Magumba *et al. J Big Data (2018) 5:31*
https://doi.org/10.1186/s40537-018-0139-2

Journal of Big Data

RESEARCH Open Access

Ontology boosted deep learning for disease name extraction from Twitter messages

Mark Abraham Magumba[1*], Peter Nabende[1] and Ernest Mwebaze[2]

*Correspondence: magumbamark@hotmail.com
[1] Department of Information Systems, Makerere University College of Computing and Information Sciences, Kampala, Uganda
Full list of author information is available at the end of the article

Abstract

This paper presents an ontology based deep learning approach for extracting disease names from Twitter messages. The approach relies on simple features obtained via conceptual representations of messages to obtain results that out-perform those from word level models. The significance of this development is that it can potentially reduce the cost of generating named entity recognition models by reducing the cost of annotating training data since ontology creation is a one-time cost as the conceptual level the ontology is meant to be fairly static and reusable. This is of great importance when it comes to social media text like Twitter messages where you have a large, unbounded lexicon with spatial and temporal variations and other inherent biases that make it logistically untenable to annotate a representative amount of text for general purpose models for live applications.

Keywords: Epidemiology, Twitter, Sentiment analysis, Text classification, Concept ontology, Data mining, Knowledge engineering

Introduction

In previous years there has been massive interest in the use of unstructured web data such as tweets [1–6], web searches [7] and images for disease surveillance [8]. This interest is spurred by the unprecedented growth of user generated content through social platforms like Twitter and facebook. In the studies cited it has been shown that disease reports made by users in their public social posts correlate with actual disease activity on the ground and can therefore be employed as a basis for so-called internet based syndromic surveillance. The integration of these systems into the public health surveillance process confers several advantages. First of all these systems have global reach for instance facebook, the largest service has 1.47 billion daily active users who generate unsolicited information at a rate of 293,000 status updates, 510,000 comments and 136,000 photo updates a minute [9] and Twitter which is the subject of this paper has about 330 million daily active users who generate more than 500 million tweets a day [10].

A system capable of sifting through this deluge of real time user information for mentions of disease occurrence has a very high chance of detecting outbreaks far quicker than traditional surveillance approaches moreover at a fraction of the cost due to the

fact that process elements like data collection can essentially be kept external to the surveillance process. Most modern social systems allow some form of access to their data through APIs (Application programming interfaces)[1] and barring the costs of API implementation the data is completely free as it is provided by unpaid users motivated by the need to communicate with each other in the form of status updates. In addition social APIs usually provide additional utilities like keyword and location based filtering making a lot easier to obtain fairly dense location specific samples for any topic of interest.

However, the vast majority of systems have been limited in scope to particular diseases such as the flu and dengue fever [11]. The general approach entails the use of large lists of keywords in conjunction with keyword lookup approaches. To monitor multiple diseases the same technique has simply been extended by incorporating multiple definitions into hierarchical taxonomies or ontologies [12, 13]. For both cases systems rely on a priori definitions of disease events that are supplied by system implementers. True general threat detection systems like Promed-Mail and the Global Public Health Intelligence Network (GPHIN) [14, 15] have had to rely on human mediation.

However ongoing advances in machine learning and natural language processing are making it increasingly feasible to deploy fully automated disease surveillance and minimize the need for human mediation. The key merits of an automated approach include the fact that it would have a significantly lower operational cost than a human mediated approach coupled with a very high throughput. Given the interconnected nature of the modern world and the speed at which epidemics may disperse it is important to process as much data as possible and as quickly as possible. However, the need to dictate the epidemiological focus of such systems by predetermining diseases of interest means that it is not possible to accommodate new health concerns such as emerging diseases[2] without significant rework. In this paper we demonstrate an ontology boosted neural architecture for generalized automated disease name extraction from Twitter messages.

Related work

Previous attempts to detect disease names in natural language text have been made on less noisy technically oriented sources such as discharge summaries, echocardiogram, radiology, and ECG reports [16] and specialized databases such as DistiLD [17] and COSMIC (Catalog of Somatic Mutations in Cancer) as described in [18]. In natural language parlance this task is referred to as named entity recognition. For disease named entity recognition techniques such as machine learning methods like Conditional Random Fields (CRFs) [16] and dictionary lookup techniques [19] have been employed. In these cases because the text originates from predominantly technical writers semantic annotation is fairly straight forward using any of the several publicly available medical thesauruses like UML (Unified Medical language)—SNOMED-CT.[3]

Work on tweets is limited, previous work has employed CRFs and log linear models. For most work involving named entity recognition several features are required to

[1] Programming libraries that allow external code to make calls to a system.

[2] Infectious diseases that have recently appeared within a population or those whose incidence or geographic range is rapidly increasing or threatens to increase in the near future.

[3] https://www.snomed.org/snomed-ct.

obtain good performance. These features include the words themselves, word prefixes, word suffixes, part of speech tags of words, semantic category of word in reference ontology, word surface form etcetera. These approaches, with the exception of work by Magumba et al. [20], implicitly assume that the lexical composition of training data and target data is similar and have generally employed very small data sets. However, Twitter text is known for its large, unbounded lexicon in addition to other distortions like misspellings and made up words but even in these experiments which assume a fixed lexicon, the performance of automated named entity recognition approaches for disease name extraction significantly lags the state of the art on Twitter for instance Jimeno-Yepes et al. [21] report an F_1 score of only 0.63 on a dataset of only 1300 tweets in which 1200 tweets are used for training and 100 for testing. In addition it is difficult to reliably obtain some of the most informative features such as part of speech tags and chunk tags on Twitter messages due to the afore-mentioned problems.

Materials and methods

Our approach builds on previous work by Magumba and Nabende [22], in which an ontology is used to successfully obtain state of the art performance for classification of messages about diseases using a single model for lexically divergent data. The approach generates conceptual representations which are used to create classification models that generalize better than word based models over the held out test datasets. The training data comprises messages about the flu whereas the test data comprises messages about an arbitrary selection of other diseases. In this paper we build on the observation that neural word embeddings can be learned over these conceptual representations and that similarly to word level models they significantly improve message classification performance. In particular, the Doc2Vec [23] algorithm is applied which extends the word-2vec algorithm [24] by applying similar reasoning to entire documents. In this work we investigate if similar gains can be obtained for the named entity recognition task given the combination of deep learning and neural embeddings has produced state of the art results in other domains.

General rationale

To arrive at the ontology we employ two key intuitions; firstly that for the purpose of communicating disease incidence some words are more important than others. Secondly, that words themselves are representative of some higher mental concepts and that in the bounds of a given topic of discourse there is a fairly small number of concepts but a possibly infinite number of words and ways to express them. So, what we have attempted is to identify these higher concepts that communicate illness and train classifiers on them rather than words to achieve low performance variance across lexically divergent data sets. Figure 1 is an informal, visual proof of these intuitions. The figure comprises word clouds generated from four datasets containing messages about conjunctivitis, the flu and common cold, gastroenteritis (stomach flu) and norovirus. To arrive at these visualizations we exclude stop words and the names of the diseases themselves. In essence this is a visualization of the word level context of the disease names.

