Development of a national-scale real-time Twitter data mining pipeline for social geodata on the potential impacts of flooding on communities

John Luke Patrick Barker, J.L.P. Barker, C.J.A. Macleod

Development of a national-scale real-time Twitter data mining pipeline for social geodata on the potential impacts of flooding on communities

J. L. P. Barker[a] and C. J. A. Macleod[b]*

[a]Computing, University of Dundee, Queen Mother Building, Balfour Street, Dundee, DD1 4HN, UK. lukebarker@gmail.com +44 (0) 7962 440320

[b]Information & Computational Sciences Group, The James Hutton Institute, Craigiebuckler, Aberdeen, AB15 8QH, UK. kit.macleod@hutton.ac.uk. +44 (0) 1224 395157

*Corresponding author

Abstract

Social media, particularly Twitter, is increasingly used to improve resilience during extreme weather events/emergency management situations, including floods: by communicating potential risks and their impacts, and informing agencies and responders. In this paper, we developed a prototype national-scale Twitter data mining pipeline for improved stakeholder situational awareness during flooding events across Great Britain, by retrieving relevant social geodata, grounded in environmental data sources (flood warnings and river levels). With potential users we identified and addressed three research questions to develop this application, whose components constitute a modular architecture for real-time dashboards. First, polling national flood warning and river level Web data sources to obtain at-risk locations. Secondly, real-time retrieval of geotagged tweets, proximate to at-risk areas. Thirdly, filtering flood-relevant tweets with natural language processing and machine learning libraries, using word embeddings of tweets. We demonstrated the national-scale social geodata pipeline using over 420,000 georeferenced tweets obtained between 20-29th June 2016.

Highlights

- Prototype real-time social geodata pipeline for flood events and demonstration dataset
- National-scale flood warnings/river levels set ‘at-risk areas’ in Twitter API queries
- Monitoring multiple locations (without keywords) retrieved current, geotagged tweets
- Novel application of word embeddings in flooding context identified relevant tweets
- Pipeline extracts tweets to visualise using open-source libraries (SciKit Learn/Gensim)

Keywords

Flood management; Twitter; volunteered geographic information; natural language processing; word embeddings; social geodata.

Software/data availability

Name of software: Various scripts as they apply to pipeline steps.

Description: all scripts for various steps of the pipeline are described and specified in supplied Appendix A1.2.1, Table A.2.
Developer: J. L. P. Barker
Contact: lukebarker@gmail.com
Year first available: 2017
Hardware required: Intel i3 or mid-performance PC with multicore processor and SSD main drive, 8Gb memory recommended.
Software required: Python and library dependencies specified in Appendix A1.2.1, (viii) environment.yml
Software availability: All source code can be found at GitHub public repositories https://github.com/battez/analysis/releases/tag/v0.5-alpha ; https://github.com/battez/tweepy_now and https://github.com/battez/twystream/releases/tag/v0.5-beta .

Configuration file for python environment, with all necessary dependencies: https://drive.google.com/open?id=0B057bbdoYJDLYjlIOUFLMkJ0TDg

Cost: Free. Software and source code are released under the New Berkeley Software Distribution (BSD) License, which allows for liberal reuse of the software and code.

Name of data set: 1 - Unlabelled Tweets Development Dataset
Description: Unlabelled June dataset 420,218 geotagged tweets and identifiers, in CSV format: flat file of Tweet identifiers for lookup via the Twitter API. Or can search individually via Twitter website: e.g. https://twitter.com/statuses/745907644641193984
Developer: J. L. P. Barker
Availability: Free, download from:
https://drive.google.com/open?id=0B057bbdoYJDLT0REZFFWNDQyMXc
Name of data set: 2 - Labelled Tweets used in training classifier.
Description: Labelled subset of June dataset 4502 tweet identifiers in CSV format: (Note: t_class=1 Relevant class):

Developer: J. L. P. Barker
Availability: Free, download from:
https://drive.google.com/open?id=0B057bbdoYJDLVjg1YTd5d0JsdkU

1. Introduction

Social media is increasingly being used during natural catastrophes and emergency management situations, including flooding events, to improve two-way communication and understanding of potential risks and their impacts (Kryvasheyeu et al, 2016). Stakeholders can inspect actionable information on real-time dashboards that integrate different sources and provide an information channel for communities (Mukkamala and Beck, 2016). Effective data mining of social media to improve crisis response is sought by the humanitarian sector (Middleton et al, 2014) but the volume of data proves difficult to identify relevant information for decision-making (Vieweg et al, 2014). In part, this is due to dealing with large amounts of data with varying quality/information content (see Meier 2015 for a summary). Where spatial size and significance of natural catastrophes can influence the quality of social geodata (Middleton et al, 2014). Employing software (including machine learning) to alleviate information overload or data reliability issues is crucial to increasing adoption by managers, and sensemaking requires geotagging and map visualisations (Rao et al, 2017; Li et al, 2017). Indeed, stakeholders want place-based, geotagged, crowd-sourced or "volunteered geographic information" (VGI) about flooding events (Juhász et al, 2016).

There is a need to automate identification and extraction of spatial information about flood events from social media e.g. tweets (Smith et al, 2015; Eilander et al, 2016; Arthur et al, 2018). Accurately identifying localised flood extent is difficult, especially during pluvial and fluvial flooding in urban areas, and social media can help with gathering and disseminating information (Smith et al, 2015; de Albuquerque et al, 2015; Middleton et al, 2014; Herfort et al, 2014). Internationally, increases in intense localised convective rainfall events have been observed (e.g. Zheng et al, 2015). Recently, extraction of pluvial flood relevant VGI by deep learning from user generated texts and photos has been demonstrated (Feng and Sester 2018). The characterisation of catchment response during local flash flood events were shown to particularly benefit from combined authoritative sources and community data (VGI and citizen science) (Starkey et al, 2017). Analysis of social geodata during floods in Sao Paulo, Brazil showed geographical relations were useful for identifying tweets relevant to flooding (de Assis et al, 2016). Other flood related examples include: georeferenced tweets aiding real-time flood extent observations and agency response in Jakarta, Indonesia (Eilander et al, 2016), and during floods on the River Elbe, in Germany (Herfort et al, 2014).

Generally, research has focussed on using social media data in a stand-alone manner for specific events and locations, rarely attempting integration with real-time environmental sources - which are continually improving and becoming accessible via public application programming interfaces (APIs) (Table A.1). One exception was the integration of environmental sensor data with catchment polygons to prioritise tweets in Sao Paulo (de Assis et al, 2016). Existing web applications- that harness social media during emergency events- using machine learning techniques have generally been developed

for specific incidents to be searched for - typically, one-off catchment to regional scale situations (Spielhofer et al, 2016) - rather than supporting national level surveillance monitoring of natural hazard events and their social impacts, e.g. only at a city-scale (Eilander et al, 2016).

In this paper, we demonstrate and test a prototype machine learning social geodata pipeline, based on real-time flood warnings and river level information, and natural language processing of tweets during flooding events, with the aim to improve situational awareness for flood risk managers and other stakeholders. This involved novel automated selection and analysis of large volumes of geotagged and relevant social media data, and recent advances in vector-based natural language processing (NLP). Developing this prototype application required three research questions to be addressed.

1) *How to programmatically identify at a national-scale (across Great Britain) areas at-risk of flooding based on real-time flood warnings and river level information*?
   The first research question was how to identify timely shortlists of locations at-risk of flooding using available national-scale, real-time flood warnings (England/Wales) and river level (Scotland) information sources (from hundreds of potential locations). Previous studies highlighted that sensor data (e.g. river level gauges) could support prioritising social media messages during flood events (de Albuquerque et al, 2015). Internationally, there are an increasing number of near real-time environmental data sources, providing information on at-risk areas, river levels and rainfall (Table A.1).

2) *How to automate the spatiotemporal retrieval of real-time social geodata using Twitter APIs?*
   The second research question was to investigate the viability of automated retrieval of social geodata (i.e. tweets geotagged by their author) in a timely and continuous way, based on multiple locations prioritised by potential risk of flooding. Previous studies researched cross-referencing tweets with prioritised locations for mapping flooding (Middleton et al, 2014), and location-based queries during floods, using georeferencing tweets (Laylavi et al, 2016). A recent application, GeoViewer, queried a location (as defined by a user), via the Twitter Streaming API, to output maps of tweets in real-time (Tsou et al, 2017).

3) *How to automatically identify flood-relevant tweets?*
   The third research question was whether we could achieve automatic identification of individual tweets for flooding relevance. Herfort et al. (2014) noted such an automated task was difficult to achieve. We defined "relevance" as social geodata that contributed to situational awareness for managing in-progress flooding events. Previous studies in crisis contexts generally used initial keyword filtering, with subsequent georeferencing: in flooding

events (*FloodTags* app[1] of Jongman et al, 2015), hurricanes (Kryvasheyeu et al, 2016) and landslides (Musaev and Hou, 2016).

Our research aim was to develop and test a prototype integrated real-time solution based on these three research questions for stakeholder map-based dashboard visualisations, during a series of flooding events across England and Scotland between June 20-29th, 2016; that was also applicable to other countries and a range of weather-driven emergency situations.

2. Methods

Our prototype was partly inspired by several discussions with SEPA flood risk managers (April-June 2016), during these discussions we revised the aim of our research to demonstrate a prototype pipeline linking real-time river level information and social media to aid national level flood management. During a visit to the SEPA flood forecasting centre (Perth, Scotland), we were shown flood risk dashboards and learned of their usefulness for two-way communication about floods via community flood observation reporting tools (Report-a-Flood and Floodline[2]). They said there was a need to automate the use of social media, especially tweets, in real-time map-based dashboards to better understand flooding situations across Scotland. These discussions along with similar needs identified in recent literature (see introduction) provided confidence in the potential usefulness of our approach and enabled refinement of our three research questions.

To achieve our aim and answer these three research questions, we developed a prototype pipeline (Section 2.1.1, Fig.1 and Table 1), and tested its operation during a series of flooding events across England and Scotland, between June 20-29th, 2016 (Section 2.1.2). In Section 2.2, we present a systematic description of the pipeline steps and major design options and decisions. We accessed the online data sources (APIs) of two British environmental agencies: the Scottish Environmental Protection Agency (SEPA) and the Environment Agency (EA). Similar data sources exist for other countries (Table A.1), and other kinds of weather events e.g. wildfires, hurricanes, snowstorms in North America.

2.1 Pipeline structure and its application during actual flooding

2.1.1 Pipeline structure and summary of steps

[1] https://www.floodtags.com Last accessed: 20th Dec. 2017.
[2] https://www.sepa.org.uk/environment/water/flooding/ Last accessed 30th May 2018.

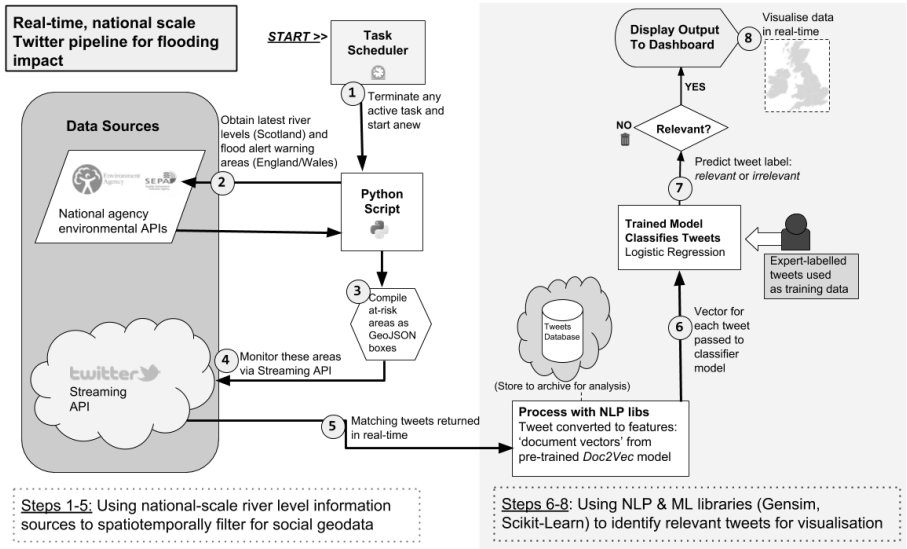


Fig. 1. Steps in the pipeline application architecture for retrieval of social geodata during flooding: *White* background indicates steps for spatiotemporal data retrieval (1-5); *Grey* background summarises filtering steps to predict relevant tweet data to output/visualisation (6-8).

