International Journal of Disaster Risk Reduction 51 (2020) 101757

Contents lists available at ScienceDirect

International Journal of Disaster Risk Reduction

journal homepage: http://www.elsevier.com/locate/ijdrr

Machine-learning methods for identifying social media-based requests for urgent help during hurricanes

Ashwin Devaraj, Dhiraj Murthy [*], Aman Dontula

The University of Texas at Austin, Austin, TX, USA

ARTICLE INFO

Keywords:
Text classification
Deep learning
Social media analysis
Disaster response

2010 MSC:
00-01
99-00

ABSTRACT

Social media is increasingly used by people during large-scale natural disasters to request emergency help. Previous work has had success in applying machine-learning classifiers to detect tweets in coarse-grained categories, such as disaster type and relevance. However, there is a dearth of work that focuses on detecting tweets containing requests for help that are actionable by first responders. Using over 5 million tweets posted during 2017's Hurricane Harvey in Houston, U.S., we show that though such requests are uncommon, their often life-or-death nature justifies the development of tweet classifiers to detect them. We find that the best-performing classifiers are a convolutional neural network (CNN) trained on word embeddings, support vector machine (SVM) trained on average word embeddings, and multilayer perceptron (MLP) trained on a combination of unigrams and part-of-speech (POS) tags. These models achieve F1 scores of over 0.86, confirming their efficacy in detecting urgent tweets. We highlight the utility of average word embeddings for training non-neural models, and that such features produce results competitive with more traditional n-gram and POS features.

1. Introduction

Social media is used during various types of disasters to disseminate relevant information to a large audience [1]. Some of the reasons for communicating on social media are to raise awareness, express sympathies, discuss causes, assign blame, and offer/request assistance [2]. Another important, but severely understudied, category of tweets posted during natural disasters is truly urgent calls for help that indicate life-threatening situations [3,4]. According to Glass [5], first-responders often arrive too late in emergency situations affecting multiple sites due to the overwhelming volume of calls and the consequent difficulty in coordinating an emergency response. During Hurricane Harvey in 2017, for example, many people were rescued not by first-responders but by fellow citizens responding to requests for help on social media [6]. Developing classifiers capable of distinguishing requests of help on Twitter would thus aid in the emergency response to large-scale disasters by allowing the public to better identify and assist those who need urgent help.

Numerous studies have been conducted regarding the classification of tweets into categories including offensiveness and sentiment using machine learning methods. Some have explored the use of traditional text features used in natural language processing (NLP), including n-grams, lexicon features, and part-of-speech (POS) tags [7], while others have incorporated features from topic models generated using statistical techniques like Latent Dirichlet Allocation (LDA) [8] to build classifiers [9]. Recent studies have strongly skewed towards deep learning methods, specifically through the use of deep convolutional neural networks (CNNs) [10–12] and recurrent neural networks (RNNs) [13].

Machine-learning classification tasks conducted on disaster-related tweets have, for the most part, addressed whether tweets are about crises or not [14], the type of disaster [14], the type of information being conveyed [14,15], informativeness [16], and general sentiment polarity [17]. Though these labeling schemes could help first responders cherry-pick useful tweets from a large dataset, none of them tackle the problem of specifically detecting urgent tweets that should elicit a first response (i.e., calls for help or reports of specific people requiring help from first responders).

The closest any labeling scheme comes to making the distinction between urgent and non-urgent tweets is by Imran et al. [15], in which the "Injured or dead people" and "Missing, trapped, or found people" categories include tweets of people requesting help or reporting that specific people need help. However, even these categories are too broad to be directly useful to first responders since they include tweets of evacuations and rescues after the fact and do not exclude news sources.

* Corresponding author.
E-mail addresses: ashwin.devaraj@utexas.edu (A. Devaraj), dhiraj.murthy@austin.utexas.edu (D. Murthy), adontula@utexas.edu (A. Dontula).

https://doi.org/10.1016/j.ijdrr.2020.101757
Received 8 March 2020; Received in revised form 1 July 2020; Accepted 2 July 2020
Available online 20 July 2020

In this study, we start to fill this gap by building classifiers specifically designed to determine whether tweets posted during 2017's Hurricane Harvey contain information actionable by first responders. This includes only personal requests for help and reports of other people who need help. In both cases an address or location is specified. Using a subset of 2474 tweets from our original dataset, we build models using both traditional text features used in natural language processing (NLP), including n-grams and POS tags, as well as word embeddings (all terms we explain later in the paper).

Word embeddings are a representation of words (and tokens in general, as elaborated in the Methods section) as vectors such that words with similar semantic meanings have vectors with a small angle of separation between them [18–20]. Each dimension of an embedding represents some semantic attribute of the associated word, though what exactly that attribute is may not be interpretable by humans. The main benefit of word embeddings is that they allow machine-learning models to generalize well to texts with words not seen in the training dataset, since if those new words have similar meanings to previously-seen words, their embeddings have values close to those of the embeddings of the seen words.

In the domain of text classification, word embeddings have traditionally been used as inputs to deep CNNs, wherein the text is input as a matrix of word embeddings [10,11,14]. Such an input format allows CNNs to use the ordering of words to make classification decisions, something that traditional NLP features like n-grams do not permit. To our knowledge, the application of word embeddings to non-convolutional text classification has not been explored in the literature. In this study we investigate the performance of non-convolutional classifiers trained on average word embeddings (i.e., the average of the embeddings corresponding to words in a tweet). Although average embeddings do not preserve the identities of every word in a tweet or their ordering, we hypothesize that since tweets are relatively short (at most 280 characters long), averaging does not discard too much information about the content of the tweets, and each dimension of the average vector effectively summarizes that respective semantic attribute of the entire tweet.

One challenging trait of our dataset is that only a small fraction (about 7%) of it consists of urgent tweets, probably owing to the fact that requests for help are infrequent in comparison to general tweets about the disaster, such as comments, reactions, and requests for donations. Imbalanced datasets can be a problem when building classifiers as the classifiers may naively classify every input as a member of the majority class to achieve high accuracy. To alleviate this problem, we experimented with oversampling, in which urgent tweets were artificially duplicated in the training dataset to prevent models from naively classifying them as not urgent.

In this study, we consider whether it is possible to successfully extract information useful to first responders from public tweets during a hurricane. Previous work has refined the process of classifying disaster-related tweets into categories [7,9,14–16]. However, our study is novel in its explicit focus on classifying tweets from people actively asking for help and those who provide enough information for first responders to act on.

2. Related work

Extensive prior research has been done on disaster-related tweet classification and more general sentence classification using both traditional textual features like n-grams, POS tags, and LDA topic models [7,9,21], as well as the more recently-developed word embedding features [10,11,19,20].

Kouloumpis et al. [7] conducted an exhaustive study analyzing the relative efficacy of n-gram, POS, and various binary features (such as the existence of abbreviations and words in all caps) in tweet sentiment classification and concluded that a combination of n-gram and binary features yields the best AdaBoost [22] classifiers.

Among previous work on disaster-related tweet classification, Ashktorab et al. [21] used unigram features to build a logistic regression model to determine whether tweets from 12 different natural disasters referenced infrastructural damage or human casualties. Imran et al. [23] used a naive Bayes classifier trained on unigram, bigram, POS, and binary features to classify tweets from the 2011 tornado in Joplin, Missouri as informative or not (achieving an F1 score of 0.78). They then built a classifier to categorize the informative tweets as "caution," "donation," "advice," or "information source."

Imran et al. [15] built naive Bayes, random forest, and SVM models to classify tweets from 19 different natural disasters (e.g., floods and earthquakes) into different information types. These categories include information about injured or deceased individuals, displaced individuals, sympathy and emotional support, and irrelevant topics.

A major shortcoming of n-gram and binary features is that they are too sparse and noisy for certain tasks like profanity classification [9]. Xiang et al. [9] took a different approach for profanity classification using Latent Dirichlet Allocation [8] to generate topic features. They built a topic model from a large dataset of tweets (collected from Twitter users with very high and low records of using profanity) and then created a feature vector for each tweet by concatenating its probability distribution over the generated topics and binary features indicating the presence of offensive words in a dictionary. Their logistic regression model outperformed models using only keyword features by 6% with respect to true positive rate while maintaining false positive rates below 4%.

Although LDA produces features that are less sparse than n-gram and binary features, it is still a bag-of-words model that treats each word as an independent entity, with no notion of semantic distance between words. Bengio et al. [18] developed a way to jointly learn a vector representation of words in a dictionary (i.e., the word embeddings) and a model that uses these embeddings to generate a probability distribution of the next word in a sentence given n previous words. These embeddings are a compact representation of a word and the distances between them correspond to the semantic distances between the associated words. Mikolov et al. [19] and Pennington et al. [20] extended these methods to efficiently train better-performing word embeddings.