As is evident the word “eye” is prominent for conjunctivitis in Fig. 1a as is the word “stomach” for gastroenteritis in Fig. 1c and with some domain knowledge we can easily

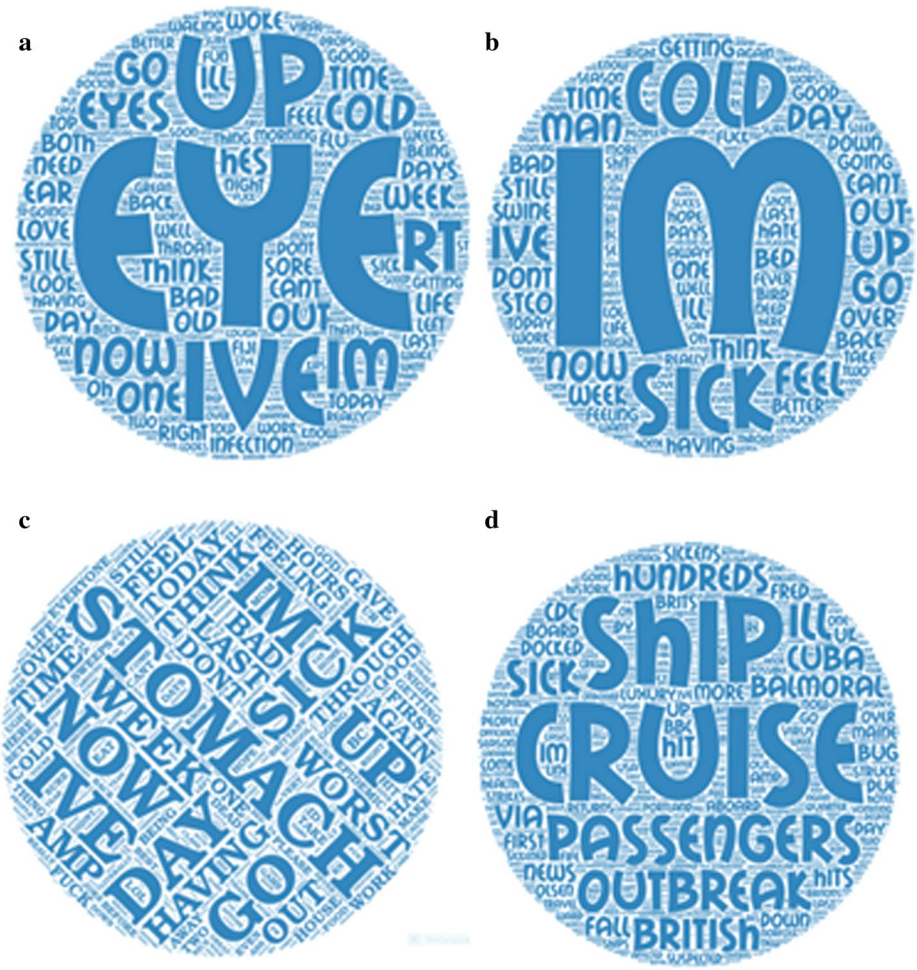

Fig. 1 Empirical word frequencies for context words for different diseases names: **a** conjunctivitis, **b** flu + common cold, **c** gastroenteritis, **d** norovirus

infer that these symbolize the sites of infection for the respective diseases. Furthermore, these words would appear as distinct elements of information to a word level model but may actually serve an equivalent purpose for a generalized concept-level model. For instance we can create a category for anatomical references and any mentions of "eye" and "stomach" would account as mentions of this category rather than as unique data points allowing models to employ knowledge about the conceptual context of one disease to messages about another disease. Furthermore, even though we do not encounter the word "leg", as humans we can easily predict that it too is a member of this category and hence extend it.

In addition, even though the term "interstitium" refers to a part of the body it is way too technical and specific to expect it to occur in casual Twitter posts with significant regularity and it is our opinion that we can safely ignore it and simply register it as a noun. We can therefore identify such concept categories and almost exhaustively describe them by creating lists of words that describe that category. In deriving concepts for the ontology we simply repeat this process several times until we obtain a set

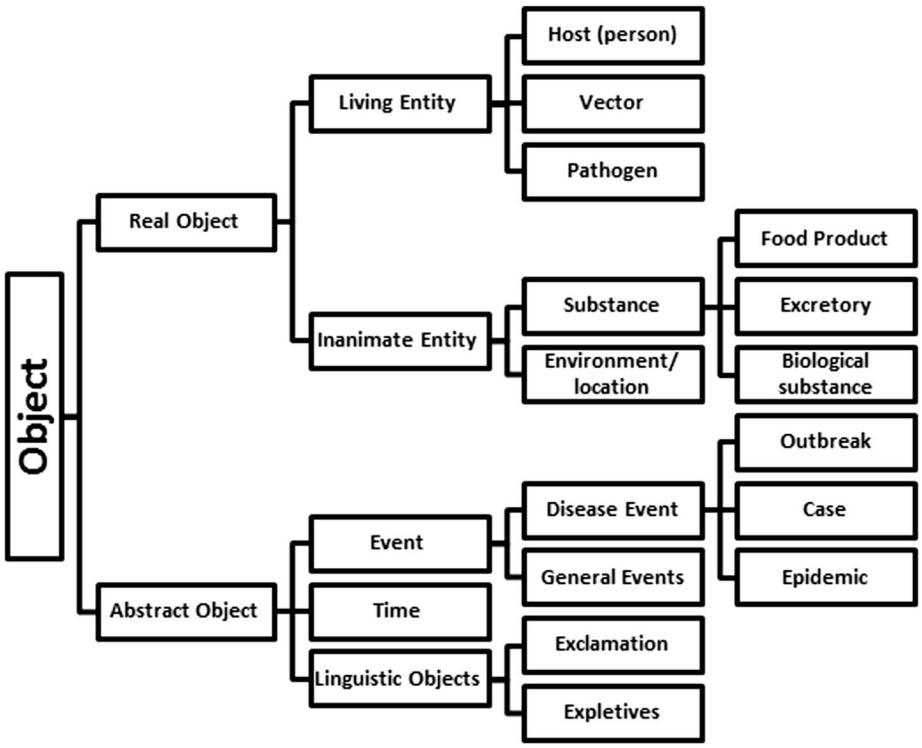

Fig. 2 Partial object hierarchy for disease communication ontology

of concepts that we think are representative of the domain. In general we avoid technical terminology and names of specific diseases and drugs and focus on the basic linguistics of communicating illness by leveraging domain knowledge to identify those words that are likely to be in the same contexts as names of diseases.