Table 1 Details of the prototype pipeline steps and the research questions addressed by each step.

Research question	Step	Description
1) *How to programmatically identify at a national-scale (across Great Britain) areas at-risk of flooding, based on real-time flood warnings and river level information*?	1	Reset retrieval task at regular periods, i.e. every three hours a script ran to update the latest at-risk areas from online sources (See Section 2.2.1 for details of period duration).
	2	Obtained at-risk areas from national-scale environment data sources, namely via SEPA river gauge levels for Scotland and EA flood warnings. SEPA flat files were polled to retrieve all the latest river levels; these were compared to their average level for an indication of flood risk associated with each gauge's surrounding area. The top ranked EA flood warning and alert areas of risk were extracted from results returned by the EA API[3] (see Sections 2.2.1 and 3.2 for details).
	3	Prioritised at-risk areas were collated to a combined shortlist, up to a maximum of 25, as per Twitter Streaming API limits. Areas were

[3] https://environment.data.gov.uk/flood-monitoring/doc/reference#flood-warnings Last accessed: 30th May 2018.

		converted to bounding boxes, as required by this API.
2) *How to automate the spatiotemporal retrieval of real-time social geodata using Twitter APIs?*	4	A Streaming API location filter query monitored geotagged tweets with the at-risk shortlist from Step 3. (See Table A.2. (i). Queried areas were updated throughout: see log Table A.2, (vi)).
	5	Matching social geodata were returned in real-time, totalling 420,218 tweets for 20-29th June 2016. Tweets matched when their geotag intersected an at-risk area in the *location* filter at the time the tweet was posted. Tweets returned by the API were archived to a MongoDB NoSQL database[4], which formed a development dataset. The Streaming API uses a heuristic to determine whether a given tweet falls within a bounding box: ● If the Coordinates field is populated, the values there will be tested against the bounding box. ● If Coordinates is empty but Place is populated, the region defined in Place is checked for intersection against the location's bounding box. Any overlap will match[5]. More detail on this process is provided in Appendix 1.2.3.
3) *How to automatically identify flood-relevant tweets?*	6	The tweet text was pre-processed, providing inputs to a Paragraph Vector model (via Gensim "Doc2Vec" implementation) which embedded the tweets and words as numeric feature vectors. This model could convert (or infer) incoming, unseen tweets as document vectors, for this step to pass to the next step.
	7	Using a tweet's vector, identified if it was of relevance. The classifier was developed by an annotator manually labelled a subset of 4,502 tweets from June 23rd as either *relevant* or *irrelevant*. These were split (randomly, and at 9:1 proportion) into training and test sets to fit a SciKit Learn Logistic Regression classifier. The class label for new, unseen, inferred tweet vectors could then be predicted by the model.
	8	This step outputted relevant tweets to a dashboard, for viewing and exploration by potential stakeholders. We prototyped several visualisations of social geodata on maps at their coordinate

[4] https://www.mongodb.com/what-is-mongodb Last accessed: 20th Dec. 2017.

[5] https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/basic-stream-parameters.html#locations Last accessed: 30th May 2018.

locations (see Figs. 3, 4 and 5).

2.1.2 Applying and testing pipeline during national scale flooding events

There were widespread flooding events during June 2016, due to above average rainfall in Great Britain (147% of the 1971-2000 average), and above average river flows for most of England, Wales and North East Scotland[6]. The initial pipeline stages (Steps 1-5, see Fig. 1 and Table 1), were assembled and then tested in early June 2016 with flooding events across England and North East Scotland, and in northern and central France in late May 2016[7]. Then, these same steps (i.e. Steps 1-5), were applied nationally across Great Britain, running continuously, between 20-29th June 2016[8]. Finally, Steps 6-8 were carried out using the data collected (Fig. 1 and Table 1).

2.2. Description of pipeline steps and major design options and decisions

In this section we describe our research addressing the three research questions; we systematically describe a series of steps and major design options and decisions that enabled production of a prototype machine learning social geodata pipeline, based on real-time flood warnings and river level information.

2.2.1 Steps 1-3: How to programmatically identify areas across Great Britain at-risk of flooding based on real-time flood warnings and river level information?

All steps (see Fig. 1 and Table 1) were controlled by a scheduled pipeline task: a shell script in Python that ran on a local computer server, the script was reset every three hours (Step 1) in order to update areas at-risk of flooding from the latest national, environmental data sources. A period of three hours was chosen as an intended trade-off between tracking the latest at-risk area forecasts (API updates varied between 15 minutes and several hours), and to allow capturing reaction from those areas on Twitter. In a production application, this could be tuned to user requirements and to reflect the nature of pluvial and fluvial flood events: optimising this was not within the scope of this study. Python was chosen as the server-side language since it is widely used on servers especially for data science applications; alternative options at the time included PHP and Ruby. Additionally, it provided the widest range of libraries to construct our pipeline, including those for accessing Twitter APIs and machine learning. This script, and all the project code, were version-controlled using Github- as Git is

[6] http://www.hydoutuk.net/archive/july-2016/further-information-july-2016/ Last accessed: 27th Dec. 2017.
[7] https://www.ecologique-solidaire.gouv.fr/prevention-des-inondations Last accessed: 25th Jan. 2018.
[8] We consulted: http://www.metoffice.gov.uk/public/weather/forecast and see month of June historic summary at http://www.metoffice.gov.uk/climate/uk/summaries/2016/june Both last accessed: 27th Dec. 2017.

a widely used version control system, and Github is a popular repository hosting service for free and open source projects (see Table A.2). Our choice of APIs for real-time flood-related information was limited to those provided by SEPA in Scotland, and the EA in England and Wales. To establish national-scale coverage, the script jointly checked their respective APIs (see Appendix Table A.1): Scottish (i.e. SEPA) river level data feeds and England & Wales (i.e. EA) flood risk areas (Step 2).

a) the script first cycled through a set of SEPA web addresses for river gauge readings, parsing these flat files using Python web scraping code and obtained all the latest river levels (351 gauges in total). Then each reading was compared with its respective cached average level, so as to identify the highest relative levels present. The gauges for these levels then provided (by using their coordinates as a centre-point for 16 km wide square boxes) the locations at-risk across Scotland. This size was arbitrarily chosen to cover the areas nearby to each gauge; future research could explore the influence of this value and ideally tailor it to the morphological characteristics of the rivers.
b) the script then queried the EA's public web API for its summary of latest flood risk warning and alert areas.

These locations were ordered by risk severity, logged to a text file, and then converted to rectangular bounding box coordinates (as per Twitter Streaming API requirements, Step 3). Further investigations would optimise this process, e.g. to establish the ideal spatial range for each box, but that was not the focus of our research. All the SEPA gauge locations were represented by simple polygon boxes in a spatial JavaScript Object Notation (JSON) format[9]. Generating these areas would be improved by accounting for e.g. topography and flood defences. The EA API flood areas[10] vary in size and can comprise multiple catchments. These were converted from complex polygon shapes to simpler containing bounding boxes (see Appendix A1.2.2 and A1.2.3, Figs. A1 and A2).

To generate a shortlist of locations for this task's duration i.e. three hours: we combined the 14 most at-risk SEPA areas with up to 10 at-risk EA areas (warning levels 1-4)[11]. We divided the proportion of SEPA and EA areas arbitrarily (it was expedient to have more Scottish locations due to the use of river levels): future work would explore and improve this proportion. The maximum of 24 at-risk areas was selected at each time point to respect the Twitter API. The current at-risk areas were recalculated once the scheduled task was reset, Fig. 2 illustrates at-risk areas for the period (and Fig. 6 shows those from the 23rd June in Southeast England). See Figs A.1 and A.2 for further details of how

[9] GeoJSON was used - an open standard format for geographic features.
[10] https://environment.data.gov.uk/flood-monitoring/doc/reference#flood-areas Last accessed: 20th Dec. 2017.
[11] For details of EA severity, please see https://environment.data.gov.uk/flood-monitoring/doc/reference Last accessed: 20th Dec. 2017.

the bounding boxes for England and Wales encompass their associated at-risk flood area, as provided by the EA.

2.2.2 Steps 4-5: How to automate the spatiotemporal retrieval of real-time social geodata using Twitter APIs?

To tackle the second research question required two main design choices: how to access the Twitter APIs, and what Twitter API to use. We chose the Python Twython library[12], which is a wrapper for the Twitter APIs, as the rest of our pipeline was in Python. We could have used the Tweepy library[13], as both provided the same functionality and were well documented. In relation to Twitter APIs there were two options: the REST (now called 'Search') and the Streaming. REST Search methods were more suited to singular, specific, queries for tweets over the past seven days, whereas the Streaming API provided a real-time stream of tweets. To complete this pipeline we chose the Streaming API as stakeholders had highlighted their need for real-time information on flooding events, we discuss additional reasons for our use of the Streaming API in the context of other studies in section 3.1. Experimentation with both Twitter Search[14] and Streaming APIs to retrieve flood-related tweets from Great Britain (and France) was done during May and early June 2016. An example from this experimentation (Fig. A.2) demonstrates a monitored at-risk location intersecting with a geotagged tweet. A web-based console[15] was used to test Twitter API queries and view results. A visualisation tool- GeoJSONLint[16] was used to inspect the polygons.

We monitored solely georeferenced tweets, filtered by the shortlisted areas, which retrieved tweets whose geotag intersected: a second Python script (see (i) in Table A.2) queried the Twitter Streaming API for the prioritised list of locations (Step 4) - a maximum of up to 25 location bounding boxes were permitted at a time[17]. These locations were monitored for any newly posted tweets whose geotag intersected a bounding box, which constituted a "match": location queries used the */statuses/filter* endpoint of Twitter's Streaming API, and tweets' *Place* or *GPS coordinates* (as according to a user's choice for the tweet[18]) were matched against, with priority given by the API to coordinates when

[12] Twython - Twitter API wrapper library in Python: https://github.com/ryanmcgrath/twython Last accessed: 16th Nov. 2018.
[13] Tweepy- Twitter API wrapper library in Python: https://github.com/tweepy/tweepy Last accessed: 16th Nov. 2018.
[14] https://developer.twitter.com/en/docs/tweets/search/overview Last accessed: 20th Dec. 2017.
[15] Website to experiment with public APIs https://apigee.com/console/twitter Last accessed: 20th Dec. 2017.
[16] GeoJSON is a JSON-based, open standard format for geographic features; "linting" helps ensure valid format http://geojsonlint.com Last accessed: 20th Dec. 2017.
[17] https://developer.twitter.com/en/docs/tweets/filter-realtime/overview ; e.g. a *filter* endpoint of the form https://stream.twitter.com/1.1/statuses/filter.json?locations=-74,40,-73,41 Last accessed: 20th Dec. 2017.
[18] See Adding your location to a Tweet | Twitter Help Center. https://support.twitter.com/articles/122236 [Accessed 20 Mar. 2017].

possible (Appendix 1.2.3 Fig. A2 and Appendix B for their specifics). The script then stored the retrieved tweets[19] (Step 5) in a MongoDB database. MongoDB was chosen as it was a widely used open-source document database, ideal for storing tweets in JSON format, with helpful geospatial indexing and querying and full-text searches.

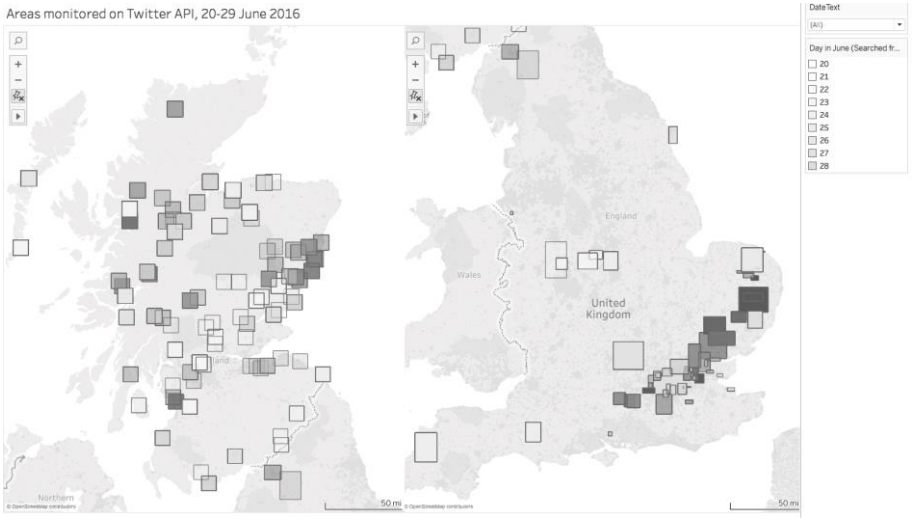


Fig. 2 All 163 unique at-risk areas, as sourced from two British national agencies' data APIs, and queried via Twitter Streaming API 20-29th June 2016. Catchment polygons from EA risk areas were converted to containing bounding boxes - i.e. rectangular polygons, as required by Twitter API[20]. Box shading gives corresponding date from which an area was first monitored, and size indicates actual bounding box for each location. Fig. 6 shows a subset of these boxes in more detail.