More recently, research in short text classification has emphasized the use of deep neural networks trained on word embeddings to drastically improve model performance [10,11,14]. One such class of deep neural networks are CNNs, which are especially well-suited for spatially-organized data like image pixels, since their convolution layers apply a fixed filter across inputs in a manner inspired by biological visual neurons in living organisms [12]. Kim [11] developed a simple and popular CNN architecture for sentence classification, and Dos Santos and Gatti [10] developed a CNN trained on both word and character-level embeddings for the same task. Zhang et al. [24] provide a detailed guide on how to tune Kim's hyperparameters, including reasonable feature ranges to grid search over and sensitivity analysis. All of these CNN networks take as input a matrix of embeddings dependent on the word order in tweets (see the Methods section for a more detailed discussion).

Recent work in text classification has also explored the use of RNNs, which are neural networks containing recurrent layers whose outputs depend on their own outputs from the previous time step. For example, Lai et al. [13] developed a recurrent analog of the CNN that outperformed Kim's CNN model in text classification on 3 different datasets. Hassan et al.'s [25] approach is inspired by Kim's [11] framework and replaces the max-pooling layer after the convolutional one with a bi-directional long short-term memory (LSTM) module [26], which is a type of RNN with additional gating parameters that control the relative contributions of the current input versus inputs from previous time steps to stabilize model training. Hassan et al.'s [25] use of an LSTM allows their model to take as input tweets of arbitrary length and to capture distant semantic dependencies in sentences. Their model slightly outperforms Kim's CNN [11] on the Stanford Sentiment Treebank dataset

[27] in both fine-grained and binary classification accuracy.

In the domain of disaster tweet classification, Burel et al. [14] used Kim's [11] architecture to accurately classify disaster-related tweets from the CrisisLexT26 dataset [28] based on relevance, type of disaster, and type of information conveyed [31]. Nguyen et al. [49] used a CNN architecture similar to Kim [11] trained on custom word embeddings generated from a crisis-specific corpus to classify tweets from (CrisisNLP, CrisisLex, and Artificial Intelligence for Disaster Response (AIDR))[1]the CrisisNLP dataset [15] based on informativeness and type of information conveyed, achieving high area under curve (AUC) scores (explained in the Machine-learning Models portion of the Methods section).

Prior work has focused on the classification of disaster tweets into different information types [14,15] but has not dealt with the task of detecting tweets with specific information about people requiring emergency assistance from first responders. In other words, while useful for separating tweets relevant to the disaster from the vast majority of unrelated tweets, previous work does not focus on classifying tweets from individuals asking for help. For example, Burel et al.'s "affected individuals" category [14]—inspired by the categories in the exploratory study by Olteanu et al. [31]—and Imran et al.'s "missing, trapped, or found people" category [15] both include tweets that contain specific information about people needing rescue. However, they do not make a distinction between the presence of requests for help/ongoing crises and reports of situations that have already been resolved.

3. Research questions

Our study investigates the performance of both non-convolutional and convolutional machine-learning classifiers on the task of determining whether tweets posted during Hurricane Harvey contain current information about people requesting emergency assistance that first responders can act upon. Although prior work does include categories encompassing these urgent tweets, like "affected individuals" and "missing, trapped, or found people," existing work does not acknowledge urgent tweets as a separate category [14,15]. The lack of existing classifiers focused on detecting these urgent tweets motivates the following research questions.

RQ1: How prevalent are requests for urgent help on Twitter during natural disasters like Hurricane Harvey?
RQ2: How well do both non-convolutional and convolutional machine learning models trained on a dataset of tweets posted during Hurricane Harvey perform as urgent tweet classifiers?

We pursue this second research question since an initial manual inspection of our dataset indicates that the vast majority of tweets do not contain specific requests for help that would be useful to first responders. Therefore, there is a clear need to evaluate whether we can create machine learning models that can successfully detect urgent tweets from among the millions posted during a disaster, rather than leaving this time-intensive task to human labelers.

RQ3: How does the performance of non-convolutional urgent tweet classifiers trained on average word embeddings compare to that of non-convolutional classifiers trained on traditional textual features like n-grams?

We explore this question because prior work using average embeddings as features in text classification tasks was not identified. Therefore, we evaluate whether our models perform well with these features despite the loss of word ordering information. If so, average embeddings could be another feature type that researchers consider when building disaster-related text classifiers.

[1] See Alam et al. [30] for a more detailed discussion of these data sets; Table 1 provides a quantitative comparison of these and other established crisis tweets data sets.

4. Methods

We now describe our method of processing the dataset, running our machine learning models on the dataset, and evaluating their performance. The code for these steps can be found here.

4.1. Data

For this study, we use a subset of 2,072,715 unique tweets after the removal of duplicates that was sampled from a much larger dataset of tweets directly purchased from Twitter via its data reseller, GNIP. 5,604,200 tweets were provided by Twitter for the following query:

from: Harvey, OR hurricane, OR flood, -is:retweet, -RT, -follow, -like, -new, -movie, -show, -gouging, -billion, -million, -redcross lang:en since:2017-08-22 until:2017-08-29.

This query is given in the syntax used by Twitter Advanced Search and translates into the following:

- **from**: No parameter is provided, so there is no restriction on the identity of the user who sent a tweet.
- **Harvey, OR hurricane, OR flood**: Only include tweets containing at least one of the words "harvey," "hurricane," or "flood." This restriction is not case-sensitive, so for example a tweet containing the word "HaRveY" would be included. We included this constraint to favor tweets related to Hurricane Harvey and its consequences (e.g. floods).
- **-is:retweet**: Exclude retweets. We do this since we want to create a dataset of unique requests for help.
- **-RT, -follow, -like, -new, -movie, -show, -gouging, -billion, -million, -redcross**: Exclude tweets containing any of the listed words. The word "RT" is found in retweets.
- **since:2017–08–22 until:2017–08–29**: Only allow tweets sent between August 22, 2017 and August 29, 2017. We included this constraint since this time range captures the peak of Hurricane Harvey's activity, since the storm achieved hurricane status on August 24 and receded from Texas on August 28.

Data was delivered as zipped JSON files containing tweets and associated metadata, including the time of creation, username, language, and a unique id. Since the focus of our study is to classify tweets based solely on the text, we only keep the text of each tweet. Random, manual inspection of this large dataset shows almost no urgent tweets, with most tweets being completely unrelated to emergency response or requests for help. We therefore filter our dataset by hashtags, specifically those containing at least one of the following keywords: "help," "rescue," and "911." We choose "help" and "rescue" since nearly every urgent tweet we encounter contains a hashtags with one of these words (e.g., #needhelp and #needwaterrescue), and we choose "911" based on our own judgment that tweets referencing emergency services are more likely contain requests for help from first-responders.

Filtering by these rescue-related keywords and removing tweets with duplicate text bodies produces a much smaller dataset of around 5000 tweets, which upon further manual inspection contains a much higher proportion of urgent tweets. Of these tweets, we manually label 2474 of them to produce our final dataset.

4.2. Labeling process

The entire dataset was manually labeled by 2 authors, with intercoder reliability measured to be 1.0 (i.e., 100% agreement) on a random sample of 70 tweets. This extremely high level of intercoder reliability is the result of a detailed and well understood codebook. Labeling was not delegated to a third-party service like Amazon's Mechanical Turk given the presence of personal data such as addresses in some of the tweets. Moreover, our own labeling of the dataset provided an opportunity to continually refine the definition of urgency. Ultimately, our final labeling taxonomy can be used by future researchers to reliably outsource labeling to other services.

4.3. Labeling taxonomy

The following is our definition of "urgent" tweets, which we then use to label our dataset and build the classifiers:

- Urgent: The user is urgently requesting help for themselves or on a specific person's/people's behalf. The tweet suggests that people's lives are in danger and/or a rescue is necessary. Specific information about the danger is included in the tweet text. Requests for animal rescues are excluded.
- Not urgent: All other tweets.

The following are the most common tweet types that are not considered urgent but may be mistaken for urgent requests for help: donation requests, general descriptions of hardship, descriptions of property damage without references to specific people and bodily harm, and requests for help for pets.

4.4. Preprocessing

4.4.1. Tokenization and lemmatization

Raw text is not easily interpretable by machine learning models, so the preprocessing of our pipeline involves the conversion of the raw text of a tweet into a numeric form that can be fed into a classifier. The first step in this process is to split the tweet text into tokens. Tokens are the individual logical symbols that make up human language, such as words, numbers, and punctuation. For example, the sentence "The cat ate 5 rats yesterday!" is made up of the list of tokens ['the', 'cat', 'ate', '5', 'rats', 'yesterday', '!']. To tokenize a tweet, we first make it lowercase and then use the Python Natural Language Toolkit's (NLTK) [32] TweetTokenizer class.

We represent Twitter user handles (e.g. @alice, @bob), URLs, numbers with generic <user>, <url>, and <number> tokens, since there are too many different users, numbers, and URLs to represent individually with distinct tokens. We experimentally determined that replacing hashtags with the designated hashtag token provided by GloVe hurts model performance, so we instead just replace hashtags with a word token excluding the hashtag (e.g. "#hurricane" becomes "hurricane"). Finally, we remove all non-word tokens except for the Twitter-specific ones just mentioned. For example, our tokenization method would convert the tweet "@alice @bob 12 donuts is a dozen! www.donuts.com #donuts" into the list of tokens ['<user>', '<user>', '<number>', 'donuts', 'is', 'a', 'dozen', '<url>', 'donuts'].