An ontology for disease incidence detection on Twitter

The ontology models the minimum set of semantic concepts required to communicate disease incidence. These concepts are of two broad categories, those that directly describe disease incidence such as references to disease causing organisms such as bacteria and general linguistic terminology like words that imply negation and references to temporality and descriptions of space and time. As depicted in Fig. 2, at the top of the hierarchy everything is conceptualized as an object, there are two types of objects that is, real objects and abstract objects. Real objects refer to tangible things and abstract objects refer to concepts like time, negation and events.

Living things comprise the key biological actors such as hosts (the organism that suffers disease which in this case is a person), pathogens (disease causing organisms) and vectors (disease spreading organisms). Inanimate objects are of two major classes namely environments and substances. There are three types of concepts namely relationships and properties and actions. Relationships describe interactions between concept classes in the object hierarchy and correspond to OWL (Web Ontology Language) object properties whereas properties describe object and relationship attributes and correspond to OWL data properties. Actions describe object behavior for certain objects

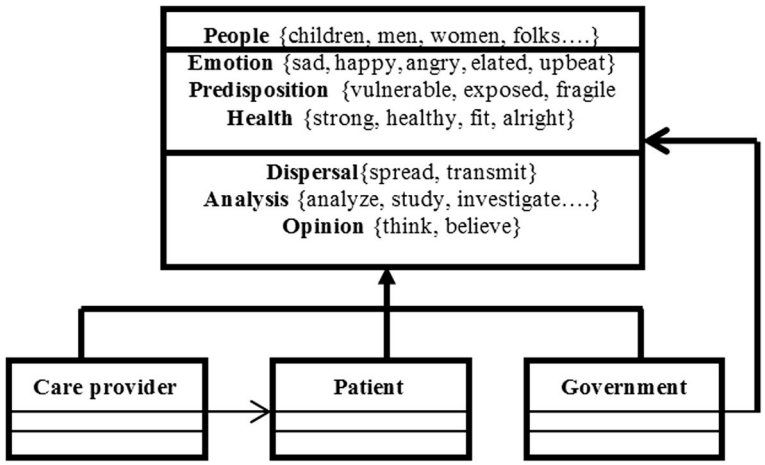

Fig. 3 People object sub hierarchy with corresponding concept dictionaries indicated in curly braces

like people where a typical action is to exercise and can be thought of as data properties for which the range and domain are the same.

Concept dictionaries

Finally, for each conceptual object be it a relationship, property or action there is a corresponding concept dictionary. Figure 3 depicts a partial decomposition for the people conceptual object with corresponding concept dictionaries indicated in curly brackets. In the figure we use a UML class diagram notation as opposed to the more relevant RDF schema diagram for compactness. The figure depicts the "People" concepts and the "government", "care provider" and "patient" as sub classes of the "People" object. The reason these are depicted as such is because we think in terms of the conceptual context of disease mentions and these concepts generally have the same context at the conceptual level and the same or similar relationships and properties. For instance governments and government agencies like the CDC or the Department of Health can be viewed as special collections of people and in fact within the context of disease occurrence will perform typically human actions like investigating, apologizing and reporting. In addition there are other interactions between concepts for instance the directed line from government to people indicates an unspecified interaction between the government and the individual.

The concept dictionary is simply a list of words related to the concept. Roughly speaking the members of a given concept dictionary are related by three relationships namely synonymy, hyponymy/hypernymy and meronymy. Synonymy refers to semantically equivalent words such as "skin" and "dermis", hypernymy is refers to increasing generalization for instance "location" is a hypernym of "house", hyponymy is the inverse of hypernymy as in "house" would be a hyponym of "location". The full ontology along with a description is publically available in OWL/XML format at this Github repository.[4] Also the process of creating and populating concepts with

[4] https://github.com/MarkMagumba/Twitter-Disease-incidence-Description-Language-Ontology.

concept dictionaries is largely independent of the data. At the conceptual level the ontology is intended to be a static and reusable resource.

Recurrent neural networks

A recurrent neural network is a form of neural network in which sequential information is retained as opposed to a basic feed forward neural network. A neural network is a form of computational graph that approximates some target function by minimizing the error on a provided set of examples. A neural network breaks down the learning problem by defining a set of computational units or neurons organized into layers. Assuming the most basic neural network comprising a single layer with a single neuron, where x is the input, w is the weight, y is the output also referred to as the activation, b is some bias and the target value is τ and ϕ is some activation function and E is the error Eq. 1 describes the basic architecture of a neural network.

$$y = \phi(w * x + b) \tag{1}$$

$$E = -(y - \tau) \tag{2}$$

$$w \leftarrow w + \eta E \tag{3}$$

The learning task attempts to learn the most suitable values for w given some training data by minimizing E via an iterative procedure with two components namely forward and back propagation. During forward propagation, the value of y is computed for each training example and the error obtained via Eq. 2. During back propagation the error is used to adjust the weight, w, as described by expression 3. The adjustment is scaled by η, the so called learning rate. This is important since in practice, neural networks do not learn the exact target function but rather the best approximation given the provided parameters and data. The error term is computed over all examples as a root mean squared error. It is critical that the learning rate is not too big as this may cause the learning routine to overshoot the optimal solution and hence never "converge" on the other hand too small a learning rate results very large training times. For modern machine learning libraries, this is handled by the backend with optimization methods such as adam.

It turns out that neural networks are very powerful learners often obtaining state of the art results in machine learning tasks such as image recognition, natural language processing and speech recognition. It has been shown that a two layer neural network can be used to approximate any function [25]. In the case of multiple layers, the layers between the input and output layers are referred to as hidden layers. As demonstrated in Fig. 4, given a_k^l is the activation of the kth neuron in the lth layer and $w_{i,k}^l$ is the weight between the ith neuron of the lth layer and the kth neuron of the $l+1th$ layer. The following equation holds for a neural network in which the $l-1th$ layer contains n neurons.

$$a_k^l = \phi\left\{\sum_{i=1}^{i=n}(a_i^{l-1} * w_{i,k}^l + b_i)\right\} \tag{4}$$