This pipeline application was in active operation between 20-29th June, it consisted of Steps 1-5 and stored tweets to our database, providing a real-world dataset, which we then explored and used for development and testing of the remaining pipeline steps. Additional database queries were run on the collected tweets to facilitate time/date queries, and convert tweet georeferences to valid formatting for performing geospatial queries. Data exploration of the various days (heaviest rainfall was on 23rd June) via basic text query searches (e.g. "flood", "storm"), demonstrated the retrieval of social geodata from the at-risk areas and gave a general indication of the proportion of potentially relevant

[19] https://developer.twitter.com/en/docs/tutorials/tweet-geo-metadata and https://twittercommunity.com/t/streaming-filter/51132 Last accessed: 20th Dec. 2017.
[20] https://developer.twitter.com/en/docs/tweets/filter-realtime/api-reference/post-statuses-filter.html Last accessed: 20th Dec. 2017.

tweets (Table 3). Queries to add centroids to a tweet's "*Place"* geotag enabled creation of map-based visualisations.

2.2.3 Steps 6-8: How to automatically identify flood-relevant tweets?

To achieve this there were three main design choices: the first related to how to identify flood-relevant tweets (Steps 6 and 7), the second was which machine learning classifier to use (Step 7), and finally what web-mapping library to use to display the tweets on a dashboard (Step 8).

The approach presented here does not use keywords to query the Twitter API but dynamic shortlists of locations. Keyword queries have disadvantages, and they retrieve only a fraction of all disaster-related tweets (Nazer et al, 2017). Some have recommended initial retrieval of broader "contextual streams" for crisis tweets, with later filtering stages (Palen and Anderson, 2016). Analysis of tweets during River Elbe floods concluded that using keyword queries is dependent on using the "right" terms and that distance-based prioritization could provide better filtering of relevant tweets (Herfort et al, 2014). Dispensing with maintaining set keyword searches, and increasing automation of identifying relevant tweets is still in its infancy (Caragea et al, 2016). Further work is required on how collection strategies impact the data obtained and analytic results. For instance, selecting messages in the geographical region affected by a disaster vs. selecting messages based on a keyword-based query may return datasets having different characteristics (Imran et al, 2015). Location filtering as a first step may reduce misclassification of off-topic tweets, which is an issue for keyword-based retrieval (de Albuquerque et al, 2015).

We chose to use Paragraph Vector (or "Doc2Vec" [21]) embeddings to provide a practical way to learn dense word and document vectors from large datasets, enabling applications such as distributed representations of varying length tweet texts, along with additional benefits of capturing word order and reduced pre-processing. The approach was chosen to represent tweets because of its generation of dense feature vectors for such short texts, thus computationally beneficial, which suited our goal of a near real-time pipeline. The more conventional Latent Dirichlet Allocation (LDA) was understood as less performant than word embeddings with short texts, suffering from data sparsity (Sridhar 2015). Also, word embeddings had been favoured over LDA in recent research into crisis-related Twitter context (Imran et al, 2016).

To automatically identify if any development dataset tweets were relevant to flooding, we wrote a set of Python scripts (see Table A.2), which leveraged the machine learning, NLP capabilities of Gensim and SciKit Learn libraries. We chose Gensim, as at the time it was the main Python library for

[21] Paragraph Vector approach is implemented as Doc2Vec in Gensim https://radimrehurek.com/gensim/models/doc2vec.html Last accessed: 20th Dec. 2017.

Paragraph Vector embeddings, was open-source and well documented. The pipeline Steps 6 and 7, attempted to identify tweets as relevant output for visualisation in Step 8. Step 6 performed unsupervised learning, via the Paragraph Vector approach (Le and Mikolov 2014), obtaining a feature vector for each tweet (treated here as "documents"), and in Step 7, these generated vectors were inputs for supervised learning by a SciKit Learn classifier, which enabled automatic labelling of new tweets as relevant or not.

A trained Doc2Vec model generated an embedding of the documents and words in a "feature space", i.e. each tweet was represented within an n-dimensional space as a vector of real numbers. The generated document and word embeddings, respectively, encode both semantic and syntactic relationships. This was achieved by using a window of surrounding context words (along with a token to represent the document itself) in order to predict target words. Extending its predecessor's approach (Word2Vec), a separate matrix is used in Doc2Vec for encoding documents as well as the various words; the latter matrix is treated as a shared vocabulary between all documents (see diagram, Appendix Fig. A.5, which shows one of the two algorithms, used in Doc2Vec). As in Word2Vec, each unique word's vector, a column in matrix W, begins as a random n-dimensional vector, which is later "learned" by the context of its surrounding words. Doc2Vec also learns a document vector, a column in matrix D, per unique document identifier ("tag" in Gensim) - thus, iteratively learning a vector for a document, as opposed to a combination of word vectors. Training is done by an artificial neural network, for k times (termed "epochs"), which tries to predict words from their context (window of a specified length). The weights learned by the trained neural network abstractly represent the meaning of the words and documents.

Step 6 firstly, pre-processed the tweets received from the Streaming API filter, as required for valid input to Gensim's Doc2Vec. This involved normalising each tweet's text (converting to lowercase and removing most punctuation); then split it by whitespace into "tokens" (and since they provide context, *stopword*[22] tokens were kept); emoji[23] characters were kept as they arguably form meaningful embeddings akin to words; hashtags (a common social media convention) were kept, and the "#" character removed; numeric digits, URLs, @mentions (and usernames) were stripped out for simplicity and research scope time constraints. The processed tweets were each tagged with a unique document identifier and exported to a CSV flat file.

Next, the pipeline passed (one-off procedure) the resulting "document corpus" (i.e. the flat file), which consisted of all 420,218 tweets (from 20-29th June, given in Appendix A1.2.5), as input to the

[22] https://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html Last accessed: 20th Dec. 2017.
[23] emoji are pictorial symbols widely used in computer-mediated communication

Doc2Vec training algorithm, with each tweet being treated as a separate paragraph or 'document'. Doc2Vec generated a model of numeric feature vector representations for each tweet/document and for the words (i.e. the separate word-tokens). The document vectors were essentially an averaging of the words in the tweet. As recommended by the technique's originators (Le and Mikolov, 2014), we combined both available Paragraph Vector algorithms: Distributed Memory and Distributed Bag-of-Words. Distributed Memory used the document vector and previous words, and can capture word order to an extent. The latter used the document vector to predict the words for the document.

Gensim Doc2Vec also has several parameters which can affect the results. We manually varied these and found the best results used n=160 dimension vectors (combining both algorithms this meant n=320 vectors). Other parameters were set as follows: window size of 10 (the maximum distance between a predicted word and its context in the document); minimum count of 4 (i.e. words occurring less than this in the vocabulary was ignored) and sample was 1e-4 (the threshold for configuring which higher-frequency words were randomly downsampled). We also experimented with the number of training epochs - how many times all training vectors are employed to update the learned weights - ranging from between 12 to 40, and our best result (see Section 3.4) used 24 epochs. Further details of the Doc2Vec training are in Appendix A1.3. The cosine similarity measure (see Appendix 1.3.2) identified tweet and word vector similarities, which were useful to explore the model e.g. for the term "lightning", the most similar word vectors were[24]:

```
[('lightening', 0.6080838441848755), ('storms', 0.4550410509109497),
('lighting', 0.42861834168434143), ('thunderstorm', 0.38076531887054443),
('thunder', 0.36179378628730774), ('storm', 0.35902613401412964),
('pouring', 0.35672375559806824), ('biblical', 0.33286309242248535),
('shook', 0.3284866213798523)]
```

This example illustrates tweet tokens' idiosyncrasies being encoded automatically in the model: including misspellings and related words.

Once the Doc2Vec models were trained, they were saved to disk, which permitted later loading to memory and inferring vectors for new, unseen tweets in an operational pipeline. Vectors were stored as Numpy[25] array structures, which permitted direct input to a classifier by the SciKit Learn library (Step 7).

In Step 7 these document vectors provided feature representations of the tweets' text, suitable to train a classifier, (once we had labelled the class for a subset of them), i.e. perform supervised learning by a standard algorithm from SciKit Learn. We chose the SciKit Learn library, since in 2016 it was the

[24] https://radimrehurek.com/gensim/models/doc2vec.html#gensim.models.doc2vec.Doc2Vec.most_similar Last accessed: 20th Dec. 2017.
[25] A Python extension package providing efficient support for multi-dimensional arrays http://www.numpy.org/ Last accessed: 20th Dec. 2017.

main Python machine learning library and provided access to a range of well-documented machine learning algorithms.

Logistic Regression was chosen as it is an effective discriminative model in two-class (binary) settings; for example, it performed well with word embeddings in: i) labelling relevant tweets of landslide events (Musaev and Hou, 2016), and ii) a semantic relatedness task (Vosoughi et al, 2016). In a study classifying flood-relevant UK tweets (Spielhofer et al, 2016), Naive Bayes performed worse than Logistic Regression with a class-imbalanced dataset (as our dataset exhibited). Next, we trained the classifier model, and used the unseen test set of tweets to evaluate its performance. As the focus for our research was on feasibility, rather than optimising all parameters, we did not use grid-search or k-fold cross-validation here. Nonetheless, we should have followed standard conventions in developing the classifier, and done so[26]. Instead, we used a 90/10 split of training to test cases because we had a relatively small labelled dataset as recommended in Dobbin and Simon, (2011). Our resulting classifier model was evaluated using a confusion matrix and a Receiver Operating Characteristic (ROC) curve.

First, a model was trained using a set of 4,502 tweets which had been manually labelled as 'relevant' or 'irrelevant' (section 2.2.4 explains these criteria and describes developing the classifier). Of these, 414 tweets were labelled 'relevant', which we took as sufficient as other studies using NLP to classify tweets for situational awareness used comparable numbers (e.g. Sen et al, 2015). We experimented with Doc2Vec parameters of between 100 to 400 dimension feature vectors, inline with other studies (e.g. Word2Vec as used in Nguyen et al, 2016).

In terms of processing time for the pipeline, the one-off cost for training (fitting) of the Doc2Vec models was dependent on the vector size and the number of epochs. This was done using a Macbook Pro i7 2.4GHz, 8Gb RAM for our best result, which took 10 hours (faster, current hardware would improve this considerably). Then fitting a Logistic Regression model required several minutes (a larger labelled dataset would require longer). In an operational pipeline, therefore, tweets can be retrieved in close to real-time, inferred as vectors by the fitted Doc2Vec model in seconds, and passed to the classifier model for output to a map visualisation (Step 8), which required seconds per tweet, for a mid-performance PC. Thus, the delay would be small, in the order of seconds to a minute, between a tweet being posted by a user and plotting it visually for stakeholders.

[26] E.g. See researcher Ng, Andrew recommendations: http://cs229.stanford.edu/notes/cs229-notes5.pdf

In Step 8 the final step of the pipeline was to demonstrate the potential to output tweets in a live setting to a geospatial dashboard. In production, this would occur after incoming, unseen tweets were first inferred by the trained Doc2Vec model, and then passed to the Logistic Regression classifier. We did not attempt this in a live production setting. However, we would now be able to use our final models as the basis of an operational pipeline in Great Britain for flooding. The tweets predicted as relevant, would then be stored in a database (e.g. MongoDB) and plotted via either an interactive Web mapping library like Folium/Leaflet[27] or a business intelligence tool like Tableau Desktop[28]. In this demonstration, we generated visualisations (see Figs. 3, 4, and 5) which used tweets from the development dataset captured. We chose a web mapping library over a tool like Tableau to enable a more integrated web-app to be produced. In 2016, there were a range of JavaScript web mapping libraries including Leaflet and OpenLayers[29]. Leaflet was chosen as it was widely used, well documented and we did not require all of OpenLayers functionality. Folium was a widely used open-source Python wrapper for Leaflet.