We then apply lemmatization, which is a procedure that takes a word token and replaces it with its "lemma," or simplest form. For example, in English lemmatization replaces verbs with their infinitive tense without the word "to" (e.g. the infinitive of "running" is "to run," so lemmatization outputs "run") and replaces nouns with their singular form ("mice" becomes "mouse"). We use the WordNet Lemmatizer found in the NLTK library for this step. Since tweets often contain misspelled and joined words, we use a Python implementation of the SymSpell word segmentation tool [33,34] to attempt to correct incorrectly-spelled and run-on words in each tweet.

4.4.2. Word embeddings features

After converting each tweet into a cleaned-up list of tokens, we replace each token with a 100-dimensional vector representing its meaning, called a word embedding. Word embeddings are vector representations of tokens (not just word tokens, despite the name) that are learned using a neural network such that tokens with closer meanings have vectors with high cosine similarity (a metric based on how small the acute angle between 2 vectors is). Furthermore, the vector representations of word meanings are often linear, so that meanings can be altered with addition and subtraction. For example, when word embeddings are trained using the word2vec method [19], subtracting the vector of the word "Spain" from that of its capital "Madrid" and adding the vector for "France" produces a new embedding vector that is closest to the capital of France, "Paris." The benefit of using word embeddings to represent words in machine learning is that they allow models to better generalize to new inputs [18]. Even if an input text contains words not seen by the model during training, if those words are conceptually related to previously-seen words, then their embeddings would be similar to the seen words, allowing the model to effectively generalize what it learned during training to the new input.

For our models, we use 100-dimensional GloVe word embeddings pretrained on a corpus of 2 billion tweets [20]. These embeddings provide vectors not just for words and punctuation, but also for the aforementioned Twitter-specific tokens, URL tokens, and number tokens.

The CNN model just takes a matrix of embeddings, where the rows of the matrix are the tokens in the tweet in order and the 100 columns specify the embedding vector of each token. The CNN is designed to take fixed-length inputs, that is, matrices with a fixed number of rows and columns. The number of columns is equal to the dimensionality of the embeddings and is thus fixed at 100. The number of rows equals the number of tokens in a tweet, which can vary. Since the longest tweet in our dataset contains 27 tokens after preprocessing, we just pad each tweet to this maximum length with vectors containing all zeros (since GloVe does not provide a dedicated <pad> token). We expect that during training the CNN can learn the semantic meaning of the zero padding vector. For the non-convolutional models, embedding vectors for the tokens in each tweet are averaged together component-wise (e.g., [1,2,3] and [0,0,0] averaged produce the vector [0.5,1,1.5]) to produce a fixed-length vector of 100 numerical features.

4.4.3. "Traditional" NLP features

To compare the performance of our use of average word embeddings to the use of more "traditional" features like n-grams and POS tags, we train the non-convolutional models on these other types of features and compare their performance with respect to F1, precision, recall, and accuracy (defined later in the Evaluation Metrics subsection).

One of the types of features we use is n-grams with and without negation detection [7,35]. An n-gram is merely a sequence of n words observed in a piece of text. For example, the sentence "The cat ate the rat." has the following bigrams (i.e., 2-grams): ['the', 'cat'], ['cat', 'ate'], ['ate', 'the'], and ['the', 'rat']. N-gram features are generated by assigning each piece of text in a dataset a binary vector with a length equal to the number of distinct n-grams in the dataset. An entry in the binary vector is 1 if that n-gram is found in the piece of text and 0 if not. We generate n-gram features after tokenizing the dataset and removing stopwords.

Negation detection is implemented by not representing negation words as individual tokens but rather merging them with the immediately adjacent word [35]. The example provided by Pak et al. [35] is to tokenize the phrase "I do not like fish" into bigrams as: "I do + not," "do + not like," and "not + like fish." In accordance with the preprocessing done by Pak et al. [35], we only remove articles ("a," "an," and "the") as stopwords, though we still use NLTK's TweetTokenizer to split the raw text into words. We experiment with the following combinations of n-gram features with and without negation detection: just unigrams ($n = 1$); just bigrams ($n = 2$); just trigrams ($n = 3$); unigrams and bigrams ($n = 1, 2$); and unigrams, bigrams and trigrams ($n = 1, 2, 3$). Our results using just unigrams are consistently the best with and without negation detection for nearly all the non-convolutional models (with the exception of the SVM, which performs only slightly better in other settings), so we only report the unigram results. To further regulate the volume of results we report, we don't implement any feature pruning (except when POS features are added, discussed below) and just train the models on the full feature set.

Another type of feature we use is POS tags, counts of the number of words of different parts of speech for each tweet. These features are not used in isolation but added to the above n-gram feature sets without negation detection. To increase the influence of these features in model behavior relative to n-gram features, we only select the top 1000 n-gram features using the information gain metric when including POS features. The number 1000 is experimentally determined in a sentiment analysis setting by Kouloumpis et al. [7]. Once again, our results using POS features with unigrams are the best across for all our models, so we only report the results of the unigram case.

4.4.4. *Oversampling*

A potential hindrance to the development of classifiers using our dataset is the infrequency of urgent tweets, which only make up 7% of the dataset after hashtag filtration. This class imbalance problem is common in machine learning and can cause models to learn to naively classify everything as the majority class to achieve high accuracy. For example, a model that classifies every tweet in our dataset as not urgent would achieve 93% accuracy. One approach we explore to alleviate class imbalance is oversampling, by randomly duplicating urgent tweets in the training set during each iteration of cross-validation (see the Training and Evaluation subsection) until it consists of one-third urgent tweets. This prevents models from naively classifying everything as not urgent to achieve high accuracy. We experiment with both the regular and oversampled datasets and compare their performances.

We choose to oversample after each train-test split in cross-validation to prevent each test split from containing artificially duplicated tweets, some of which could also be present in the training set. Thus models cannot overfit to the test set by being exposed to some of the same tweets at train time. Another concern related to overfitting is that since the models are trained on relatively few distinct urgent tweets, they could overfit to the specific urgent tweets in the training set and perform poorly on the test set. However, our models' cross-validated performance scores indicate that this likely does not happen. In fact, the performance of most models trained on word embeddings (averaged in the non-convolutional case) is not significantly affected by oversampling, so we do not experiment with non-convolutional models trained on traditional NLP features in the oversampled case.

Fig. 1 illustrates the pipeline we use to query the original data from Twitter, filter it to increase the proportion of urgent tweets in the

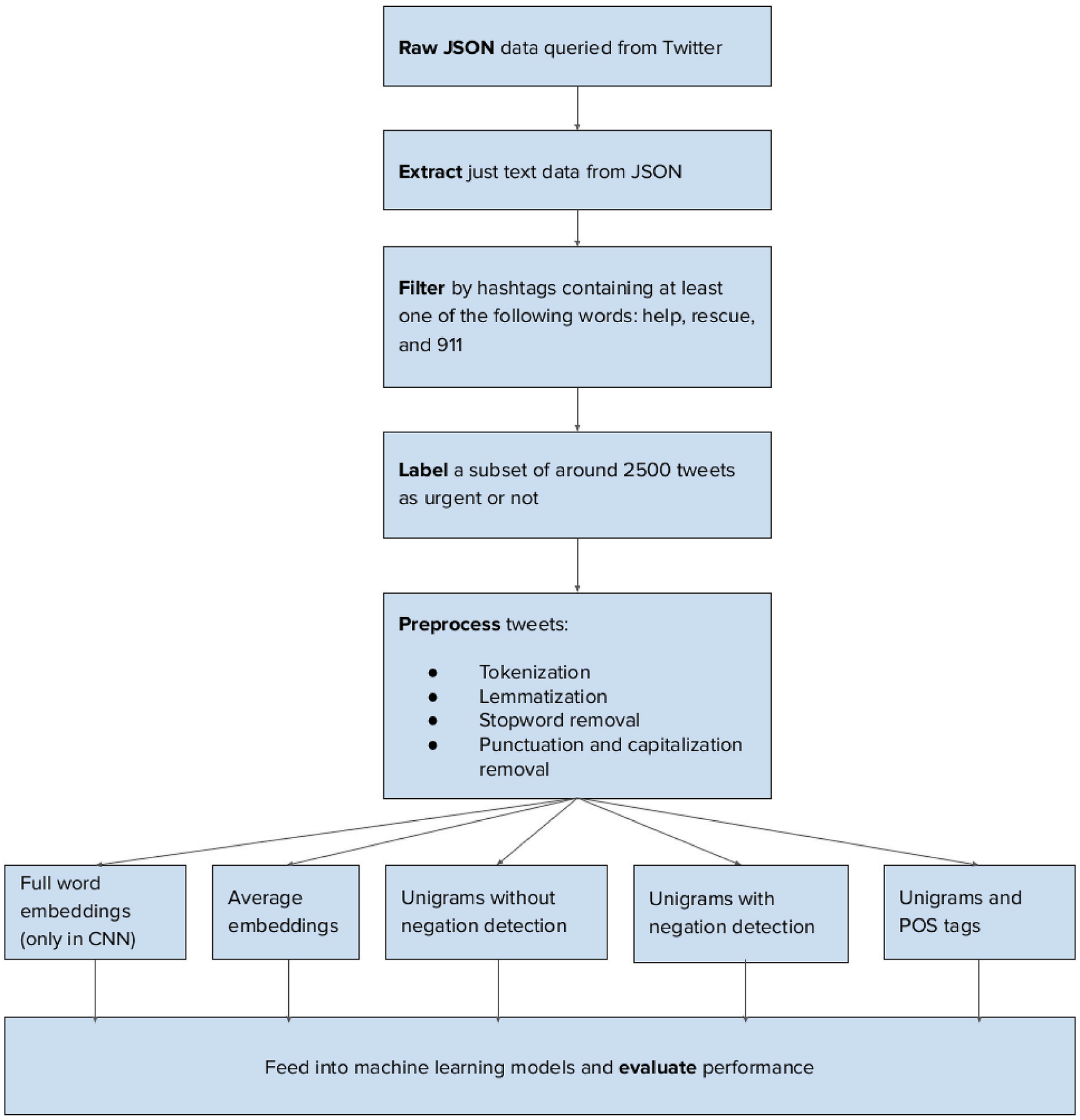
Raw JSON data queried from Twitter
Extract just text data from JSON
Filter by hashtags containing at least one of the following words: help, rescue, and 911
Label a subset of around 2500 tweets as urgent or not
Preprocess tweets:
Tokenization
Lemmatization
Stopword removal
Punctuation and capitalization removal
Full word embeddings (only in CNN)
Average embeddings
Unigrams without negation detection
Unigrams with negation detection
Unigrams and POS tags
Feed into machine learning models and **evaluate** performance