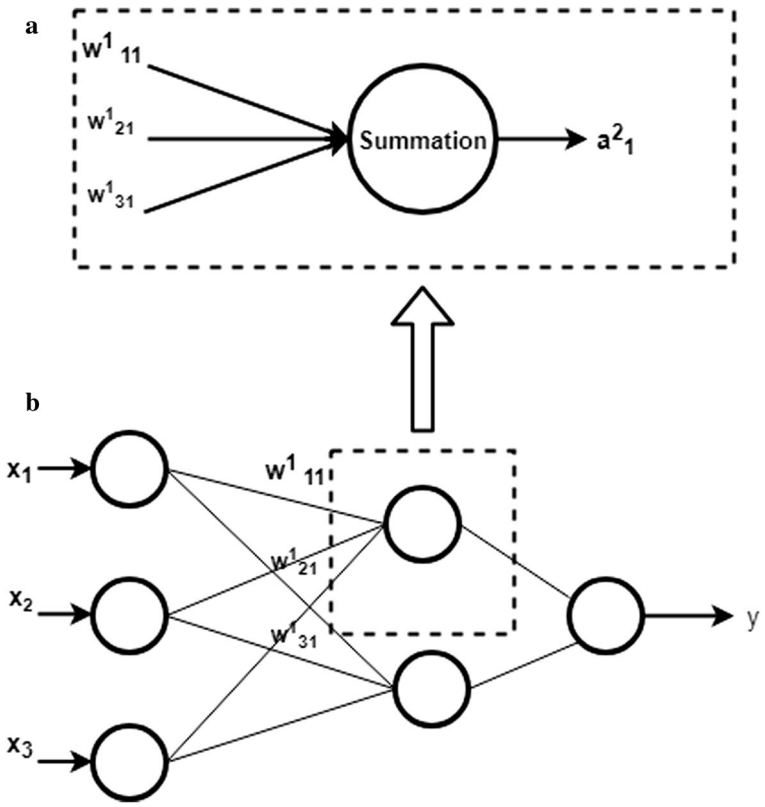

Fig. 4 Feed forward neural network, **a** shows the inputs and output at a single neuron, **b** depicts a multi-layer neural network

Different activation functions are possible each with their own distinct characteristics such as Rectified linear units (ReLUs), tanh, sigmoid units and softmax units. Stacking additional layers allows neural networks to learn finer decision boundaries and create better approximations of the target function. The weight update rule in Eq. 3 can be generalized as follows for multi-layer neural networks:

$$\mathrm{w} \leftarrow \mathrm{w} + \eta * \frac{\partial \mathrm{E}}{\partial \mathrm{w}} \tag{5}$$

In this case the error is computed in the forward pass and then each weight is updated by subtracting the product of the derivative of the error with respect to that weight and a scalar (the learning rate). In addition the error is defined such that it is differentiable and for n training examples takes the form:

$$\mathrm{E} = \frac{1}{n} \sum_{i=1}^{i=n} \left(y - \tau\right)^2 \tag{6}$$

The derivatives for shallower layers are obtained by repetitive application of chain rule. As an example the first layer 1 gradient in Fig. 4a is computed as follows:

$$\frac{\partial E}{\partial w_{11}^1} = \frac{\partial E}{\partial w_{11}^2} * \frac{\partial w_{11}^2}{\partial w_{11}^1} \tag{7}$$

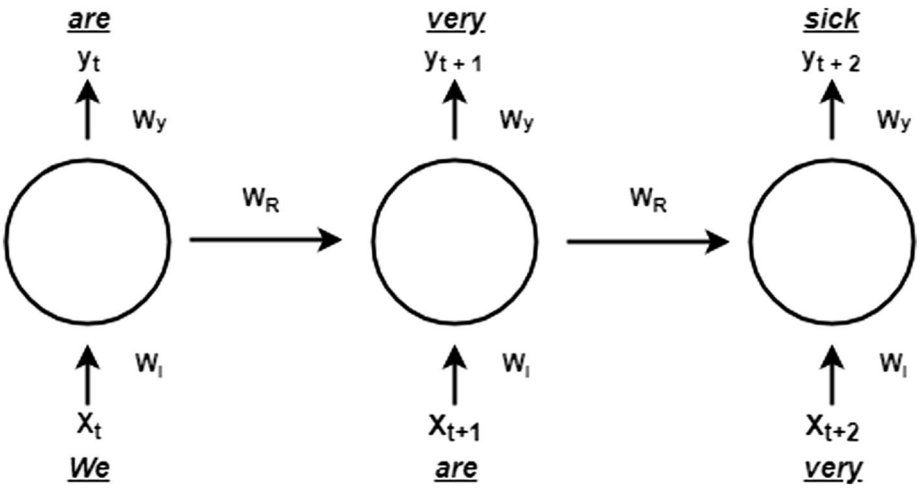

Fig. 5 Recurrent neural network

In the case of the feed forward neural network time is considered static and the inputs independent, however this is suboptimal for certain tasks like sequence tagging. In this case the input is considered to be sequential and the results of previous steps are important in determining the current state. Recurrent neural networks introduce the idea of time steps. For instance for the input sequence, "we are very sick", the input a time t = 0 is the token "We". In this case Fig. 5 summarizes the effective model architecture for a one layer recurrent neural network:

The following equations describe forward propagation for the architecture:

$$h^t = f\left(W_l X^t + W_R h^{t-1} + b_h\right) \tag{8}$$

$$y^t = f\left(W_y h^t + b_y\right) \tag{9}$$

At each time step, two outputs are computed the "history" and the current prediction. The history is given by Eq. 8 and the current prediction by Eq. 9. The weights W_y, W_R, W_l and bias vector b_h are shared across the layer. In this case f is referred to as the recurrence formula; it is common to use tanh activation function for f. The output layer employs softmax activations allowing the output to be interpreted as probabilities. The error is interpreted as a cross entropy error, however the total error from all time steps is considered as the loss. Similarly the gradients with respect to W_y, W_R, W_l for all time steps are summed up into a single gradient since W_y, W_R, W_l are shared across time steps.

Back propagation works similarly to back propagation in feed forward neural networks but is commonly referred to as back propagation through time since gradients are computed with respect to weights at previous time steps in addition to the current time step. It is also common to limit the length of the sequence through which to back propagate for the sake of memory efficiency and in this case it is referred to as truncated back propagation through time.

A key advantage of recurrent neural network is that it can use the full context of the entire sequence. For methods such as CRFs and log linear models the complexity is quadratic in the length of the sequence and length of the label set requiring most implementations to limit the context, however for recurrent neural networks this is not an issue as the entire history at time t is represented in the vector h^t.

Vanishing and exploding gradients

Recurrent neural networks can result in arbitrarily long gradient computations particularly in the computation of errors with respect to W_R and W_l as these are obtained by the gradient chain rule as opposed to the error with respect to W_y which depends only on values at the current time step. In situations where the result of these computations is a consecutive sequence of numbers greater than or less than 1.0 it can result in arbitrarily large or exceedingly small gradients referred to as exploding and vanishing gradients respectively. In either case the effect is to impede further learning as weight updates become too small or too large. Several solutions exist for this problem including making changes to the architecture, gradient clipping (for exploding gradients) in which minimum and maximum bounds are defined for the gradient a priori and employing special learning units referred to as gates, in particular LSTM (Long Short Term Memory) gates and GRUs (Gated Recurrent Units).