A Python script ((vii) in Table A.2) collated tweets from MongoDB and visualised them with Folium (see Section 3.1, Fig. 5), which generated a prototype dashboard. This used OpenStreetMap basemap tiles, and plotted each tweet based on its geographic coordinates, with an indication when the centroid of the Place bounding box or specific coordinates were used. Each tweet was represented by a clickable marker, which provided a pop-up box of the tweet when scrolled across. Alternative interactive map visualisations of retrieved geotagged tweets were plotted in Tableau, in order to confirm dashboard potential and show the varying granularity of geotagged tweets (i.e. the associated geotag Place box dimensions). Fig. 3 shows an illustrative selection of tweets from the period and Fig. 4 shows tweets from the day of highest rainfall (23rd June).

[27] Python wrapper of commonly used Leaflet open source interactive mapping library https://github.com/python-visualization/folium ; http://leafletjs.com Last accessed: 20th Dec. 2017.
[28] http://tableau.com Last accessed: 30th May 2018.
[29] https://openlayers.org/ Last accessed: 16th Nov 2018.

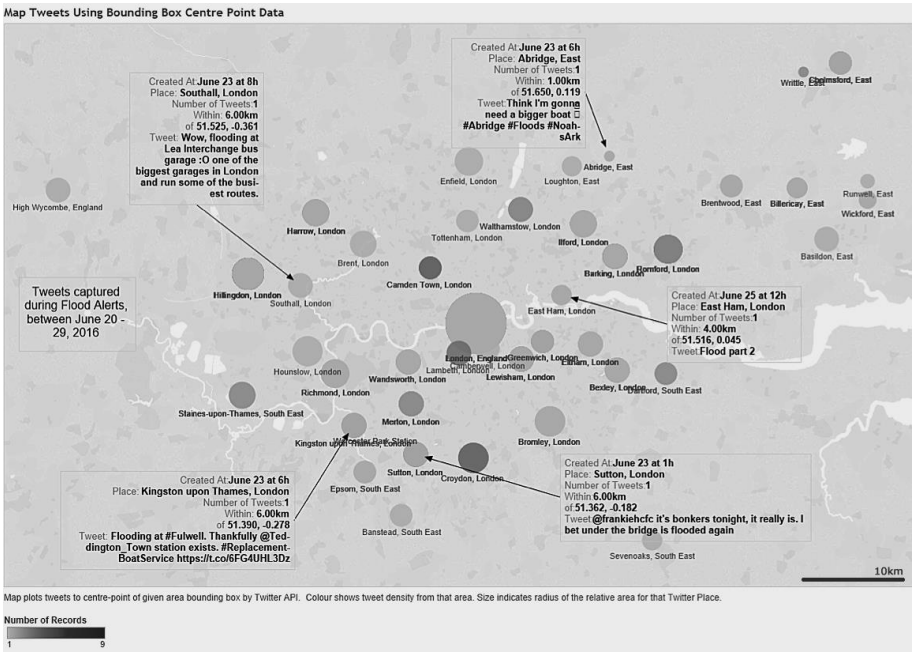


Fig. 3 Potential dashboard visualisation example using a selection of tweets from the development dataset. Circle diameter varies to reflect the tweet Place area size.

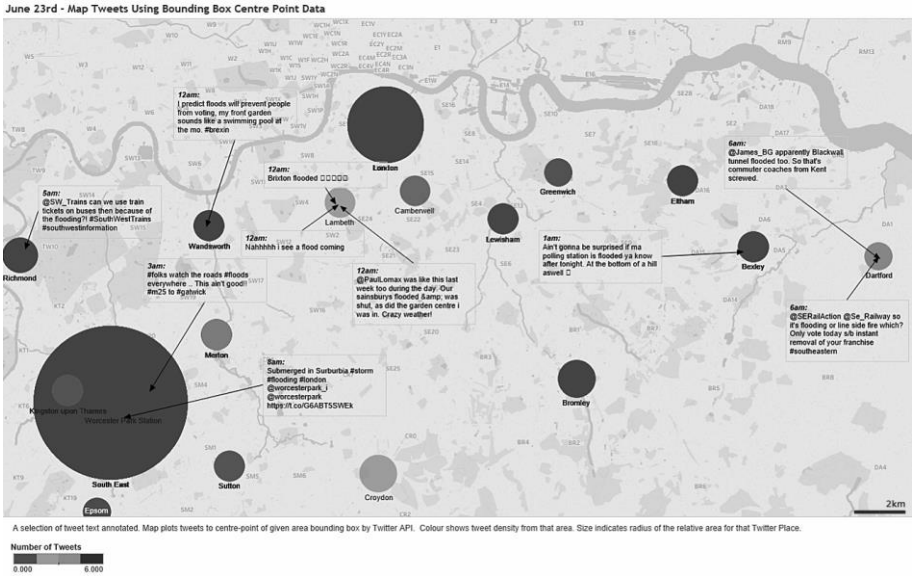


Fig. 4 Some tweets from 23rd June as another illustrative dashboard prototype. Circle diameter varies to reflect the tweet Place area size.

2.2.4 Developing a binary classifier to predict relevant tweets

To train a binary classifier to distinguish relevant from irrelevant tweets (see Section 2.2.3, Step 7), a Logistic Regression model was trained to predict the data's class probabilities from a class-labelled subset of the development dataset (we provide this dataset in Appendix 1.2.5). Table 2, below, presents five actual tweets, along with their geotag, which were retrieved on the 23rd June and labelled as relevant.

Table 2 Examples of tweets as manually labelled as "relevant" by the author (annotator). Full labelled set is given in Appendix 1.2.5

ID	Tweet Status Message	Time/Location Geotag
745887110234791936	I'm guessing our weather had something to do with that. #safetravels	8:51 AM - 23 Jun 2016 from Basildon, East
745907644641193984	Wtf was there thunder last night?????	10:13 AM - 23 Jun 2016 from Loughton, East
746027965344014336	Just got to love camping in the UK! Noah's ark has got to be due a visit to Stubbers tonight. This is getting silly	6:11 PM - 23 Jun 2016 from London, England
745905001579581441	jealous - our view! STRANDED!	10:03 AM - 23 Jun 2016 from Greenwich, London
745985316566204420	Here comes the storm! #Portsmouth #Chichester #BognorRegis	3:22 PM - 23 Jun 2016 from South East, England

The subset that was labelled were first randomly selected from the tweets retrieved on 23rd June, as it had the most flood related tweets (see Table 3 below; and exploratory data analysis and supporting information e.g. news reports). This was done to try to ensure sufficient tweets of the more rare relevant class would be among the labelled sample, to reduce difficulties of the class imbalance. It comprised 4,502 tweets (which were randomly split 9:1 into a training and a test set). We chose an amount in the thousands as recommended by a survey of tasks in similar contexts (Imran et al, 2015). The first author of this paper labelled this training set, taking six hours, using a script ((iii) in Appendix Table A2). A relevant label was assigned to tweets which referred to the ongoing adverse

storm weather/flooding events, by applying a set of criteria consistently: mentioned either the weather severity, any flooding status or impact, related travel disruption or sharing flood information with the community. Owing to time constraints of the work, the labelling process used was qualitatively weak, as only one person carried this out, without inter-rater reliability checking, assessed with a statistic such as Cohen's Kappa to ensure consistency of labels. Multiple annotators should be used in future (e.g. as was used for a similar task in Sen et al, (2015)).

3. Results and discussion

3.1 Integrated prototype pipeline for identifying tweets for stakeholders

In this study, we were able to successfully develop and test a prototype real-time pipeline integrating national environmental data sources and the Twitter Streaming API, with later machine learning steps. Using document and word embeddings together with a classifier, we processed relevant, geotagged tweets, as outputs to demonstration visualisations for in-progress flood-risk incidents. Our prototype offers an end-to-end, open source, web-based, pipeline architecture to address the crucial issue of how stakeholders can be better informed in emergency management situations. In particular, by extending and advancing automation techniques for identifying relevant social geodata, i.e. tweets which possess user-assigned geotags, a small, but significant, proportion of all tweets. We demonstrated this with a national-scale flooding case-study and corresponding development dataset. A study of computational methods processing social media crisis data highlighted the need to go beyond reliance upon keyword queries (see Imran et al, 2015; Palen and Anderson, 2016). We attempted this by using Twitter API location filters (other social media APIs such as Instagram and Facebook offer similar filtering - for further investigation). Our research addressed calls for crisis mapping platforms to improve on using known, unchanging locations with the Streaming API, and attempt adaptively monitoring wider spatial areas e.g. country-wide (Middleton et al, 2014). Further development and testing of social geodata machine learning pipelines is required, for example, how best to combine automatic and manual labelling of tweets and make greater use of unsupervised learning (Li et al, 2017). In this study, we have demonstrated the potential of available national level environmental data relevant to flooding, in England this was polygons rated at risk of flooding, and in Scotland there was only access to real-time river level data, which may not provide the best indicator of flood risk. The potential of this machine learning pipeline needs to be demonstrated during a range of different types (and causes) of flooding events e.g. pluvial and fluvial, as well as exploring how real-time rainfall information may provide more precise information of areas at risk of pluvial flooding especially during summer months.

One difficulty facing environmental modellers and software developers when developing complicated web-based modelling/machine learning applications, such as this pipeline, are the range of choices for individual components and steps of their application, and successfully linking these (and the associated dependencies) to work effectively in real-time. In their review of web-based environmental modelling, Vitolo et al. (2015) presented an example of these choices and the difficulty of linking the components.

Monitoring of pipelines with multiple components, like this one, is a key task (see Sadilek et al, 2016; Wu et al, 2016; Ngai 2017). In this prototype, we made use of a task scheduler and logging to text files, which helped isolate issues and improve robustness, which enabled our retrieval of a development dataset of significant size. This should be further automated in an operational pipeline using workflow management and scheduling tools (e.g. Luigi), or analytics tools (e.g. Fabric and Crashlytics)[30]. Setting out the pipeline requirements and maintaining the pipeline build is also important - we used Python virtual environments (Anaconda) and version control (Git). An operational pipeline would use current best practice tools (which are constantly evolving) e.g. containerisation (e.g. Docker) and bundlers/build-tools (e.g. Browserify, Gulp, WebPack)[31].

A key choice developers and researchers need to make is which particular Twitter API to use. In this study, we settled upon the public Twitter Streaming API (rather than the Search API), which provides access to tweets as posted in real-time. Initially, to explore retrieval to dashboard visualisations and to establish the feasibility of a social geodata pipeline, we utilised the Search API[32], which allows short-term historical access only, though lacks the continuous real-time benefits of the Streaming API. We used a series of queries to gather tweets from heavy rainfall events in Scotland, in early June 2016, and plotted these to a web-based dashboard, using a Python interface (Fig. 5) for interactive map library Leaflet.

[30] https://get.fabric.io ; https://try.crashlytics.com ; https://pypi.python.org/pypi/luigi Last accessed: 20th Dec. 2017.
[31] https://www.docker.com ; https://gulpjs.com ; http://browserify.org ; https://webpack.github.io Last accessed: 20th Dec. 2017.
[32] https://developer.twitter.com/en/docs/tweets/search/overview/basic-search.html , accessed with tools: http://www.tweepy.org and https://tags.hawksey.info Last accessed: 20th Dec. 2017.

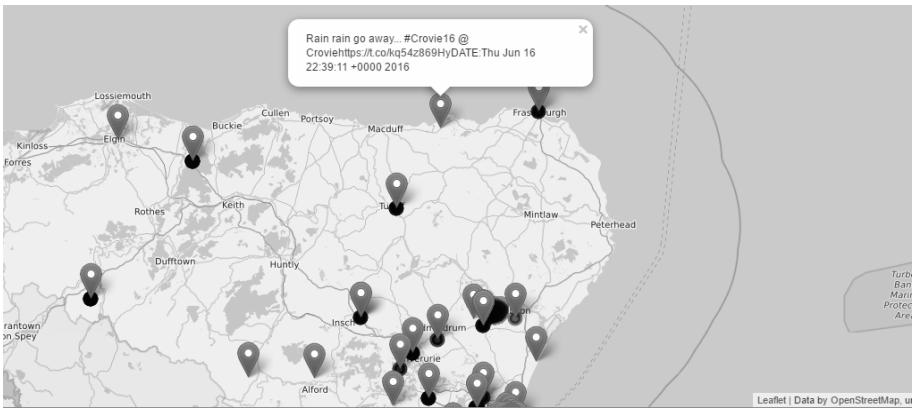


Fig. 5 Screenshot of a web browser displaying a dashboard prototype of geotagged tweets retrieved, post-hoc, via Twitter Search API using specific location searches combined with keywords, (as the Search API permits).