Fig. 1. Project architecture.

remaining dataset, preprocess the tweets into tokens, apply word embeddings or traditional NLP features, and feed these features into machine learning models.

4.5. Machine-learning models

The non-convolutional models we evaluate with the average embedding, n-gram, and POS features are naive Bayes [36], decision tree [37], AdaBoost [22], SVM [38], multilayer perceptron (MLP) [39], logistic regression [40], and ridge regression [41]. We consider naive Bayes due to its conceptual simplicity and ability to work well with small datasets. We consider AdaBoost, logistic regression, and SVM for their past successes in text classification [7,21]. The remaining models are considered since they are widely used in machine learning and good frames of reference to compare against.

The convolutional model we use with word embeddings is the CNN designed by Kim [11], and the implementation we use is largely borrowed from Trevett [42]. It takes as input an $n \times k$ real-valued matrix, where n is the number of words in the tweet and k is the length of each word's embeddings ($n = 27$ and $k = 100$ in this study). The input layer is followed by a single convolutional layer consisting of multiple filters with lengths equal to the dimension of the embeddings and widths spanning a window of words in each input tweet. The outputs of the filters are then concatenated and fed to a max-pooling layer, which is then followed by a fully-connected layer with dropout, to which the softmax function is applied to output a probability distribution over the possible labels ("urgent" or "not urgent"). The architecture is illustrated in Fig. 2. We do not use average embedding, n-gram, or POS features with the CNN since the CNN is designed for spatially-organized data, and those features are not directly tied to the locations of words or n-grams in a tweet.

The number of convolutional filters and their widths are hyperparameters that we tune. Another hyperparameter that we tune is the choice of activation, which is the function applied to the output of a node in a neural network layer before it is passed as input to the next layer in the network. The two most popular activation functions for this architecture are the hyperbolic tangent function (tanh) and rectified linear unit (ReLU) [24]. We only report the results using tanh since we experiment with both activations and find that tanh produces higher final F1 scores.

Our main motivation for choosing Kim's [11] architecture is that it is a relatively simple network with only 1 convolutional layer that could serve as a baseline to determine whether CNNs are effective for our classification task. Furthermore, we judge that tweets are probably too short to reap benefits from recurrent or LSTM modules as used by Hassan et al. [25].

4.6. Evaluation Metrics

The metrics with which we evaluate our models are F1, precision, recall, accuracy, and AUC. The reason we use multiple metrics is that due to the unbalanced nature of our dataset (with the vast majority of tweets being "not urgent"), a model with high accuracy is not necessarily desirable since it could achieve high accuracy by merely classifying every tweet as not urgent. Thus, we need a notion of what fraction of the few "urgent" examples the classifier correctly detects, as well as what fraction of the tweets classified as "urgent" are in fact urgent. This is where precision and recall are particularly relevant.

In the following discussion, let *TP* (true positive) denote the number of correct urgent classifications, *TN* (true negative) denote the number of correct not urgent classifications, *FP* (false positive) denote the number of tweets incorrectly classified as urgent, and *FN* (false negative) denote the number of tweets incorrectly classified as not urgent.

Recall is defined as the fraction of urgent tweets classified as urgent, or mathematically:

$$\text{recall} = \frac{TP}{TP + FN}$$

Precision is defined as the fraction of the tweets classified as urgent that are actually urgent, or mathematically:

$$\text{precision} = \frac{TP}{TP + FP}$$

Recall favors models that successfully detect most of the positive examples without any regard for how many negative examples are falsely classified as positive along with the true positives. Precision, on the other hand, favors classifiers that selectively classify examples as positive to maintain a low false-positive rate. Clearly a strong classifier is one that accurately detects a large percentage of the positive examples, and an intermediate metric measuring this is F1, which is defined as the harmonic mean of precision and recall:

$$\text{F1} = 2 \cdot \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

AUC is the area under the receiver operating characteristic (ROC) curve, which plots recall (the true positive rate) vs the false positive rate (FPR) for different model decision thresholds.

$$FPR = \frac{FP}{FP + TN}$$

The decision threshold of a classifier is some value used along with the model output to determine the output class. For example, in binary classification logistic regression models output a single probability. The output class of the model is determined by whether this probability is higher or lower than a chosen decision threshold (usually 0.5). AUC is

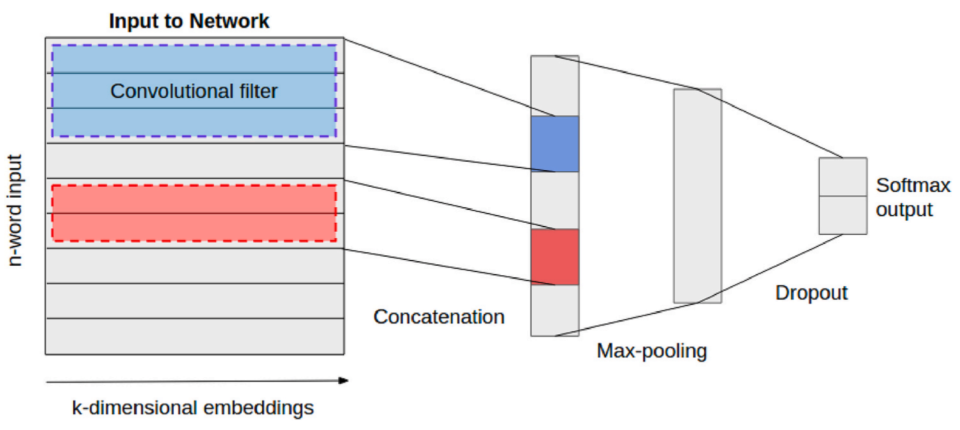

Fig. 2. Architecture of the CNN as described by Kim [11].

thus useful for determining how well a model is able to classify true negatives as negatives and true positives as positives over all possible decision thresholds (0.0–1.0 in the logistic regression case).

Both AUC and F1 are better metrics than accuracy since they take into account different aspects of a model's predictive ability. In this study we focus on F1 since that metric is used in the related disaster classification studies that we compare our results to, whereas AUC is not. Since our study is the first to address the problem of detecting urgent requests for help, we also provide AUC scores so that future work in this area has the ability to assess the quality of their models comparative to ours.

4.7. Training and Evaluation

A major hurdle to accurately estimating the performance of a machine learning model is that the performance score achieved on the training dataset (the dataset to which the model parameters are fit) is usually higher than that achieved on unseen data (the test dataset). One method used to estimate generalization performance on unseen data is k-fold cross-validation. In this method, the training dataset is split evenly into k parts, and in each of k iterations, the model is trained on a combined dataset of $k-1$ of the parts and assessed on the 1 left out. Finally, the k performance scores are averaged to yield a final estimate of generalization error. This estimate of generalization performance is more accurate than just training performance since in each iteration, the model is tested on data that it was not trained on.

To train and evaluate the aforementioned models, we perform grid search (training models with all possible parameter combinations) for important hyperparameters for each model, using 10-fold cross-validation to determine the best parameter combination with respect to the F1 metric. The CNN model in particular has a very large number of hyperparameters to search over, such as the convolutional filter widths and parameters pertaining to training the network, like batch size and number of epochs. To aid in the grid search, we use Zhang et al.'s [24] recommendation which outlines parameters ranges to try on Kim's [11] architecture.

5. Results

5.1. RQ1: Prevalence of urgent tweets

Urgent tweets were identified after manual inspection of tweets in our collected data (described in detail under "Methodology"). We very infrequently, but consistently, encountered tweets of individuals expressing personal peril and requesting help at specific locations. The tweets are distinct from more general reports like news updates and donation requests since they contain specific, actionable information about people in imminent danger that would be more useful to first responders.