Experimental setup

Data sets

We employ the same dataset used in Magumba et al. [20]. These tweets were obtained via basic Twitter account using a python script through Twitter's Streaming API [26]. The tweets are marked as public by the authors; which is the default security level, making them available to third party applications like ours. We employ simple keyword filters to extract the desired tweets. These keywords are names of diseases like flu and pink eye.

Not all messages mentioning diseases necessarily talk about them in the context of reporting active cases of disease incidence and since this is the context we are interested in, we see no point in training our model on tweets that talk about diseases in these alternative contexts. Therefore, we further examine the tweets manually to extract only those that report ongoing or recent cases of disease activity within some time window of up to a few weeks where we assume that diseases may still be in their communicable period. Using this approach we generate three datasets: An influenza + common cold + Listeria data set which comprises 73,848 tokens in 7835 tweets which we employ as the training data set, a mumps + measles + pneumonia data set which comprises 3866 tokens in 316 tweets a pink eye data set which comprises 3428 tokens in 267 tweets.

The mumps + measles + pneumonia and the pink eye data sets are used for testing. The former is used to test the models on cases where the disease name is a singular word and the latter to assess the performance where the disease name is a multi-word expression. In addition we eliminate duplicates by removing retweets, (tweets with the "RT" tag) and also manually checking for duplicates that may not be marked as "RT". Finally we remove all punctuation except the "#" and "@" symbols where they appear at the beginning of tokens where they are used to denote hashgtags and users respectively.

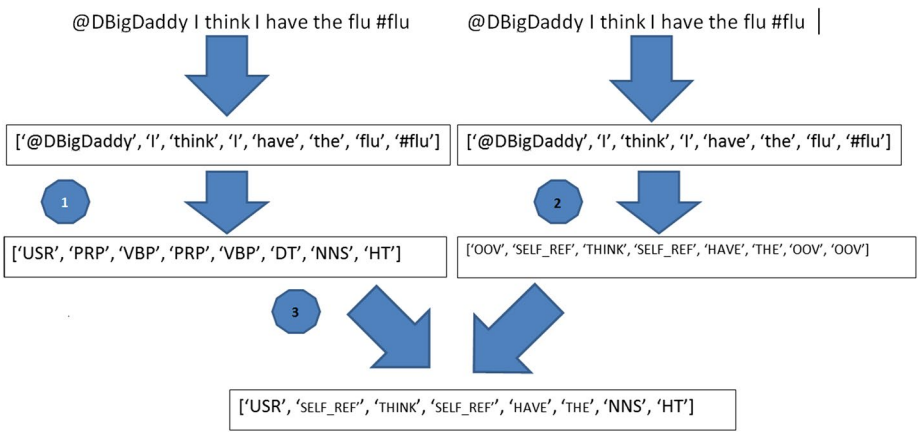

Fig. 6 Derivation of conceptual CNF representation from plain tweet

Annotation

For each word in our data set, we assign a label of ET (Entity True) or PERS for persons or ORG for organizations or PLACE for geo-political entities or MC (Miscellaneous category) for everything else. The Entity True labels refer to words that are part of a disease name.

Feature engineering

We transform each tweet into a vector of features as follows. Firstly, we flatten out our ontology into a list of its constituent concepts. As stated in "General rationale" section each concept is associated with a group of words or tokens referred to as the concept dictionary. Each concept is effectively a list of words and the full ontology is basically a list of lists. In this sense it is a heavily redacted English dictionary containing only words we consider to be of epidemiological relevance. To obtain the feature vector we simply tokenize each tweet and for each token we do a dictionary look up in our flattened ontology.

If the token exists in the ontology, we simply replace it with the concept in which it occurs. As an example the sentence "I have never had the flu" is encoded as "SELF_REF HAVE FREQUENCY HAVE DT NN". SELF_REF refers to "Self references" which is the concept class for terms that persons use to refer to themselves such as "I", "We" and "Us" used as an indicators of speaking in the first person, "HAVE" is the concept class for "have" or "had" which is a special concept class since the verb "to have" is conceptually ambiguous as it can legitimately indicate two senses that is falling sick or possession. The "FREQUENCY" terms refers to a reference to the "FREQUENCY" concept which denotes temporal periodicity. Figure 6 depicts the derivation for the statement "@DBigDaddy I think I have the #flu".

To arrive at the concept representation we merely perform a simple list lookup; for each token we iterate through all concepts to see if it exists in any concept's dictionary. If it does, then that concept's label is returned otherwise the token is replaced with its part of speech tag. The original version of our ontology has 136 concepts corresponding to 1531 tokens versus a vocabulary of about 59,000 tokens for our full corpus (or several billion

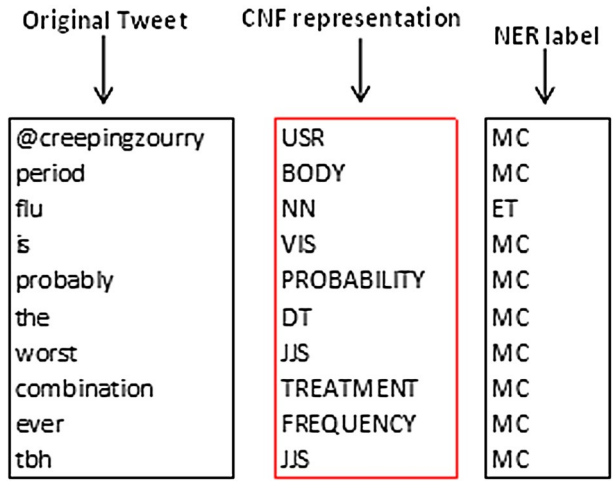

Fig. 7 Sample training data annotation

words in English). Needless to say, most words are out of vocabulary. We have found that replacing these terms with their part of speech information rather than ignoring them produces better results particularly on message classification tasks. To obtain the part of speech tag we employ the GATE twitter tagger [27] which uses the Penn Treebank tag set [28] in addition to three more tags "HT", "USR" and "URL" corresponding to twitter specific phenomenon namely hashtags, users and URLs. We refer to this form as the Concept Normal Form (CNF). Figure 7 is a concrete example of a tweet and its representation in CNF which is used as the input for our named entity recognition models, and the corresponding named entity tags per token that are the training targets.

Embedding derivation

To obtain the embeddings to be used as input for the neural network we use the Doc2vec algorithm over unlabeled examples of our entire corpus in the conceptual representation. From the Doc2Vec model we can extract word2vec embeddings which are able to preserve non trivial semantic relationships between words. For instance where v(King) is the vector for king it has been found that the expression $v(King) - v(Man) + v(Woman)$ returns a vector that is roughly equivalent to v(Queen). We do not train the word2vec algorithm directly but extract out word2vec vectors from the best performing Doc2Vec model in Magumba and Nabende [22]. We do this to demonstrate that a single model may be used for multiple purposes.