We found that the Streaming API met our aim to maximise the retrieval of geotagged tweets via location filtering, and this finding corresponds with other studies (see Morstatter et al, 2013; Ekta et al, 2017; Tsou et al, 2017). The alternative Search API preferentially relies on a tweet's geotag presence and therefore reverts to the more common (but spatially less accurate) user profile location. The user profile location is manually set by the user, and is quite different to the specific tweet's current and actual location. Moreover, this field often contains noisy and redundant data, which requires later cleaning and geoparsing steps (Alex et al, 2016; Laylavi et al, 2016). Crucially, the Search API permits just a single location per query, and is not suitable for real-time data (Ekta et al, 2017), so an adaptive query of a spatiotemporal window was not possible - as we achieved with the Streaming API. The Streaming API allowed simultaneous monitoring of multiple locations, necessary for real-time, national-scale applications. More research is needed to evaluate if Search API queries could complement data from the Streaming API. For example, a combination of both APIs was implemented in the "Twitcident" application, as applied to a storm event in Belgium (Terpstra et al, 2012).

3.1.1 Case study: development dataset retrieved during real-time flooding events

The pipeline retrieved a development dataset during widespread rainfall and flooding events in Great Britain (see Section 2.2.2). The characteristics of the dataset are summarised in Table 3 e.g. number of tweets retrieved per day. At the outset, to broadly assess whether or not the retrieved data included flood related tweets, we used a series of basic keyword database queries to identify the proportion of relevant tweets (Table 3), some of which were plotted in Fig. 3.

Table 3 Text search queries on database, exploring all tweets collected by our pipeline for England, Wales and Scotland during June 20-29th, 2016. The database query used MongoDB indexed text search, which includes an English language "stemmer"[33] - e.g. "flood" would match "floods", "flooding", "flooded" etc. as well as "flood".

Date-Period	Total No. Tweets	Text Query	Query results	Proportion of Date-Period Total (%)
20.06.16	12341	Flood	8	0.1
		Thunder OR flood OR rain	104	0.8
21.06.16	42115	Flood	28	0.1
		Thunder OR flood OR rain	525	1.2
22.06.16	68497	Flood	46	0.1
		Thunder OR flood OR rain	1003	1.5
23.06.16	55619	Flood	532	1.0
		Thunder OR flood OR rain	1876	3.4
24.06.16	61730	Flood	220	0.3
		Thunder OR flood OR rain	768	1.2
25.06.16	43072	Flood	92	0.2
		Thunder OR flood OR rain	781	1.8
26.06.16	38456	Flood	64	0.2

[33] See https://docs.mongodb.com/manual/reference/operator/query/text Last accessed: 20th Jan. 2018.

		Thunder OR flood OR rain	553	1.4
27.06.16	43502	Flood	39	0.1
		Thunder OR flood OR rain	510	1.2
28.06.16	30032	Flood	18	0.1
		Thunder OR flood OR rain	556	1.9
29.06.16	24854	Flood	13	0.1
		Thunder OR flood OR rain	460	1.9
20.06.16-29.06.16	420218	Flood	1060	0.3
		Thunder OR flood OR rain	7136	1.7

The early hours of 23rd June featured the greatest number of storms and heavy rainfall, especially in Southeast England and showed higher incidence of results for such keywords (Table 3); some of these tweets were shown as a dashboard might display them (see Fig. 4). The tweet dataset included tweets from across Great Britain (Fig. 2 gives the locations monitored on Twitter).

3.2 Programmatically identify areas, at national-scale (within Great Britain) at-risk of flooding based on real-time flood warnings and river level information

In this study we successfully identified a dynamic set of at-risk areas using two very different web-based river level and flood warning API sources from Scotland and England/Wales. This list of prioritised areas was updated every three hours (in future this time period requires optimising), as the environmental information changed.

The at-risk locations were based on the rules set out in Section 2.2.1, and they varied in size and shape (see Figs. 2 and 6). The English flood risk area polygons[34] apply to the Alert and Warning Areas we retrieved and sorted from the EA API. These determined the bounding box dimensions, and ranged from 1.5 km² to 1600 km², which were monitored on Twitter. While Scottish gauge locations set the centrepoint for their corresponding derived squares of equal size. We tracked a total of 163 unique locations over the 10 day period. In this study, our focus was to demonstrate that these locations could be collated, so as to guide social geodata collection, and not to optimise this step. In Scotland, the identified bounding boxes would have benefitted from using river flow direction and catchment characteristics to reflect fluvial flood risk, which would be an advisable improvement. For gauges in Scottish urban areas, these may well be too large, given Twitter usage will be higher – and a box area is checked for overlap by the API with tweet geotags. However, it could be argued that tweets from the general vicinity should be sought as users may geotag as such.

Compared to our location-driven filtering strategy, sourced by environmental data, previous studies have tended to focus on using Streaming API keyword queries to collect relevant tweets: then doing subsequent tweet geoparsing to plot crisis maps (Middleton et al, 2014); or combining satellite flooding data with tweet results to identify flood extent (Jongman et al, 2015); and a real-time flood model framework filtered the Streaming API on keywords or just a single, fixed city location, then used subsequent keywords to check relevance (Smith et al, 2015). Generally, studies recommend integrating environmental and social media data to automatically prioritise messages, and to be flexible regarding availability of sources (Castanhari et al, 2016). By using prior geographical data to spatially filter social geodata, we can parse and reduce the information space in which we must search for relevant information (de Albuquerque et al, 2015). Recent studies have tried automated geographic prioritization of Streaming API tweets for flood risk based on sensor data which showed potential for near real-time filtering (de Assis et al, 2016). In another study, georeferenced tweets returned from the Streaming API were filtered with keywords and a blend of geographic data sources to show areas affected by a flood, at a regional scale (Cerutti et al, 2016).

3.3 Spatiotemporal retrieval of Twitter social geodata using the Streaming API

We found that by monitoring locations on Twitter, we demonstrated that a high number of geotagged tweets from at-risk areas can be retrieved concurrently with a flood event. Using a regularly updated shortlist of 24 at-risk areas, we were able to query the Twitter Streaming API. This resulted in 420,218 tweets (see Table 3 for a daily breakdown). Other studies have monitored the Streaming API in an emergency response setting (e.g. Middleton et al, 2014; Kryvasheyeu et al, 2016) and

[34] https://environment.data.gov.uk/flood-monitoring/id/floodAreas Last accessed: 30th May 2018.

occasionally using location queries (Laylavi et al, 2016; Smith et al, 2015). The latter two studies both used single, static location search on flooding events for a single city (Sydney and Newcastle respectively). Official authorities have used location queries for emergency management, but only a single location at a time (e.g. Tsou et al, 2017). To our knowledge, a national-scale pipeline strategy using environmental data to automatically track multiple risk locations (regularly updating these) with the Streaming API has not been attempted before for emergency contexts (Fig. 6). Indeed, this spatial query strategy could be applied in other countries and other weather related emergency situations (e.g. wildfires and hurricanes; see Table A.1 for potential data sources).

Our dataset of 420,218 tweets was retrieved by dynamically querying tweets filtered by multiple locations over the 10 day period. The spatiotemporal querying of tweets is given in Fig. 6, where shaded boxes show the areas at-risk of flooding for Southeast England, as supplied by the EA API for the morning of 23rd June. The varying set of locations reflected the updating risk priorities over the time period.

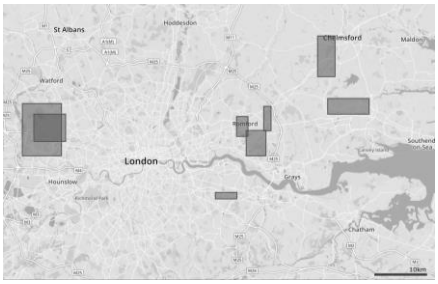
	
23rd June, 1.30 am	23rd June, 4.30 am
	
23rd June, 7.30 am	23rd June, 10.30 am

Fig. 6. At-risk flood areas sourced from environmental data from June 23rd (day of greatest rainfall). Dark boxes indicate areas at-risk of flooding for Southeast England, as per Environment Agency API. See Fig. 2 for an overview of all monitored areas 20-29th June, and Fig. 4 plots some tweets retrieved from these areas.

The Streaming API heuristic to match geotagged tweets to locations is given in Table 1, Step 5 and Section 2.2.2. The data quality of a tweet's geotag exhibits variation and associated error distance when reliant on the tweet's "Place". We investigated the proportion of tweets which had precise coordinates as opposed to those with less precise Place information. Of all the retrieved tweets, a majority (91.9%) had a geotagged Place bounding box, rather than a precise (i.e. GPS) point-coordinate georeference (8.1%). Analysis of tweet location based on Place showed a mixed level of spatial precision (see Appendix 1.1.4, Figs. A.3 and A.4). About two thirds of the tweets had a small Place bounding box diagonal, of between 1-10 km (64%), but there were also many (25%) with a Place covering a larger area of 100-500 km e.g. East England. Furthermore, apart from the variations in shapes and size of Places, a tweet's Place is by its nature, ambiguous: the documentation states "When present, indicates that the tweet is associated (but not necessarily originating from) a Place"[35]. Also, as of writing, the Places presented via the user interface are regionally dependent, with Twitter using Foursquare data predominantly in the US and Canada, and Yelp data in the UK and Japan[36]. For more detailed information, see work by Laylavi et al. (2016) who found evidence that Place can contribute to accurate location inference.

Extracting valuable location information from tweets i.e. geoparsing is problematic (Eilander et al, 2016). Streaming API tweets retrieved via location filter, were all (as a consequence) geotagged and varied in terms of spatial precision (having either less exact "Place" metadata or GPS coordinates). Analysis of tweets during flooding (River Elbe) indicated messages within 10 km of severely flooded areas had a much higher likelihood of being related to such events (de Albuquerque et al, 2015).

3.4 Automatically identifying relevant tweets

Since all tweets posted from the at-risk areas were returned, a large number of these tweets were not relevant to understanding the flooding situation and aiding decision-making (see Table 3); recent studies have recognised the need to improve identifying relevant and informative social geodata for decision-making (de Assis et al, 2016; Rosser et al, 2017; Caragea et al, 2016). We successfully used recent advances in NLP, document and word embeddings (via Gensim Doc2Vec), in combination with a machine learning predictive classifier to automatically identify tweet relevance.

---

[35] https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object Last accessed: 26th Jan. 2018.

[36] See https://twittercommunity.com/t/foursquare-location-data-in-the-api/36065 Last accessed: 26th Jan. 2018. ; https://techcrunch.com/2016/04/22/twitter-integrates-with-yelp-for-location-tags-in-the-uk-and-japan-bypassing-foursquare/ Last accessed: 26th Jan. 2018.

To evaluate the classifier, we used the labelled tweets (n=4502), training used 90% (n=4052), and was tested with a 10% (n=450) hold-back set, and achieved an accuracy of 95% in automatically predicting tweet relevance. Due to all tweets being analysed from the at-risk areas, there was a class imbalance issue, i.e. 9% of the tweets were positive "relevant" class, so potentially an accuracy of 91% could be achieved by a model predicting all tweets as "irrelevant". We were interested in minimising misclassified relevant tweets (often called Type II errors), and therefore retrieving primarily excellent true positive rates (i.e. recall) without sacrificing much positive predictive value (PPV, i.e. precision). The classifier scored 60% recall and 79% precision (Fig 7).

