natural hazards events: what twitter may contribute to situational awareness". In: *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, pp. 1079–1088.

Zeng, Y., Yang, H., Feng, Y., Wang, Z., and Zhao, D. (2016). "A convolution BiLSTM neural network model for Chinese event extraction". In: *International Conference on Computer Processing of Oriental Languages*. Springer, pp. 275–287.