The Doc2Vec algorithm may be implemented using two alternative architectures namely distributed memory (PV-DM) or distributed bag of words (PV-DBOW) which are analogous to the two alternative word2vec architectures namely continuous bag of words (CBOW) and skipgram. Unlike the PV-DBOW architecture in which there is no joint training of word vectors and paragraph vectors the PV-DM architecture generates word vectors as a by-product of training. The experiments in Magumba and Nabende [22] employ the PV-DM architecture. It is for this reason that we are able to extract word2vec concept vectors for this experiment. Figure 8 depicts the word2vec skipgram model (on the left) alongside the Doc2Vec PVDM model (on the right).

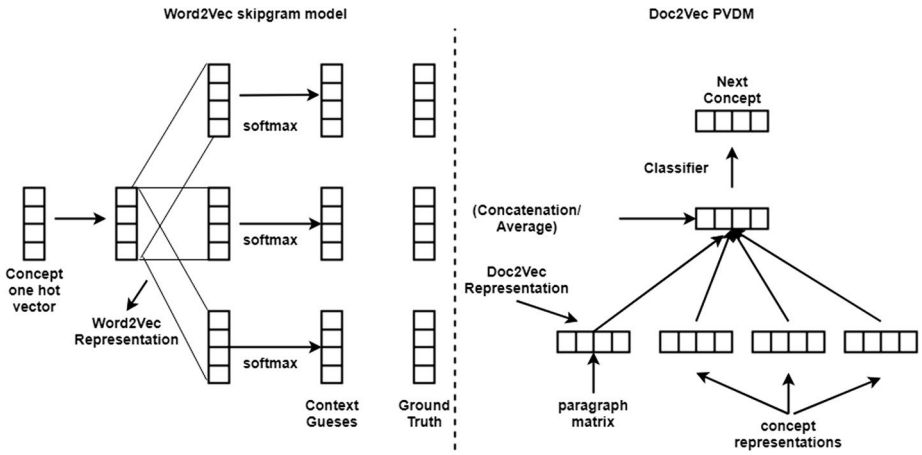

Fig. 8 Word2Vec skipgram architecture versus Doc2Vec PVDM architecture

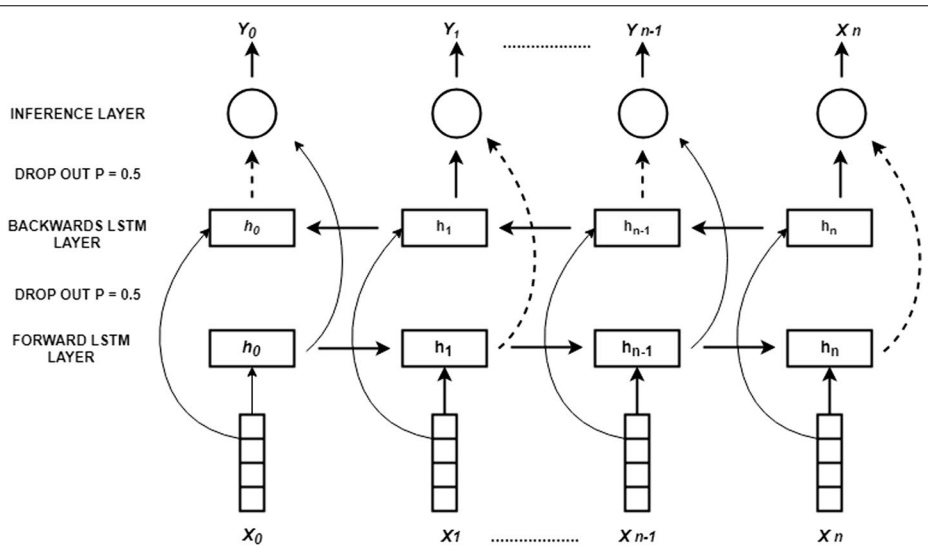

Fig. 9 Model architecture for recurrent neural network

Model training

We employ the following model architecture to train our named entity recognition model. The only input to the model is word2vec representations of tweets in the conceptual representation from "Embedding derivation" section. We employ a bi-directional LSTM RNN. The architecture is depicted in Fig. 9. To achieve a bidirectional neural network two layers are stacked with one layer learning the forward sequence and the other layer learning the backwards sequence. The outputs from both layers are then combined using some "merge mode" which describes how to combine the output from the bi-directional RNN. We use the python Keras[5] package whose default merge model is "concat" which concatenates the output from both

[5] https://keras.io/.

layers. We employ 150 units for each LSTM layer. The layers are interspersed with dropout layers. Dropout layers combat over-fitting by randomly switching off some units with a given probability. We use a dropout probability of 0.5. We train the model on an NVIDIA Quadro K610M GPU. We employ 20 training epochs.

We also perform two additional experiments to demonstrate the utility of our concept level embeddings. The first experiment employs word level embeddings from a custom word2vec model trained from our corpus with a vocabulary of 59,000 words. However, we transform our corpus such that we transform any hashtags into "ht", urls into "url" and user names into "usr" and ignore any tokens occurring less than 2 times thereby truncating our corpus to 9206 tokens. The second employs word level embeddings from Google's word2vec model trained from a 6 billion word corpus with a vocabulary of 3 million unique words. For these additional experiments we essentially use the same architecture except the input shape is slightly different because the optimal model with the custom word2vec model is obtained with 100 dimensions and Google's 6 billion word corpus word2vec model employs 300 dimensions.

We do not perform any parameter tuning and employ what we expect to be typical values for dropout, number of layers and number of epochs. Since the goal is to obtain a model that performs reasonably well on multiple test datasets that are lexically divergent we deliberately try to employ a simple architecture. The full vocabulary for the CNF representation is only 166 concepts.

Results and discussion

The results are summarized in the Table 1. In addition to the results from these experiments we also include the results from Magumba et al. [20] who employed conditional random fields and log linear models on the same data.

The results show the precision, recall, and F measure of the different methods and representations with respect to the "ET" tag. The LSTM with concept level

Table 1 Experimental results

Methods	Measure	Data set	
		Mumps + measles + pneumonia	Pink eye
CNF + log linear model	Precision	0.68	0.71
	Recall	0.43	0.58
	F measure	0.53	0.64
CNF + CRF	Precision	0.79	0.68
	Recall	0.58	0.63
	F measure	0.67	0.65
LSTM + custom Word2Vecmodel	Precision	0.00	0.74
	Recall	0.00	0.05
	F measure	0.00	0.02
LSTM + generic Word2Vec model	Precision	0.55	0.35
	Recall	0.04	0.17
	F measure	0.08	0.23
LSTM + CNF Word2Vec	Precision	0.78	0.71
	Recall	0.67	0.82
	F measure	0.72	0.77

embeddings clearly outperforms word level embeddings in addition to previous experiments with CNF with CRF. In addition the LSTM model employs fewer features. The results on the CRF and log linear models require part of speech tags and chunk tags in addition to the CNF concepts. These themselves require their own separate pipelines which owing to the issues pointed out in "Related work" section perform unreliably with Twitter text. On the RNN only the concept embedding is required. Assuming the CNF representation can be obtained directly this is probably a more reliable approach and definitely computationally cheaper particularly at run time.