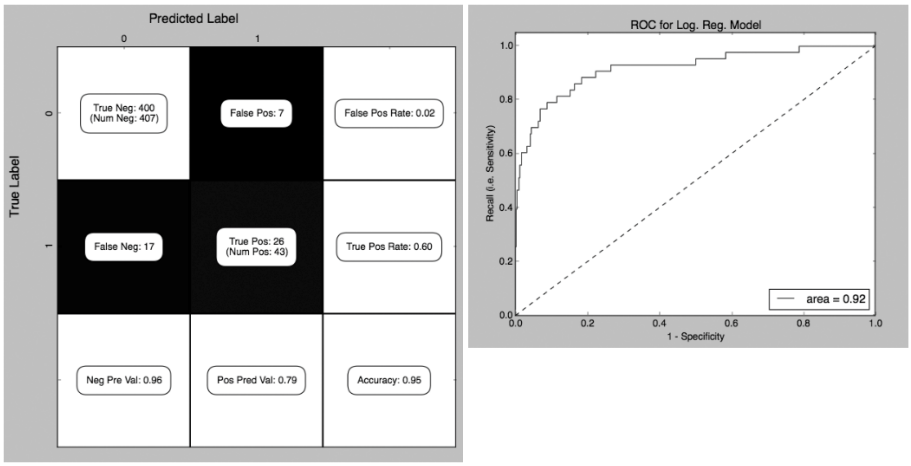


Fig. 7 Confusion Matrix for Logistic Regression classifier of relevant tweets, shows test set predicted and ground truth, 1=Relevant Class, 0=Irrelevant Class. Also ROC Curve showing AUROC=0.92 for fitted Logistic Regression model of relevant tweets. The dashed diagonal indicates a random predictor, 0.5 AUROC.

The ROC curve in Fig. 7 shows the classifier's true positive rate vs. false positive rate (i.e. 1 - specificity) across various thresholds. As a summary of the discrimination ability of the fitted Logistic Regression model, the ROC Area Under Curve (AUROC) achieved was 0.92. This is much better than a random predictor (AUROC=0.5) and a high probability that the classifier will rank any randomly selected "relevant" tweet higher than a randomly selected "irrelevant" one. It is a more representative metric of the classifier than the accuracy score, given the class imbalance (see Fawcett, 2006).

Although Paragraph Vectors have not been used previously with flooding situations, related Word2Vec was used in finding relevant tweets in landslides (Musaev and Hou, 2016) and word embeddings were trained on crisis event tweets (Imran et al, 2016). Other recent NLP techniques such

as Convolutional Neural Nets (CNN) have performed binary supervised classification to predict informative tweet messages in a flooding context using a dataset from CrisisLex (Caragea et al, 2016), and CNN were combined with word embeddings to also benefit from automatic feature extraction and minimal feature engineering for supervised classification of crisis tweets (Nguyen et al, 2016).

NLP libraries based on word embeddings and their variants, such as Glove, have readily available implementations in Gensim, FastText and SpaCy[37], and are practical computationally, as well as competitive in a variety of tasks, compared with more traditional NLP methods[38], e.g. Bag Of Words and one-hot vectors. For example the latter does not capture semantics or word order, and leads to high-dimensional and sparse data (Chen et al, 2016). The use of unsupervised techniques like Paragraph Vectors are thus a promising way to overcome the issue of information overload which impedes social media data adoption in operational settings (Li et al, 2017).

The performance of our classifier, like all supervised learning tasks, was limited by the size and coverage of the training dataset. With a larger (and more robustly annotated) dataset of labelled tweets, our classifier could be improved in its real-world efficacy, and the use of learning curves is recommended to assess the influence of labelled sample size on classification performance (Figueroa et al, 2012). In general, each flooding (or other emergency situation) event has its own characteristics and training models on one event will see a reduction in accuracy/performance for another event (Nazer et al, 2017). The machine learning algorithm steps could be improved by preprocessing, to identify locations in text via Named Entity Recognition, and then weight such features accordingly, as recommended by others (e.g. Sen et al, 2015). Implementing a conventional document classifier based on Latent Dirichlet Allocation (LDA) topic models (see Xing et al, 2014) and also a CNN-based model (see Nguyen et al, 2016) would help assess performance. Experimentation with other libraries' word embedding approaches (which have improved since 2016), e.g. that of SpaCy or FastText would be worthwhile; this would be possible with the modular pipeline presented by adapting Step 6 alone.

An innovation of this study was the programmatic integration of the near real-time environmental data with social geodata (i.e. guiding the collection of flood relevant tweets), and then automatically processing this with later machine learning steps to filter further for relevance. Others have noted the challenges to integrate VGI (such as social geodata, which is non-standardised - semantically inconsistent - and non-authoritative) with existing reliable data sources to make it a useful data source for flood risk management (Castanhari et al, 2016). Consequently using social geodata should be

[37] See https://github.com/stanfordnlp/GloVe; https://github.com/facebookresearch/fastText ; https://spacy.io
[38] E.g. see Alvarez, J.E. and Bast, H., 2017. A review of word embedding and document similarity algorithms applied to academic text (Doctoral dissertation, University of Freiburg).

subject to quality assurance. For our approach this could include labelling of the training dataset with direct stakeholder input, and the visualisation steps should feature interactive user feedback on output tweets.

We believe the approach should work for other extreme weather events. We successfully retrieved abundant tweets based on NOAA/NWS API weather alerts (Appendix 1.1.1), during snowstorms in north-eastern USA, early 2017, applying Steps 1-5 of our approach (below, Fig. 8). By restricting a bulk visual plot to those tweets with photographic media, thus foregoing NLP steps, the interactive map promisingly documents the snowstorm's progress (available online[39]).

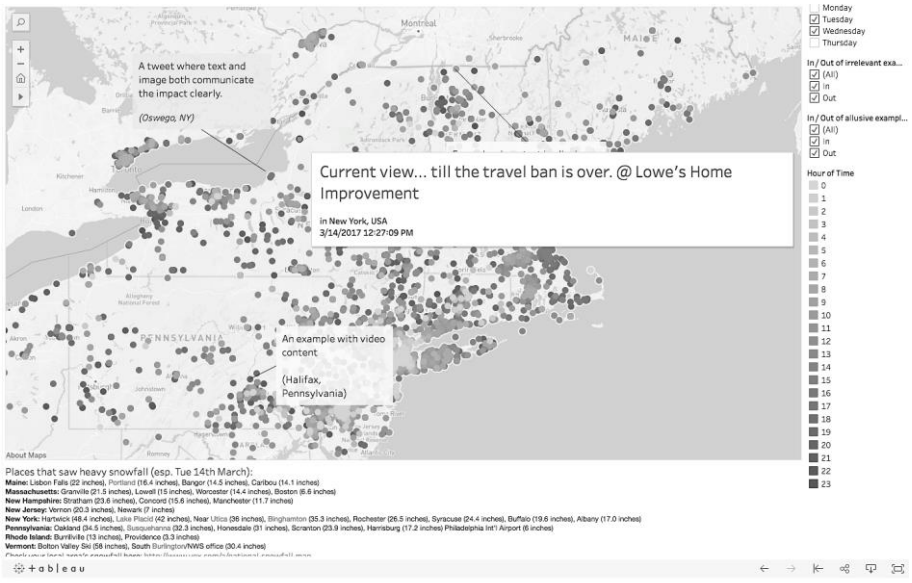


Fig 8. Interactive visualisation demonstration plotting tweets collected using Steps 1-5 only, and adapted to American weather alerts as a basis for Twitter Streaming API location search during snowstorms in March 2017.

For a production application of this pipeline, it would be necessary beforehand to have trained a Doc2Vec model and a corresponding classifier with labelled tweets (see Section 2.2.4, Steps 6 & 7). These fitted models would then be passed individual, incoming tweets from the active pipeline (i.e. tweets from Step 5 go to a Doc2Vec model to infer the tweet's vector values; then the classifier to identify relevant tweet-vectors). There are some limitations to this: inferred vectors with words not in

[39] https://public.tableau.com/profile/luke.barker#!/vizhome/raw-stella-insta-tweetpics/Pre-filteringGPSpointtaggedwithinstagramonly Last accessed: 30th May 2018.

the trained model's vocabulary would have those particular words ignored. This could be alleviated using a large tweet corpus (as we did), during training. The time taken by the pipeline application to identify incoming tweet relevance (i.e. first inferring individual tweet vectors and then classifying each), would be dependent on tweet volume and the online system performance. We appreciate there is a stakeholder need for near real-time information in crises situations (Imran et al, 2015). For this pipeline approach, the time between a tweet going online and visually plotted for a stakeholder as being potentially relevant (in terms of location and content) would be in the order of seconds to minutes, since this would be handled via steps 6-8 of our pipeline: Automatically identifying relevant tweets with open-source NLP and machine learning libraries.

3.5 Limitations of our prototype and further work

This novel demonstration and testing of a national-scale social geodata pipeline could be improved in several ways. In the method section we highlight design decisions that need to be optimised e.g. scheduling of scripts (2.2.1) and size of bounding boxes (2.2.2). In the future, how to set the bounds of at-risk areas requires further attention and evaluation, given this affects which tweets intersect bounding boxes and yield a match. In an urban setting, increasing this area would likely include significantly more tweets, and the chosen proximity will depend on requirements of the application. Enabling a border threshold level to be set by application users would be useful to widen or narrow the overlap with tweets in the locality, as preferred - similarly for other data layers such as population density. Furthermore, people do not always tweet "on the spot", but often nearby or afterwards. Leveraging further multiple information sources would likely also be beneficial (Castanhari et al, 2016), e.g. include observation of rainfall data as well as flood risk areas. Though our aim and research questions were produced with operational flooding stakeholders, we have not yet fully evaluated how it can improve their situational awareness; in part, as designing and analysing an evaluation by stakeholders is a large research project in its own right.

4. Conclusions

Commented [KM1]: At least some of this section needs to be deleted, as moved to the methods. Unless substantial points (the SEPA one about images is not relevant/impt enough), then delete.

In this study we have demonstrated a prototype social geodata machine learning pipeline that integrated real-time environmental data (river levels and flood warnings) at the national-scale and recent advances in word embedding NLP to identify flood relevant tweets. This prototype was shown to work during a 10 day period of flood events across Great Britain with over 420,000 geotagged tweets collected from 163 dynamic, potentially at-risk areas. This work contributes to calls to improve crises situational awareness by automatically identifying tweets using Paragraph Vectors and a logistic regression based classifier. We demonstrated an adaptable and flexible solution for how to

successfully link up multiple steps using open-source libraries, in a continuous and timely manner for sensemaking output. We worked with operational flood stakeholders to revise our aim and three research questions. In this paper we set out the major design choices and decisions made, we appreciate these discussions could be far longer but believe our research and choices are more transparent and reproducible than many other similar studies. We acknowledge there is a need to evaluate pipelines like ours with stakeholders. There is potential to apply our approach in other countries and to other emergency situations. The approach is, likewise, applicable to other languages (not only English) as word-embeddings NLP techniques are, future work should establish this, given the idiosyncratic nature of Twitter messages.

Acknowledgements

Author contributions: L.B. and C.M. designed the research; L.B. performed the research and analysed the data; and L.B. and C.M. wrote the paper. The research developed from a project for University of Dundee MSc. Data Engineering programme, 2016. We would like to thank Andy Cobley for his joint-supervision of the MSc. project, and Mark Wilkinson for providing comments on this paper. The authors are grateful to colleagues from SEPA for early discussions related to this project and also to the Data Lab, Edinburgh who sponsored the MSc. The Scottish Government's Strategic Research Programme enabled C.M.'s contributions. We would like to thank three reviewers, as their comments helped improve the paper.

Funding Sources

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Appendix A. Supplementary data and materials

A1.1 Environmental Data Sources and Datasets

Appendix 1.1.1 International Environmental Data Sources

In this study, in Scotland we used: http://apps.sepa.org.uk/waterlevels; and in England and Wales: https://environment.data.gov.uk/flood-monitoring/doc/reference.