Our original dataset consists of 2,072,715 unique tweets, and after a cursory examination of this dataset we determined that the urgent tweets were too infrequent for us to manually pick out to approximate the frequency of urgent tweets. For this reason, we choose to filter tweets by hashtags containing specific words which we reasoned would be more prevalent in urgent tweets. The first phrase we considered was just the word "help." We calculated the fraction of tweets containing each hashtag and ranked the hashtags in decreasing order of frequency. This revealed that the first hashtag with the word "help" in it is only the 80th most frequent and found in just 0.034% of the tweets. Table 2 lists the top 10 hashtags containing the word "help," the percentage of tweets they represent, and their frequency rank. The table illustrates that tweets containing help-related hashtags are highly infrequent in the data. Therefore, we expand our set of words to include "rescue" and "911." This results in a subsample of 4901 tweets. We then label a subset of 2474 of these tweets as urgent or not urgent and find that 7.6% of these hashtag-filtered tweets are urgent.

Table 1
Examples of urgent and non-urgent tweets as per our definition. Personal information (e.g., phone numbers, specific addresses, etc.) are redacted using asterisks.

Category	Example
Urgent-personal call for help	RT @OneofTwin: @HCSOTexas We need help in CE King Parkway Forest Subdivision, **** Sherrywood Drive Hou, Tx 77044, HELP. #harveyrescue #Har. . .
Urgent-on others' behalf	Please help my classmate and her elderly mom **** Homewood 77078 #harveyhouston #HarveyFlood #harvey #harvey2017texas … #USCG #HarveyRescue
Not Urgent but disaster-related	@JASONPAVA @AlyxandriaErryn #helpneedednow #harvey if you are stuck - please try calling 713-***-**** or ***-***-**** https://t.co/fF7sbfCsD9

Table 2
Top 10 most frequent hashtags containing the word "help," along with their frequency in the dataset and rank when sorted by frequency.

Frequency Rank	Hashtag	% of Tweets
80	help	$3.41 \cdot 10^{-2}$
155	helphouston	$1.76 \cdot 10^{-2}$
446	houstonhelpneeded	$5.92 \cdot 10^{-3}$
447	harveyhelp	$5.33 \cdot 10^{-3}$
709	helptexas	$3.62 \cdot 10^{-3}$
1101	texanshelpingtexans	$2.14 \cdot 10^{-3}$
1209	howtohelp	$1.95 \cdot 10^{-3}$
1276	helpers	$1.87 \cdot 10^{-3}$
1400	needhelp	$1.71 \cdot 10^{-3}$
1633	houstonhelp	$1.43 \cdot 10^{-3}$

Though we do not have an exact estimate of the frequency of urgent tweets in the original dataset, the fact that only 7.6% of even the filtered dataset is urgent illustrates the rarity of urgent requests for help among disaster-related tweets. It also illustrates that it is not feasible for workers to manually pick out urgent tweets in even relatively small datasets of a couple million tweets. Furthermore, the urgent tweets include examples of highly urgent, life-or-death content (e.g., individuals trapped by flooding or experiencing a medical emergency). Thus, despite their rarity, the high-stakes nature of these tweets confirms the necessity to develop classifiers to detect them.

Table 3
Results of non-convolutional models trained on average word embeddings (without oversampling) and CNN trained on full word embeddings (with oversampling). The error range given in Tables 3–7 is ±1 standard deviation, computed over the 10 folds of cross-validation.

Method	F1	Precision	Recall	Accuracy	AUC
CNN	0.87 ± 0.04	0.84 ± 0.08	0.92 ± 0.06	0.98 ± 0.01	0.99
SVM	0.87 ± 0.04	0.90 ± 0.06	0.85 ± 0.07	0.98 ± 0.01	0.99
MLP	0.82 ± 0.06	0.84 ± 0.06	0.81 ± 0.08	0.97 ± 0.01	0.98
AdaBoost	0.77 ± 0.05	0.82 ± 0.05	0.73 ± 0.08	0.97 ± 0.01	0.95
Logistic Regression	0.74 ± 0.09	0.87 ± 0.09	0.67 ± 0.13	0.97 ± 0.01	0.98
Naïve Bayes	0.50 ± 0.03	0.34 ± 0.03	0.93 ± 0.06	0.86 ± 0.02	0.96
Decision Tree	0.61 ± 0.09	0.62 ± 0.08	0.60 ± 0.11	0.94 ± 0.01	0.83
Ridge Classifier	0.61 ± 0.13	0.84 ± 0.09	0.49 ± 0.13	0.95 ± 0.01	0.97

5.2. RQ2: performance of machine-learning classifiers

Table 3 details the performance of our non-convolutional models trained on average word embeddings (without oversampling) and the CNN trained directly on word embeddings (with oversampling). We found that the best performers are the SVM and CNN which received the same F1 score of 0.87. The CNN has a significantly higher recall than the SVM but at the cost of precision. This indicates that it prioritizes classifying tweets as urgent at the cost of a higher false positive rate.

We do not report the performance of the CNN without oversampling. Given the heavy imbalance of the dataset, the CNN naively classifies almost every tweet as not urgent and thus achieves a very low F1 score, a behavior that has been observed in other disaster-related studies [43].

We also report the results of the non-convolutional models and CNN training with oversampling in Table 4. The CNN achieves the highest F1 score of 0.87, and the SVM is a close second with an F1 of 0.86. The only models that see a significant improvement in F1 when trained on the oversampled data are the decision tree and ridge regressor. This is due to an improvement in recall likely induced by the increased importance of detecting urgent tweets due to oversampling. Most of the other models in fact perform worse when trained on the oversampled data despite achieving higher recall due to a decrease in precision that overall brings down their F1 scores. For this reason, we do not report results for models trained on traditional NLP features with oversampling.

The SVM performance across all four metrics is similar with the introduction of oversampling. This is likely due to the fact that it learns a decision boundary between urgent and non-urgent tweets in the feature space and model accuracy is mainly determined by how accurately the SVM classifies points near the boundary. Oversampling urgent tweets at random—many of which are probably far from the decision boundary—likely does not improve crucial boundary point classification.

5.3. Average embeddings vs. traditional features

Tables 5–7 illustrate the performance of non-convolutional models trained without oversampling on traditional unigrams, unigrams with negation detection, and combined POS and unigram features respectively. We find that in all 3 cases, the MLP achieves the highest F1 of the non-convolutional models but a lower F1 than the CNN trained with oversampling. We also find that the addition of negation detection does not significantly affect the F1 scores of most of the classifiers, with the one exception being the naive Bayes classifier, which exhibits more than a double in F1. The inclusion of POS tags with unigrams also has little effect, and the naive Bayes classifier exhibits none of the drastic increase in F1 shown when negation detection is included.

The optimal hyperparameters for each of the models in Tables 5–7 are provided in the Appendix. Fig. 3 compares the F1 score of each model trained on average word embeddings without oversampling (with the exception of the CNN, which is only evaluated with oversampling) to the best F1 achieved by the model over all 3 n-gram feature variations explored (again with the exception of the CNN, which is only trained on word embeddings).

These results illustrate that whether average embeddings or n-gram features are the better choice depends on the type of model and that among the best-performing classifiers (SVM, MLP, and AdaBoost), average embeddings are a competitive choice for identifying urgent tweets from hurricane-related corpora. It would therefore be useful for future researchers in disaster text classification to treat average

Table 4
Results of non-convolutional models trained on average word embeddings and CNN trained on full word embeddings, both with oversampling.

Method	F1	Precision	Recall	Accuracy	AUC
CNN	0.87 ± 0.04	0.84 ± 0.08	0.92 ± 0.06	0.98 ± 0.01	0.99
SVM	0.86 ± 0.05	0.86 ± 0.08	0.87 ± 0.07	0.98 ± 0.01	0.99
MLP	0.83 ± 0.06	0.81 ± 0.09	0.87 ± 0.06	0.97 ± 0.01	0.99
AdaBoost	0.75 ± 0.05	0.75 ± 0.09	0.76 ± 0.09	0.96 ± 0.01	0.96
Logistic Regression	0.74 ± 0.07	0.64 ± 0.08	0.88 ± 0.08	0.95 ± 0.01	0.98
Naïve Bayes	0.48 ± 0.03	0.33 ± 0.03	0.93 ± 0.06	0.85 ± 0.02	0.96
Decision Tree	0.66 ± 0.09	0.63 ± 0.11	0.69 ± 0.09	0.94 ± 0.02	0.82
Ridge Classifier	0.69 ± 0.06	0.57 ± 0.07	0.88 ± 0.10	0.94 ± 0.02	0.98

Table 5
Results of non-convolutional models trained on unigrams without negation detection and without oversampling.