What is remarkable is how poorly word level embeddings with the LSTM perform. This is significantly worse than even the previous work by Jimeno-Yepes et al. [21]. There are two ways to account for this. The first is that as far as we are aware there has not been any work in which there was a deliberate separation of training and test data such that test data and training data contained mentions of different diseases. The typical approach for separation of training and test data has been via a simple train/test split of the same dataset. Therefore in many cases it seems likely that at test time the models would simply be attempting to determine the correct labels for names of diseases that they had already encountered in training which is a far simpler task than if all disease mentions in the test data are previously unseen and the test data potentially has a different lexicon. This we feel is a great omission since the most persuasive argument for applying machine learning based named entity recognition in this scenario is the discovery of previously unseen ailments such as emerging diseases.

The second and more concrete reason why we observe such poor results is the fact that LSTMs are extremely powerful learners. When we couple this with the fact that we have a very large number of features relative to the number of examples particularly for the word level models means that there is a very large chance of over-fitting. For instance our training data comprises 73,848 tokens versus a vocabulary of 59,000 tokens; furthermore nearly 85% of the tokens appear only once. In such a scenario it is very difficult to create representative training data for good general models since data annotation is a manual task which is always resource constrained. Therefore, the more powerful the learner the higher the chance of over-fitting, bearing in mind that ours is a very contrived experiment we would assume it only gets worse for live data. On the other hand the CNF form has only 166 features. Assuming these features to be correct, it would be far easier to generate representative data for concepts than for a vocabulary with millions of tokens. With the CNF representation the length of the word level vocabulary is inconsequential as the procedure always compresses it into a fixed size vocabulary of just 166 concepts.

Error analysis

Although it is important to maximize the performance in terms of precision, recall and F_1 score, since it is impossible to obtain a perfect classifier we opine that it is equally important to be able to automatically determine false positives. We posit that given the classifier does not contain any systematic bias, its probability of incorrectly assigning the "ET" tag to tokens that are not names of diseases should be uniformly spread across all

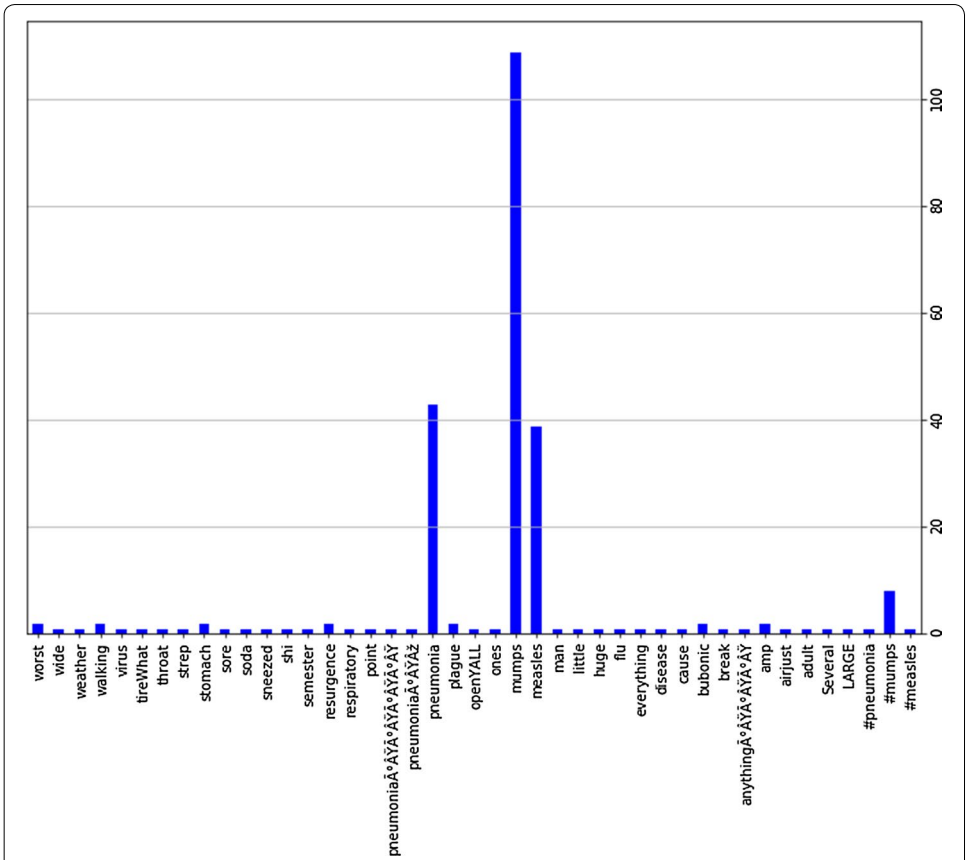

Fig. 10 Token-wise assignment frequency for ET tag

non-disease names and it should also assign the "ET" tag to disease names with a significantly higher probability than words referencing other entities.

Figure 10 is a plot of the token-wise frequencies for the "ET" tag revealing that indeed words referencing diseases are assigned the "ET" tag with a much higher frequency. Given this observation, it is possible to setup an automatic error detection solution by employing simple thresholding techniques. For instance if we decide to only consider those tokens for which the "ET" tag is assigned more than 5 times as true positives, we would essentially eliminate all false positives for our dataset.

Conclusion and future work

As far as we can tell the LSTM + Word2Vec + CNF is the best performing approach so far on this specific named entity recognition task in literature and unlike previous approaches we have shown that this method is generalizable to lexically divergent data sets. The typical experimental setup has relied on obtaining some examples, randomizing them to ensure a uniform lexical distribution and then splitting the data into training and test portions. In our setup we train a model on one set of messages talking about a given disease and test the model on data sets containing messages talking about other diseases. We find that the resulting model performs quite well and therefore the ontology boosted approach makes an automated general threat detection system feasible.