Country	Agency & Resource	URL

Great Britain (Scotland)	SEPA. Latest river levels via gauge readings.	http://apps.sepa.org.uk/waterlevels
Great Britain (England and Wales)	EA. At-risk flooding area information.	https://environment.data.gov.uk/flood-monitoring/doc/reference
US	NWS/NOAA. Weather alerts (fire, storms, snow, flooding etc).	https://alerts.weather.gov
US	NOAA. Environmental threat grids.	http://www.nssl.noaa.gov/projects/facets
France	SCHAPI. Flooding potential risk areas.	https://www.vigicrues.gouv.fr
Spain	AEMET. Weather alerts.	http://www.aemet.es/es/portada
Worldwide	Google. Environmental disaster risk alerts.	https://developers.google.com/public-alerts

Table A1. Examples of currently available environmental data sources and their APIs

A1.2 Supplementary materials for pipeline steps

A1.2.1 Python scripts as they apply to pipeline steps

Label & pipeline steps	Description	Web URL
(i) Compiles list of at-risk locations, then queries Streaming API (Steps 1-5)	Python shell script. Poll environmental data sources; convert locations to shortlists; query Twitter with shortlists.	https://github.com/battez/twystream/blob/master/twy_tweet_stream.py uses Twython for Twitter API, Streaming API documentation for query: https://developer.twitter.com/en/docs/tweets/filter-realtime/api-reference/post-statuses-filter.html
(ii) Preprocess tweets (Step 6)	Preprocess text of tweets for input to Doc2Vec in Gensim script.	See prepare.py in: https://github.com/battez/analysis/releases/tag/v0.5-alpha
(iii) Manual class labelling script: (Step 7)	Script to quickly present annotator with tweets to assign class label of *relevance* or not. For labelling training and test data for classifier.	See label_tweets.py in: https://github.com/battez/analysis/releases/tag/v0.5-alpha
(iv) Doc2Vec classifier script (Steps 6 and 7)	Main script for generating Doc2Vec model from unlabelled development dataset of tweets. Also then uses labelled train/test tweet dataset to generate Logistic Regression linear classifier, to predict	See classify_tweets_snapshot_working.py in: https://github.com/battez/analysis/releases/tag/v0.5-alpha

	relevant tweets.	
(vi) Log of areas queried	Log of timestamped records of at-risk area bounding boxes queried on the Twitter Streaming API, by script (i), above.	https://github.com/battez/twystream/blob/master/log.txt
(vii) Map tweets to Web page	Collate tweets from MongoDB and plot with Folium.	https://github.com/battez/tweepy_now/blob/master/map-tweets.py
(viii) Python dependencies configuration	Configuration file in YAML format for all dependencies required to replicate Python set-up, environment.yml	https://drive.google.com/open?id=0B057bbdoYJDLYjlIOUFLMkJ0TDg

Table A2. Various scripts and logs used by the pipeline steps.

A1.2.2 Step 3: Converting agency flood area polygons to bounding boxes for Twitter Streaming API location query

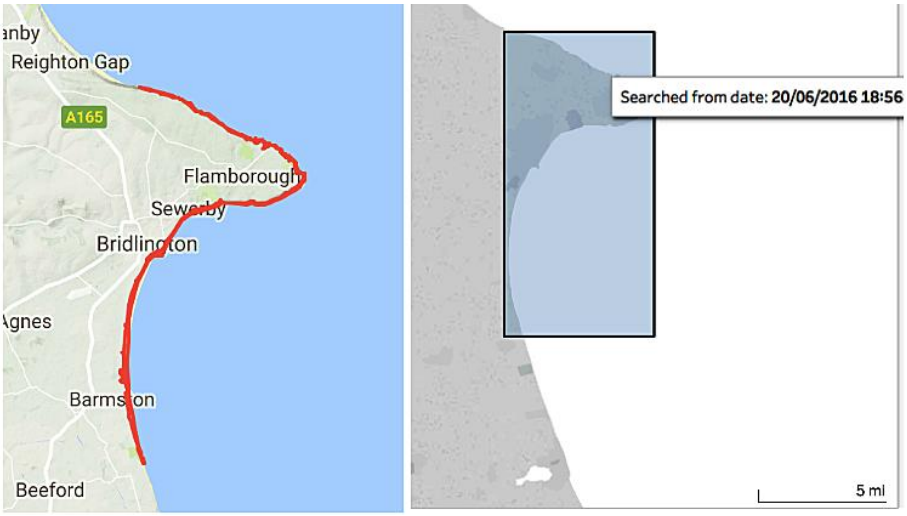


Fig. A1 Map of EA flood area and its precise polygon for Yorkshire coast, highlighted in red (left), and the corresponding bounding box (shaded), which contained the full polygon, which was then monitored by Twitter Streaming API (right), as depicted in Tableau software.

A1.2.3 Steps 3, 4 & 5: Twitter query matching geotag intersects with flood area polygon

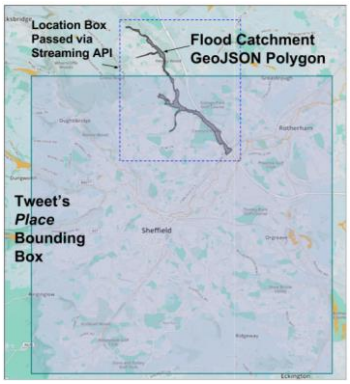


Fig. A2 Map with the original GeoJSON polygon for a flood area (here the catchment is shaded grey), its bounding box after conversion (blue dotted outline) by the pipeline (Step 3), and a tweet with geotag of Place "Sheffield" (shaded pale blue and solid blue outline) returned as result while monitoring the bounding box using a Streaming API location query (Steps 4, 5).

A1.2.4 Tweet Place metadata variation

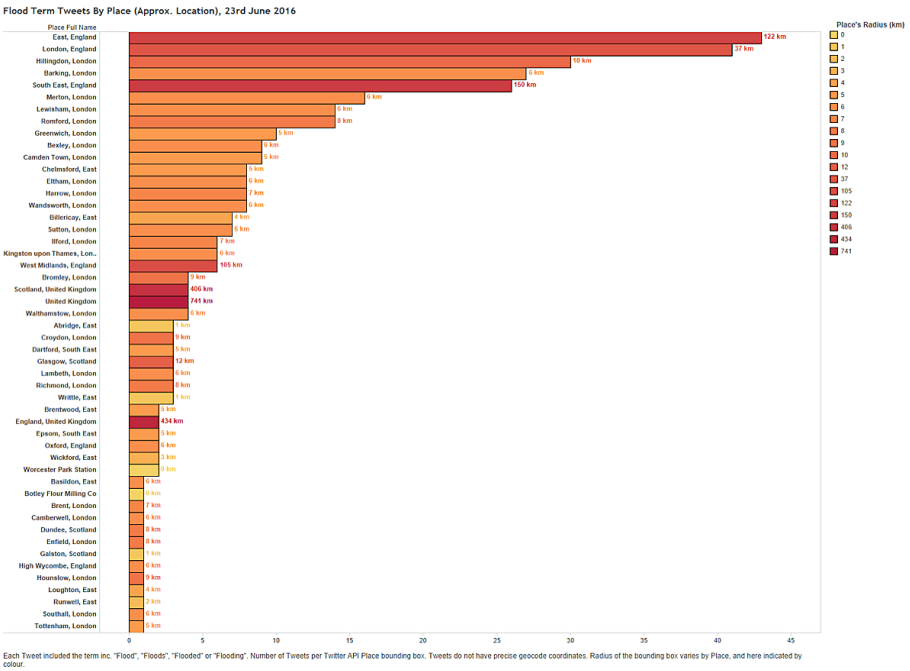


Fig. A3 Number of tweets by Place from the development dataset for 23rd June, including the keyword stem "flood-" and having a geotagged Place, which designated an originating bounding box for that tweet (i.e. no precise GPS point coordinates were attached, just a Place). Colour shows the Place size as calculated using haversine distance for length of box's long diagonal.

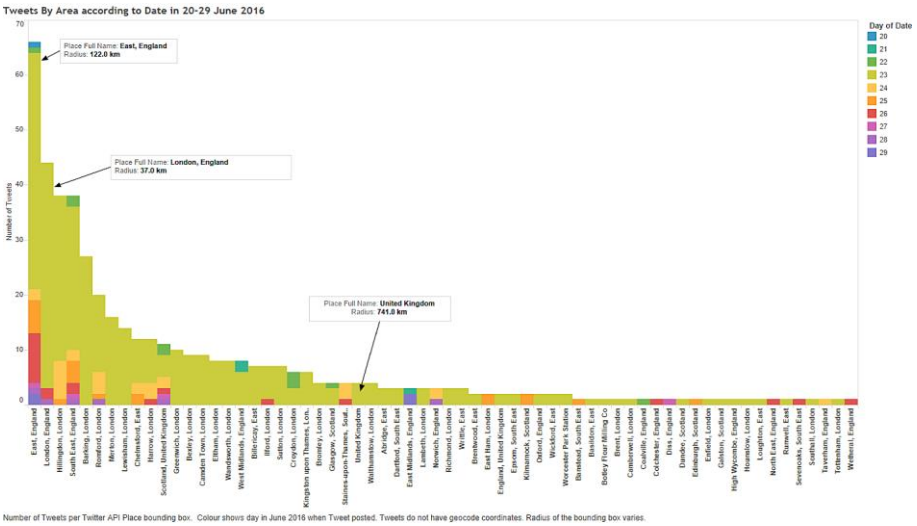


Fig. A4 Number of tweets by Place from the development dataset, June 20-29th, including the keyword stem "flood-" and having a geotagged Place, which designated an originating bounding box for that tweet (i.e. no precise GPS point coordinates were attached, just a Place). Calculated using haversine distance for length of box's long diagonal. Colour shows the date (and thus proportion) of the tweet's posting.

A1.2.5 Development dataset, unlabelled and labelled tweets.

- Unlabelled June dataset 420,218 geotagged tweets and identifiers, in CSV format: https://drive.google.com/open?id=0B057bbdoYJDLT0REZFFWNDQyMXc
  Readers can recover via API search by ID, or individually: e.g. https://twitter.com/statuses/745907644641193984
- Labelled subset of June dataset 4502 tweets, CSV (Note: t_class=1 *Relevant* class): https://drive.google.com/open?id=0B057bbdoYJDLVjg1YTd5d0JsdkU

A1.3 Doc2Vec method details and similarity measure by cosine distance

A1.3.1 Doc2Vec method details

The development dataset of 420,218 documents and ID tags of each were read into a Pandas dataframe[40] before being read as input to train a Doc2Vec model using Gensim. We used all the tweets to train the Doc2Vec models, because the training is entirely unsupervised, there is no need to hold out a test set. The model does not need any supervised information (i.e. labels), it just needs the

[40] Python data analysis library http://pandas.pydata.org/

raw text of the tweets. We used as many tweets as possible as it was believed this would build a more encompassing and robust vocabulary (see Lau and Baldwin, 2016).

After building a vocabulary table the model was trained

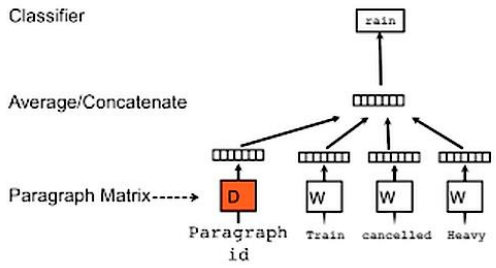


Fig. A5 Paragraph Vector/Doc2Vec Distributed Memory algorithm training schematic: concatenation of separate Document vector D (treated as a pseudo-word) and Word vectors W predicting the next word in a context.

The two algorithms of Paragraph Vector approach, in their Gensim Doc2Vec implementation, are respectively: parameter values dm=1 and dm=0: the former generates any word vectors concurrently with the document vectors, while the latter, dispenses with word vectors entirely.

A1.3.2 Cosine distance similarity between vectors

We can measure cosine similarity of the vectors (A, B in Fig. A6), and identify documents' and words' closeness to each other. This measure ignores vector magnitude (unlike Euclidean distance), using the cosine angle separating them, and calculated by taking the dot product of the vectors' *n* features. The range of cosine distance is [-1, 1], with 1 being identical level of similarity (closeness of context) and -1 being distantly related.

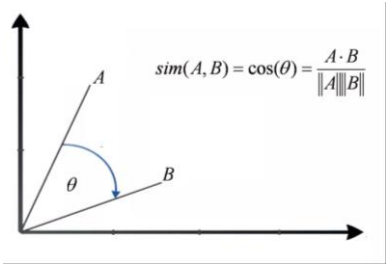


Fig. A6 Measure of cosine similarity for vectors A and B in a vector space model. Angle between vectors is θ.