Method	F1	Precision	Recall	Accuracy	AUC
SVM	0.83 ± 0.10	0.91 ± 0.07	0.78 ± 0.13	0.98 ± 0.01	0.97
MLP	0.85 ± 0.09	0.93 ± 0.04	0.80 ± 0.12	0.98 ± 0.01	0.98
AdaBoost	0.77 ± 0.09	0.87 ± 0.07	0.71 ± 0.12	0.97 ± 0.01	0.93
Logistic Regression	0.81 ± 0.13	0.94 ± 0.06	0.72 ± 0.16	0.98 ± 0.01	0.98
Naïve Bayes	0.39 ± 0.04	0.26 ± 0.02	0.77 ± 0.10	0.82 ± 0.01	0.79
Decision Tree	0.73 ± 0.09	0.79 ± 0.11	0.69 ± 0.12	0.96 ± 0.01	0.83
Ridge Classifier	0.80 ± 0.14	0.94 ± 0.05	0.72 ± 0.17	0.98 ± 0.01	0.98

Table 6
Results of non-convolutional models trained on unigrams *with* negation detection and without oversampling.

Method	F1	Precision	Recall	Accuracy	AUC
SVM	0.81 ± 0.13	0.90 ± 0.07	0.75 ± 0.16	0.98 ± 0.01	0.97
MLP	0.85 ± 0.09	0.94 ± 0.04	0.79 ± 0.12	0.98 ± 0.01	0.98
AdaBoost	0.77 ± 0.06	0.87 ± 0.06	0.70 ± 0.09	0.97 ± 0.01	0.92
Logistic Regression	0.80 ± 0.12	0.94 ± 0.06	0.71 ± 0.14	0.97 ± 0.01	0.98
Naïve Bayes	0.81 ± 0.05	0.86 ± 0.08	0.77 ± 0.08	0.97 ± 0.01	0.89
Decision Tree	0.74 ± 0.09	0.80 ± 0.10	0.69 ± 0.12	0.96 ± 0.01	0.84
Ridge Classifier	0.79 ± 0.13	0.96 ± 0.04	0.69 ± 0.16	0.97 ± 0.01	0.98

Table 7
Results of models trained on n-gram and POS features, with the top 1000 n-gram features selected using the χ^2 statistic (without oversampling).

Method	F1	Precision	Recall	Accuracy	AUC
SVM	0.83 ± 0.10	0.91 ± 0.07	0.78 ± 0.13	0.98 ± 0.01	0.97
MLP	0.86 ± 0.10	0.94 ± 0.04	0.80 ± 0.14	0.98 ± 0.01	0.98
AdaBoost	0.77 ± 0.09	0.87 ± 0.08	0.71 ± 0.12	0.97 ± 0.01	0.93
Logistic Regression	0.81 ± 0.13	0.94 ± 0.06	0.72 ± 0.16	0.98 ± 0.01	0.98
Naïve Bayes	0.39 ± 0.04	0.26 ± 0.02	0.77 ± 0.10	0.82 ± 0.01	0.79
Decision Tree	0.74 ± 0.09	0.81 ± 0.10	0.70 ± 0.13	0.96 ± 0.01	0.84
Ridge Classifier	0.80 ± 0.14	0.94 ± 0.05	0.72 ± 0.17	0.98 ± 0.01	0.98

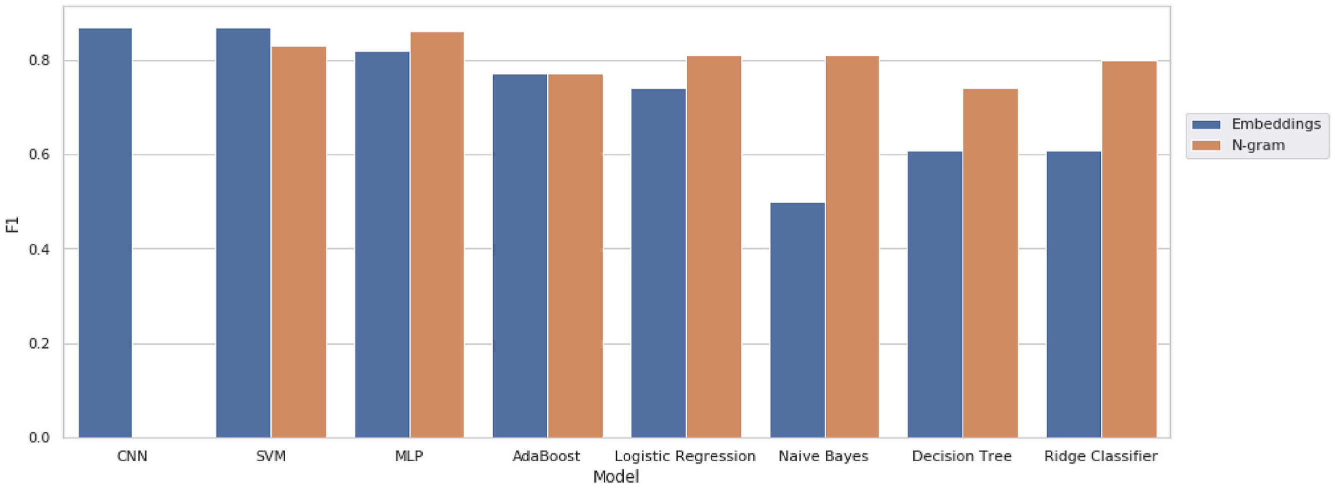

Fig. 3. Comparison of F1 scores of models trained on word embeddings and unigrams.

embeddings as another viable option when choosing a feature representation for the text.

5.4. Comparison to existing studies

Since we define a new classification problem that has not been significantly explored elsewhere, our results are not directly comparable to prior work. Purely as a frame of reference, we compare the scores of our best-performing models (CNN trained on full word embeddings, SVM trained on average embeddings, and MLP trained on unigrams and POS tags) to those of other classifiers in the domain of classifying disaster-related tweets.

We compare to Ashktorab et al. [21], who trained models on unigrams to classify tweets as mentioning specific infrastructure damage or human casualties; Imran et al. [15], who used logistic regression, random forests, and CNNs to classify tweets as relevant to a natural disaster or not; and Burel et al. [14], who used SVMs, decision trees, and CNNs (same architecture as Kim [11]) to classify tweets based on relatedness (unrelated or related), disaster type (flood, earthquake, bombing, etc.), and information type.

For brevity, we only report on 1 or 2 classification tasks from each study and only a couple of models from those that report multiple results. The following are the specific classification tasks and models examined from each study, along with the respective abbreviations as used in Table 8. The models trained in our study are not qualified with abbreviations.

- Ashktorab et al. (ASH): mentions infrastructure damage or human casualties
- Imran et al. (IMR): Relevant to Cyclone PAM or not
- Burel et al.: Related to crisis or not (BUR-R) and type of disaster referenced in the tweet (BUR-TD)

Table 8
Comparison of model performance from different studies on disaster text classification tasks.

Model	F1	Precision	Recall	Accuracy	AUC
CNN	0.87	0.84	0.92	0.98	0.99
SVM	0.87	0.90	0.85	0.98	0.99
MLP	0.86	0.94	0.8	0.98	0.98
SVM-BUR-TD	0.995	0.995	0.995	–	–
CNN-BUR-TD	0.983	0.983	0.983	–	–
SVM-BUR-R	0.829	0.833	0.83	–	–
CNN-BUR-R	0.838	0.839	0.838	–	–
CNN-IMR	–	–	–	–	0.94
Logistic Regression-ASH	0.65	0.78	0.57	0.86	0.88

The results in Table 8 indicate that our models perform the urgent tweet classification task better than most other models in the literature perform on their respective tasks. The one study with results far better than ours is the disaster type classification study conducted by Burel et al. [14], but they qualify those results with the claim that some event types contain very specific vocabulary that likely helps simplify the task.

6. Discussion

Unlike previous work on disaster-related social media classification, we focus exclusively on those individuals who are using Twitter to ask for help and provide enough information for first responders to act on (i. e., those who send what we define as “urgent” tweets). Our results indicate that we can accurately identify urgent tweets that are actionable by first responders using tweets posted during Hurricane Harvey. Additionally, our study uniquely trains non-convolutional machine learning models using average word embeddings and provides a performance comparison with those trained with more traditional POS and n-gram features. We also confirm the efficacy of Kim’s [11] convolutional neural network (CNN) architecture in detecting relevant disaster tweets, as Burel et al. [14] did previously. This provides further evidence for disaster-related text classification to utilize Kim’s [11] methods.

Furthermore, we highlight that urgent requests for help are very uncommon among tweets posted during a natural disaster (in our specific case, Hurricane Harvey), but that with pre-processing by hashtags, a sample can be generated with a small but non-negligible percentage of urgent tweets. Furthermore, we provide examples of these urgent tweets and illustrate that their life-or-death content merits the development of classifiers to detect them in the event of a large-scale disaster. The identification of these posts during a disaster could be of immense value to first responders and other emergency response stakeholders.

We test several different methods of identifying truly urgent tweets that would be useful to first responders in a disaster situation. The support vector machine (SVM) trained on average word embeddings without oversampling and CNN trained on full word embeddings with oversampling achieve equal F1 scores of 0.87, though the CNN has higher recall and the SVM has higher precision. Since the cost of a false positive (falsely classifying a non-urgent tweet as urgent) is lower than that of a false negative (ignoring an urgent tweet), the higher recall of the CNN is generally preferable.

Comparing our best models’ F1 scores to the scores of other disaster classification models indicate that our models’ performance is competitive with prior work classifying disaster-related tweets. Our models lag only behind the CNN and SVM trained by Burel et al. [14], which tackle the simpler task of identifying the type of disaster referenced in a tweet.

However, we do define a new type of classification problem, so our models are not fully comparable to the literature.

We find that representing tweets using average word embeddings pretrained on Twitter data provides results comparable to the representation of tweets with traditional n-grams and POS tags for the best-performing non-convolutional models (SVM, MLP, AdaBoost), with the SVM trained on average word embeddings achieving the best F1 score along with the CNN. This suggests that average embeddings should be added to the collection of NLP features considered by future researchers designing methods to classify disaster-related text in social media platforms.

7. Limitations and future work

For the task of identifying urgent tweets from Hurricane Harvey, the CNN, SVM, and MLP all perform well. However, the models are currently trained solely for data from Hurricane Harvey, and we have not tested their ability to generalize to different disasters. Such generalizability would be desired in practice, as the process of labeling training data and developing models is time-consuming and not feasible during an actual disaster.

We plan to create classifiers that are able to generalize and identify urgent tweets across hurricanes and subsequently across any floods, tornados, etc. (if requests for help are indeed present on Twitter for these disasters). One potential approach is to replace instances of street and disaster names (e.g. Sherrywood Drive, harvey, florence) with representative word embeddings in the preprocessing step itself. Such word embeddings would be an abstract representation of these names that could be generalized past any individual disaster.

Another limitation of our current study is that our data consists entirely of tweets relevant to Hurricane Harvey. In practice, our models would have to detect urgent tweets from among numerous tweets that may be completely unrelated to hurricanes, so it would be useful to train our models on more diverse, “noisy” data to verify that they have not just overfit a very specific type of tweet.

We believe there is also utility in evaluating whether the geographic location from which a tweet is posted during a disaster influences how likely it is to be urgent, as Neppalli et al.’s [17] sentiment analysis of tweets sent during Hurricane Sandy indicated an overall stronger negative sentiment in tweets sent in close proximity to the storm. Although Twitter allows users to share their GPS coordinates, most do not. Moreover, users who provide location information on their profile often provide descriptions that are either fake or too broad to be very useful (e.g., at the country level) [44]. Thus the geographic location of a tweet would have to be inferred through other means, like user relationships [45], n-grams [46], and specific keywords extracted using named-entity recognition [44].

Future work could also use our approach, but by using location data to broaden the scope of a search for people affected by a disaster. In other words, rather than focus on specific people who need help, it would be useful to be able to isolate small groups of people that are affected by the hurricane based on their location data and movement patterns.

8. Conclusion

In this study, we identify a specific type of urgent request for help posted on Twitter during disasters using the case study of Hurricane Harvey in 2017. Prior work has not focused on the classification of urgent requests for help that are actionable by first responders. A major problem during large-scale disasters such as hurricanes is that emergency phone services (e.g., 911 in the U.S.) often get overloaded with requests. In the U.S., people now turn to social media platforms like Twitter to make urgent requests for help. We underscore that though urgent tweets are uncommon among the tweets posted during Hurricane Harvey, the potential life-or-death nature of their content justifies the development of classifiers capable of detecting them. This marks a departure from the existing literature which either focuses on the detection of tweets mentioning damage without specifically considering actionability or focuses on broader questions entirely, such as whether a tweet is relevant to the disaster at all.

We then successfully develop classifiers to detect these urgent tweets, and our best-performing models achieve F1 scores of 0.86 or higher despite the highly unbalanced nature of the data. Our work therefore establishes that urgent requests made on social media such as Twitter can be detected using machine learning models and underscores the need for future research on ways to create models that generalize to future natural disasters. Such general models could then be used by emergency services or relevant relief stakeholders to automatically detect requests for help on social media in real-time as a disaster progresses. We also demonstrate that average word embeddings are a feature type competitive with more traditional features in NLP like n-grams and POS tags when used with non-convolutional models; they are one more option for feature representation that future NLP researchers can consider when building text classifiers.

Unlike previous work, this study focuses on specific and personal requests for help, and our research questions do not exclusively evaluate whether social media content is related or unrelated to Hurricane Harvey. Rather, we provide evidence not only that specific, urgent requests for help are posted on Twitter during Harvey, but also that machine learning models can be developed to accurately detect these requests. Ultimately, despite these posts being “needles in a haystack,” identifying any affected people during a hurricane can make a real difference to the work of first responders and other relevant stakeholders.

Funding

This work was supported by the following grants from the U.S. National Science Foundation: Award No. 1760453, project “RAPID: The Changing Nature of “Calls” for Help with Hurricane Harvey: 9-1-1 and Social Media” and Award No. 1902460, project “RAPID: Rural Loss Estimates of Hurricane Florence Enabled by Citizen Scientists”. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Optimal Hyperparameters

The following tables detail the optimal hyperparameters discovered for each model in each experiment following grid search with 10-fold cross-validation. The parameter names are those used in the Scikit-learn library [47]. All parameters not listed take the default values as per the Scikit-learn version 0.21.1 documentation. Each table corresponds to a single model type and contains the optimal parameters discovered in all the experiments using that model type. The experiments are listed in the same order as discussed in the Results section.

The following are explanations for what each of the CNN architecture hyperparameters mean, since the CNN is not implemented by Scikit-learn:

- **dropout**: The probability that one of the input nodes to the dropout layer is zeroed out
- **n_filters**: The number of convolutional filters of each size used
- **filter_sizes**: The set of filter widths used, with n_filters number of each filter size in the architecture

For example, if n_filters = 100 and filter_sizes =[1,2], then the CNN has 200 filters in total, with 100 having a width of 1 word and 100 having a width of 2 words.

Table Appendix A.1
Optimal hyperparameters for the CNN.

Experiment	dropout	n_filters	filter_sizes	epochs	batch_size
Full word embeddings with oversampling	0.3	700	1, 1, 2, 2	100	10

Table Appendix A.2
Optimal hyperparameters for the decision tree.

Experiment	max_depth	min_samples_leaf
Average word embeddings, no oversampling	5	1
Average word embeddings, with oversampling	20	1
Unigrams, no negation detection	20	1
Unigrams, with negation detection	None	1
Unigrams and POS tags	50	1

Table Appendix A.3
Optimal hyperparameters for the SVM.

Experiment	C	gamma	kernel
Average word embeddings, no oversampling	10	0.1	rbf
Average word embeddings, with oversampling	10	0.1	rbf
Unigrams, no negation detection	1	0.001	linear
Unigrams, with negation detection	1	0.001	linear
Unigrams and POS tags	1	0.001	linear

Table Appendix A.4
Optimal hyperparameters for the MLP.

Experiment	activation	alpha	early_stopping
Average word embeddings, no oversampling	relu	0.001	FALSE
Average word embeddings, with oversampling	relu	0.1	FALSE
Unigrams, no negation detection	logistic	0.0001	FALSE
Unigrams, with negation detection	logistic	0.001	FALSE
Unigrams and POS tags	logistic	0.0001	FALSE

Table Appendix A.5
Optimal hyperparameters for AdaBoost.

Experiment	learning_rate	n_estimators
Average word embeddings, no oversampling	1	50
Average word embeddings, with oversampling	1	100
Unigrams, no negation detection	1	100
Unigrams, with negation detection	1	100
Unigrams and POS tags	1	100

Table Appendix A.6
Optimal hyperparameters for the ridge classifier.

Experiment	alpha
Average word embeddings, no oversampling	0.1
Average word embeddings, with oversampling	5
Unigrams, no negation detection	10
Unigrams, with negation detection	10
Unigrams and POS tags	10

Appendix B. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.ijdrr.2020.101757.

References

[1] D. Murthy, A. Gross, M. McGarry, Visual Social Media and Big Data. Interpreting Instagram Images Posted on Twitte, Digital Culture & Society 2 (2) (2016) 113–134, https://doi.org/10.14361/dcs-2016-0208. https://transcript.degruyter.com/view/journals/dcs/2/2/article-p113.xmlP.

[2] J.B. Houston, J. Hawthorne, M.F. Perreault, E.H. Park, M. Goldstein Hode, M. R. Halliwell, S.E. Turner McGowen, R. Davis, S. Vaid, J.A. McElderry, S.A. Griffith, Social media and disasters: a functional framework for social media use in disaster planning, response, and research, Disasters 39 (1) (2015) 1–22, https://doi.org/10.1111/disa.12092, arXiv, https://onlinelibrary.wiley.com/doi/pdf/10.1111/disa.12092.

[3] M.E. Phillips, Hurricane Harvey twitter dataset. https://digital.library.unt.edu/ark:/67531/metadc993940/, 2017.

[4] N. Pourebrahim, S. Sultana, J. Edwards, A. Gochanour, S. Mohanty, Understanding communication dynamics on Twitter during natural disasters: A case study of Hurricane Sandy, Int. J. Disaster Risk Reduct. 37 (2019) 101176, https://doi.org/10.1016/j.ijdrr.2019.101176. http://www.sciencedirect.com/science/article/pii/S2212420918310434.

[5] T.A. Glass, Understanding public response to disasters, Publ. Health Rep. 116 (Suppl 2) (2001) 69–73.