Appendix B. Twitter location metadata and the Streaming API

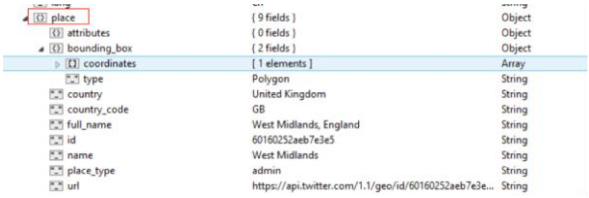


Fig. B1 Tweet JSON Place information example, see also https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object Last accessed: 26th Oct. 2017.

Attribute	Description	Attribute	Description
place	Twitter offers a set of locations to a user who can choose to geotag the tweet by selecting one. The choices are a mix of Twitter's own and Foursquare API since 2015.	coordinates	Latitude and longitude point pair, if enabled by user. Often *null.* If not, can still also have a *place* field too.
place. url	URL to JSON of the polygon for the place.	place. full_name	Human readable location
place. bounding_box	A rectangle of coordinates.	place. place_type	Category e.g. 'neighborhood', 'city', 'poi'

Table B1. Tweet location information: see https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/geo-objects and https://twittercommunity.com/t/foursquare-location-data-in-the-api/36065

References

Alex, B., Llewellyn, C., Grover, C., Oberlander, J. and Tobin, R., 2016. Homing in on Twitter Users: Evaluating an Enhanced Geoparser for User Profile Locations. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016). European Language Resources Association (ELRA), pp. 3936-3944.

de Albuquerque, J.P., Herfort, B., Brenning, A. and Zipf, A., 2015. A geographic approach for combining social media and authoritative data towards identifying useful information for disaster management. International Journal of Geographical Information Science, 29(4), pp. 667-689.

Arthur, R., Boulton, C.A., Shotton, H., and Williams, H.T.P., 2018. Social sensing of floods in the UK. PLOS ONE, 13(1), e0189327.

de Assis, L.F.F.G., de Albuquerque, J.P., Herfort, B., Steiger, E. and Horita, F.E.A., 2016. Geographical prioritization of social network messages in near real-time using sensor data streams: an application to floods. Revista Brasileira de Cartografia, 68(6).

Caragea, C., Silvescu, A. and Tapia, A.H., 2016. Identifying informative messages in disaster events using convolutional neural networks. In Proceedings of the ISCRAM 2016 Conference–Rio de Janeiro, Brazil.

Castanhari, R.E.S., Rocha, R.D.S., Andrade, S.C.D. and de Albuquerque, J.P.D., 2016. A software architecture to integrate sensor data and volunteered geographic information for flood risk management. In Proceedings of the ISCRAM 2016 Conference–Rio de Janeiro, Brazil.

Cerutti, V., Fuchs, G., Andrienko, G., Andrienko, N. and Ostermann, F., 2016. Identification of disaster-affected areas using exploratory visual analysis of georeferenced Tweets: application to a flood event. In Association of Geographic Information laboratories in Europe (AGILE) 2016 pp. 1-5.

Chen, J., Zhang, C. and Niu, Z., 2016, October. Identifying Helpful Online Reviews with Word Embedding Features. In International Conference on Knowledge Science, Engineering and Management, pp. 123-133. Springer International Publishing.

Dobbin, K.K. and Simon, R.M., 2011. Optimally splitting cases for training and testing high dimensional classifiers. *BMC medical genomics*, *4*(1), p.31.

Eilander, D., Trambauer, P., Wagemaker, J. and van Loenen, A., 2016. Harvesting social media for generation of near real-time flood maps. Procedia Engineering, 154, pp. 176-183.

Ekta, P.K., Bundela, P. and Dewan, R., 2017. Tweet Analysis for Real-Time Event Detection and Earthquake Reporting System Development. International Research Journal of Engineering and Technology, 04(05), pp. 3113-3119.

Fawcett, T., 2006. An introduction to ROC analysis. Pattern Recognition Letters, 27(8), pp. 861-874.

Feng, Y. and Sester, M., 2018. Extraction of pluvial flood relevant volunteered geographic information (VGI) by deep learning from user generated texts and photos. *ISPRS International Journal of Geo-Information*, *7*(2), p.39.

Figueroa, R.L., Zeng-Treitler, Q., Kandula, S. and Ngo, L.H., 2012. Predicting sample size required for classification performance. BMC Medical Informatics and Decision Making, 12(1), p. 8.

Herfort, B., Schelhorn, S.J., Albuquerque, J.P.D. and Zipf, A., 2014, May. Does the spatiotemporal distribution of tweets match the spatiotemporal distribution of flood phenomena? A study about the River Elbe Flood in June 2013. In Proceedings of the ISCRAM 2014 Conference–Pennsylvania, USA.

Imran, M., Castillo, C., Diaz, F. and Vieweg, S., 2015. Processing social media messages in mass emergency: A survey. ACM Computing Surveys (CSUR), 47(4), p. 67.

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

Jongman, B., Wagemaker, J., Romero, B.R. and de Perez, E.C., 2015. Early flood detection for rapid humanitarian response: harnessing near real-time satellite and Twitter signals. ISPRS International Journal of Geo-Information, 4(4), pp. 2246-2266.

Juhász, L., Podolcsák, Á. and Doleschall, J., 2016. Open Source Web GIS Solutions in Disaster Management–with Special Emphasis on Inland Excess Water Modeling. Journal of Environmental Geography, 9(1-2), pp. 15-21.

Kryvasheyeu, Y., Chen, H., Obradovich, N., Moro, E., Van Hentenryck, P., Fowler, J. and Cebrian, M., 2016. Rapid assessment of disaster damage using social media activity. Science Advances, 2(3), p.e1500779.

Lau, J.H. and Baldwin, T., 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. In 1st Workshop on Representation Learning for NLP, pp. 78-86. *arXiv preprint arXiv:1607.05368*.

Laylavi, F., Rajabifard, A. and Kalantari, M., 2016. A multi-element approach to location inference of twitter: A case for emergency response. ISPRS International Journal of Geo-Information, 5(5), p. 56.

Le, Q.V. and Mikolov, T., 2014, June. Distributed Representations of Sentences and Documents. In International Conference on Machine Learning (ICML), 32, pp. 1188-1196.

Li, H., Caragea, D. and Caragea, C., 2017. Towards Practical Usage of a Domain Adaptation Algorithm in the Early Hours of a Disaster. In Proceedings of the 14th International Conference on Information Systems for Crisis Response And Management, pp. 692-704.

Meier, P., 2015. Digital humanitarians: how big data is changing the face of humanitarian response. Crc Press.

de Meutter, P., Gerard, L., Smet, G., Hamid, K., Hamdi, R., Degrauwe, D. and Termonia, P., 2015. Predicting small-scale, short-lived downbursts: case study with the NWP limited-area ALARO model for the Pukkelpop thunderstorm. Monthly Weather Review, 143(3), pp.742-756.

Middleton, S.E., Middleton, L. and Modafferi, S., 2014. Real-time crisis mapping of natural disasters using social media. IEEE Intelligent Systems, 29(2), pp. 9-17.

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

Morstatter, F., Pfeffer, J., Liu, H. and Carley, K.M., 2013, June. Is the Sample Good Enough? Comparing Data from Twitter's Streaming API with Twitter's Firehose. In ICWSM, June 2013.

Mukkamala, A. and Beck, R., 2016. Enhancing Disaster Management through Social Media Analytics to Develop Situation Awareness What can be Learned from Twitter Messages about Hurricane Sandy?. In Pacific Asia Conference on Information Systems (PACIS), June 2016, p. 165.

Musaev, A. and Hou, Q., 2016, November. Gathering high quality information on landslides from Twitter by relevance ranking of users and tweets. In Collaboration and Internet Computing (CIC), 2016 IEEE 2nd International Conference, pp. 276-284.

Nazer, T.H., Xue, G., Ji, Y. and Liu, H., 2017. Intelligent Disaster Response via Social Media Analysis A Survey. The Association for Computing Machinery's Special Interest Group on Knowledge Discovery and Data Mining (ACM SigKDD) Explorations Newsletter, 19(1), pp. 46-59.

Ngai, A., 2017. End-2-End Monitoring and Troubleshooting a Real-Time Data Pipeline. USENIX Association, SREcon17 March 2017, San Francisco, CA.

Nguyen, D.T., Mannai, K.A.A., Joty, S., Sajjad, H., Imran, M. and Mitra, P., 2016. Rapid Classification of Crisis-Related Data on Social Networks using Convolutional Neural Networks. Proceedings of the Eleventh International AAAI Conference on Web and Social Media (ICWSM 2017). In arXiv preprint arXiv:1608.03902.

Onorati, T. and Díaz, P., 2016. Giving meaning to tweets in emergency situations: a semantic approach for filtering and visualizing social data. SpringerPlus, 5(1), p. 1782.

Palen, L. and Anderson, K.M., 2016. Crisis informatics—New data for extraordinary times. Science, 353(6296), pp. 224-225.

Rao, R., Plotnick, L. and Hiltz, R., 2017. Supporting the use of social media by emergency managers: Software tools to overcome information overload. In Proceedings of the 50th Hawaii International Conference on System Sciences, January 2017.

Rosser, J.F., Leibovici, D.G. and Jackson, M.J., 2017. Rapid flood inundation mapping using social media, remote sensing and topographic data. Natural Hazards, 87(1), pp. 103-120.

Sadilek, A., Kautz, H.A., DiPrete, L., Labus, B., Portman, E., Teitel, J. and Silenzio, V., 2016, February. Deploying nEmesis: Preventing Foodborne Illness by Data Mining Social Media. In Proceedings of the 28th Annual Conference on Innovative Applications of Artificial Intelligence (IAAI 2016), pp. 3982-3990.

Sen, A., Rudra, K. and Ghosh, S., 2015, January. Extracting situational awareness from microblogs during disaster events. In Communication Systems and Networks (COMSNETS), 2015. 7th International Conference on pp. 1-6. IEEE.

Smith, L., Liang, Q., James, P. and Lin, W., 2015. Assessing the utility of social media as a data source for flood risk management using a real-time modelling framework. Journal of Flood Risk Management, 10: 370–380 doi: 10.1111/jfr3.12154.

Spielhofer, T., Greenlaw, R., Markham, D. and Hahne, A., 2016, December. Data mining Twitter during the UK floods: Investigating the potential use of social media in emergency management. In Information and Communication Technologies for Disaster Management (ICT-DM), 2016 3rd International Conference on (pp. 1-6). IEEE.

Sridhar, V.K.R., 2015. Unsupervised topic modeling for short texts using distributed representations of words. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing* (pp. 192-200).

Starkey, E., Parkin, G., Birkinshaw, S., Large, A., Quinn, P. and Gibson, C., 2017. Demonstrating the value of community-based (‘citizen science’) observations for catchment modelling and characterisation. Journal of Hydrology, 548, pp.801-817.

Terpstra, T., De Vries, A., Stronkman, R. and Paradies, G.L., 2012. Towards a realtime Twitter analysis during crises for operational crisis management (pp. 1-9). Burnaby: Simon Fraser University.

Tsou, M.H., Jung, C.T., Allen, C., Yang, J.A., Han, S.Y., Spitzberg, B.H. and Dozier, J., 2017, July. Building a Real-Time Geo-Targeted Event Observation (Geo) Viewer for Disaster Management and Situation Awareness. In International Cartographic Conference (pp. 85-98). Springer, Cham.

Vieweg, S., Castillo, C. and Imran, M., 2014, November. Integrating social media communications into the rapid assessment of sudden onset disasters. In International Conference on Social Informatics (pp. 444-461). Springer International Publishing.

Vitolo, C., Elkhatib, Y., Reusser, D., Macleod, C. J. A. and Buytaert, W., 2015. Web technologies for environmental Big Data. Environmental Modelling & Software, 63, pp.185-198.

Vosoughi, S., Vijayaraghavan, P. and Roy, D., 2016, July. Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder. In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, 2016, pp. 1041-1044.

Wu, D., Zhu, L., Xu, X., Sakr, S., Sun, D. and Lu, Q., 2016. Building pipelines for heterogeneous execution environments for big data processing. IEEE Software, 33(2), pp.60-67.

Xing, C., Wang, D., Zhang, X. and Liu, C., 2014, December. Document classification with distributions of word vectors. In Asia-Pacific Signal and Information Processing Association, 2014 Annual Summit and Conference (APSIPA) (pp. 1-5). IEEE.

Zheng, F., Westra, S., and Leonard, M. (2015). "Opposing local precipitation extremes." Nature Clim. Change, 5(5), 389-390.