[6] M. Rodhan, 'Please Send Help.' Hurricane Harvey Victims Turn to Twitter and Facebook, Time, 08/30/2017. https://time.com/4921961/hurricane-harvey-twitter-facebook-social-media/.

[7] E. Kouloumpis, T. Wilson, J. Moore, Twitter Sentiment Analysis: The Good the Bad and the OMG!, in: Fifth International Association for the Advancement of Artificial Intelligence Conference on Weblogs and Social Media, Barcelona, Spain, AAAI Press, 2011, pp. 538–541.

[8] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent Dirichlet Allocation, J. Mach. Learn. Res. 3 (2003) 993–1022.

[9] G. Xiang, B. Fan, L. Wang, J. Hong, C. Rose, Detecting offensive tweets via topical feature discovery over a large scale twitter corpus, in: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, New York, United States, Association for Computing Machinery, 2012, pp. 1980–1984.

[10] C. dos Santos, M. Gatti, Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts, in: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, Dublin City University and Association for Computational Linguistics, Dublin, Ireland, 2014, pp. 69–78. https://www.aclweb.org/anthology/C14-1008.

[11] Y. Kim, Convolutional Neural Networks for Sentence Classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1746–1751. https://www.aclweb.org/anthology/D14-1181.

[12] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324, https://doi.org/10.1109/5.726791.

[13] S. Lai, L. Xu, K. Liu, J. Zhao, Recurrent convolutional neural networks for text classification, in: AAAI Conference on Artificial Intelligence, Austin, Texas, United States, Association for Computing Machinery, 2015. https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9745.

[14] G. Burel, H. Saif, M. Fernandez, H. Alani, On Semantics and Deep Learning for Event Detection in Crisis Situations, in: Workshop on Semantic Deep Learning (SemDeep), at the European Semantic Web Conference (ESWC) 2017, Elsevier, 2017. http://oro.open.ac.uk/49639/.

[15] M. Imran, P. Mitra, C. Castillo, Twitter as a Lifeline: Human-annotated Twitter Corpora for NLP ofCrisis-related Messages, in: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), European Language Resources Association (ELRA), Portorož, Slovenia, 2016.

[16] L. Derczynski, K. Meesters, K. Bontcheva, D. Maynard, Helping Crisis Responders Find the Informative Needle in the Tweet Haystack, in: ISCRAM 2018 Conference Proceedings - 15th International Conference on Information Systems for Crisis Response and Management, Rochester, New York, United States, Information Systems for Crisis Response And Management, 2018, pp. 649–662. http://idl.iscram.org/files/leonderczynski/2018/2139_LeonDerczynski_etal2018.pdf.

[17] V.K. Neppalli, C. Caragea, A. Squicciarini, A. Tapia, S. Stehle, Sentiment analysis during Hurricane Sandy in emergency response, Int. J. Disaster Risk Reduct. 21 (2017) 213–222, https://doi.org/10.1016/j.ijdrr.2016.12.011. http://www.sciencedirect.com/science/article/pii/S2212420916302151.

[18] Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin, A neural probabilistic language model, J. Mach. Learn. Res. 3 (2003) 1137–1645, 1155.

[19] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K.Q. Weinberger (Eds.), NIPS'13: Proceedings of the 26th International Conference on Neural Information Processing Systems 2, Curran Associates, Inc., 2013, pp. 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

[20] J. Pennington, R. Socher, C.D. Manning, GloVe: Global Vectors for Word Representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Association of Computing Machinery, Doha, Qatar, 2014, pp. 1532–1543.

[21] Z. Ashktorab, C. Brown, M. Nandi, A. Culotta, Tweedr: Mining twitter to inform disaster response, in: ISCRAM 2014 Conference Proceedings - 11th International Conference on Information Systems for Crisis Response and Management, Information Systems for Crisis Response and Management, Pennsylvania, United States, 2014, pp. 354–358.

[22] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, in: P. Vitányi (Ed.), Computational Learning Theory, Springer Berlin Heidelberg, Berlin, 660 Heidelberg, Germany, 1995, pp. 23–37, https://doi.org/10.1006/jcss.1997.1504.

[23] M. Imran, S. Elbassuoni, C. Castillo, F. Diaz, P. Meier, Extracting Information Nuggets from Disaster-Related Messages in Social Media, in: ISCRAM 2013 Conference Proceedings - 10th International Conference on Information Systems for Crisis Response and Management, Information Systems for Crisis Response and Management, Baden-Baden, Germany, 2013, pp. 791–801.

[24] Y. Zhang, B.C. Wallace, A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. http://arxiv.org/abs/1510.03820.

[25] A. Hassan, A. Mahmood, Convolutional Recurrent Deep Learning Model for Sentence Classification, IEEE Access 6 (2018) 13949–13957, https://doi.org/10.1109/access.2018.2814818.

[26] S. Hochreiter, J. Schmidhuber, Long Short-Term Memor, Neural Comput. 9 (8) (1997) 1735–1780.

[27] R. Socher, A. Perelygin, J. Wu, J. Chuang, C.D. Manning, A.Y. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, Washington, United States, 2013, pp. 1631–1642.

[28] A. Olteanu, C. Castillo, F. Diaz, S. Vieweg, CrisisLex: A Lexicon for Collecting and Filtering Microblogged Communications in Crises, in: Eighth International AAAI Conference on Weblogs and Social Media, Ann Arbor, Michigan, United States, AAAI Press, 2014.

[30] F. Alam, H. Sajjad, M. Imran, F. Ofli, Standardizing and Benchmarking Crisis-Related Social Media Datasets for Humanitarian Information Processing, arXiv, 2020, 06774. https://arxiv.org/abs/2004.06774.

[31] A. Olteanu, S. Vieweg, C. Castillo, 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, New York, NY, USA, 2015, pp. 994–1009, https://doi.org/10.1145/2675133.2675242.

[32] E. Loper, S. Bird, Nltk: the Natural Language toolkit, arXiv computing research repository (CoRR). https://arxiv.org/abs/cs/0205028.

[33] W. Garbe, Symspell, Aug. 2019. https://github.com/wolfgarbe/SymSpell.

[34] L. mmb, Symspellpy, Aug. 2019. https://github.com/mammothb/symspellpy.

[35] A. Pak, P. Paroubek, Twitter as a corpus for sentiment analysis and opinion mining, in: Seventh International Conference on Language Resources and Evaluation, Valletta, Malta, 2010, pp. 1320–1326.

[36] I. Rish, et al., An empirical study of the naive bayes classifier, in: IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, vol. 3, 2001, pp. 41–46.

[37] S.R. Safavian, D. Landgrebe, A survey of decision tree classifier methodology, IEEE transactions on systems, man, and cybernetics 21 (3) (1991) 660–674, https://doi.org/10.1109/21.97458.

[38] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297, https://doi.org/10.1007/BF00994018.

[39] S. K. Pal, S. Mitra, Multilayer perceptron, fuzzy sets, classifiaction, IEEE Trans. Neural Network. 3 (5).

[40] C.-Y.J. Peng, K.L. Lee, G.M. Ingersoll, An introduction to logistic regression analysis and reporting, J. Educ. Res. 96 (1) (2002) 3–14, https://doi.org/10.1080/00220670209598786.

[41] A.N. Tikhonov, On the stability of inverse problems, Dokl. Akad. Nauk SSSR 39 (1943) 195–198.

[42] B. Trevett, Pytorch Sentiment Analysis, Aug. 2019. https://github.com/bentrevett/pytorch-sentiment-analysis.

[43] B.W. Robertson, M. Johnson, D. Murthy, W.R. Smith, K.K. Stephens, Using a combination of human insights and 'deep learning' for real-time disaster communication, Progress in Disaster Science 2 (2019) 100030, https://doi.org/10.1016/j.pdisas.2019.100030. http://www.sciencedirect.com/science/article/pii/S2590061719300304.

[44] A. Kumar, J.P. Singh, Location reference identification from tweets during emergencies: a deep learning approach, International Journal of Disaster Risk Reduction 33 (2019) 365–375, https://doi.org/10.1016/j.ijdrr.2018.10.021. http://www.sciencedirect.com/science/article/pii/S2212420918307799.

[45] C.A. Davis Jr., G.L. Pappa, D.R.R. de Oliveira, F. de L. Arcanjo, Inferring the location of twitter messages based on user relationships, Trans. GIS 15 (6) (2011) 735–751, https://doi.org/10.1111/j.1467-9671.2011.01297.x. https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9671.2011.01297.x.

[46] R. Priedhorsky, A. Culotta, S.Y. Del Valle, Inferring the origin locations of tweets with quantitative confidence, in: Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing, CSCW '14, Association for Computing Machinery, New York, NY, USA, 2014, pp. 1523–1536, https://doi.org/10.1145/2531602.2531607.

[47] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (Oct) (2011) 2825–2830.

[49] Dat Nguyen, Shafiq Joty, Muhammad Imran, Hassan Sajjad, Prasenjit Mitra, Applications of Online Deep Learning for Crisis Response Using Social Media Information, arXiv (2016). https://arxiv.org/pdf/1610.01030.pdf.