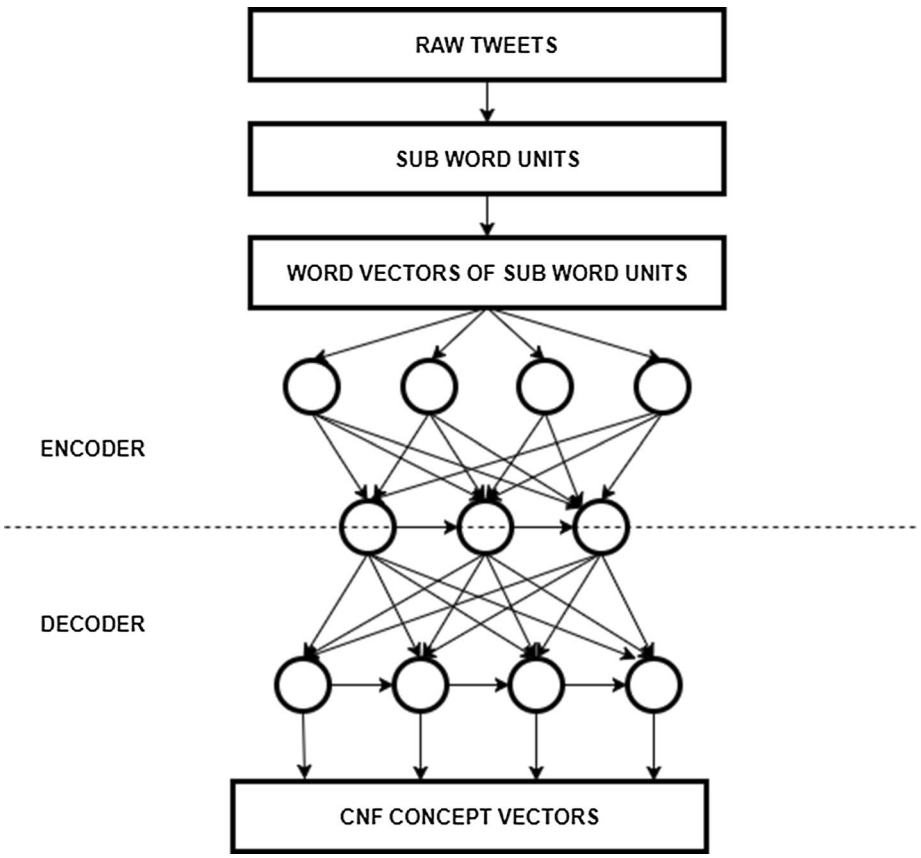

Fig. 11 Propositional neural-linguistic pipeline for feature encoding

Regarding further reducing the computational cost of the pipeline, we propose a neural linguistic pipeline using a recurrent auto encoder–decoder architecture similar to the one used by machine translation systems like Google Neural Machine Translation (GNMT) to enable us to perform these tasks in fewer steps. That is word vectors of tweets may be trained directly against concept vectors of the CNF representation. Figure 11 is a depiction of the propositional setup. For additional robustness we can employ vectors of sub word units similarly to the setup used by GNMT. The method would require parallel data sets of tweets and their equivalent CNF form. With the current setup large datasets of these can be feasibly obtained. In addition, with the correct recurrent neural architecture it would also be possible to combine the classification and named entity recognition tasks.

Finally, with the appropriate recurrent neural network architecture it is possible to combine the two tasks namely classification and named entity recognition. This would allow an end to end textual analytical pipeline in which we can simultaneously check if a message is relevant and also recover disease mentions in a single step.

Authors' contributions

MAM conceptualized the idea, prepared the datasets, designed and run the experiments, analyzed results, drafted and proof-read the manuscript, PN and EM provided general guidance concerning the work flow of the solution, verified datasets, proof-read and revised initial versions of the manuscript, verified the results. All authors read and approved the final manuscript.

Author details
[1] Department of Information Systems, Makerere University College of Computing and Information Sciences, Kampala, Uganda. [2] Department of Computer Science, Makerere University College of Computing and Information Sciences, Kampala, Uganda.

Acknowledgements
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Availability of data and materials
The best performing Keras HDF5 LSTM model and the datasets are available this Google drive folder: https://drive.google.com/open?id=18D4efjV4800IW311OUdDADQn7gum6_3A, Google's 6 billion word corpus word2vec model can be downloaded here https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit.

Consent for publication
We, the authors, consent to the publication of this manuscript in the Journal of Big Data.

Ethics approval and consent to participate
Not applicable.

Funding
Not applicable.

Publisher's Note

Received: 21 June 2018 Accepted: 21 August 2018
Published online: 10 September 2018

References

1. Culotta A. Towards detecting influenza epidemics by analyzing Twitter messages. In: Proceedings of the first workshop on social media analytics. SOMA'10. ACM, New York, NY, USA. 2010. p. 115–22. http://dx.doi.org/10.1145/1964858.1964874.
2. Eiji A, Sachiko M, Mizuki M. Twitter catches the flu: detecting influenza epidemics using Twitter. In: Proceedings of the conference on empirical methods in natural language processing. EMNLP'11. Association for Computational Linguistics, Stroudsburg, PA, USA. 2011. p. 1568–76. http://dl.acm.org/citation.cfm?id=2145432.2145600. Accessed 8 July 2018.
3. Paul MJ, Dredze M. A model for mining public health topics from Twitter. Health. 2012;11:6–16.
4. Denecke K, Nejdl W. How valuable is medical social media data? Content analysis of the medical web. Inf Sci. 2009;179(12):1870–80.
5. Diaz-Aviles E, Stewart A. Tracking twitter for epidemic intelligence. Case study: EHEC/HUS outbreak in Germany. 2011. p. 82–5.
6. YoussefAgha AH, Jayawardene WP, Lohrmann DK. Role of social media in early warning of norovirus outbreaks: a longitudinal Twitter-based infoveillance. In: Proceedings of the international conference on data mining (DMIN). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp). 2013. p. 1.
7. Polgreen PM, Chen Y, Pennock DM, Nelson FD, Weinstein RA. Using internet searches for influenza surveillance. Clin Infect Dis. 2008;47(11):1443–8. http://dx.doi.org/10.1086/59309810.1086/593098. http://arxiv.org/abs//oup/backfile/contentpublic/journal/cid/47/11/10.1086/593098/2/47–11–1443.pdf.
8. Garimella VRK, Alfayad A, Weber I. Social media image analysis for public health. In: Proceedings of the 2016 CHI conference on human factors in computing systems. ACM. 2016. p. 5543–7.
9. Zephoria.com. The top 20 valuable Facebook statistics. 2017. https://zephoria.com/top-15-valuable-facebook-statistics/. Updated Nov 2017. Accessed 22 Nov 2017.
10. Newberry C. 28 Twitter statistics all marketers need to know in 2018. 2018. https://blog.hootsuite.com/twitter-statistics/. Accessed 8 July 2018.
11. Ginsberg J, Mohebbi MH, Patel RS, Brammer L, Smolinski MS, Brilliant L. Detecting influenza epidemics using search engine query data. Nature. 2009;457:1012. https://doi.org/10.1038/nature07634.
12. Freifeld CC, Mandl KD, Reis BY, Brownstein JS. Healthmap: global infectious disease monitoring through automated classification and visualization of internet media reports. J Am Med Inform Assoc. 2008;15(2):150–7. https://doi.org/10.1197/jamia.M254410.1197/jamia.M2544.
13. Collier N, Doan S, Kawazoe A, Goodwin RM, Conway M, Tateno Y, Ngo Q-H, Dien D, Kawtrakul A, Takeuchi K. Biocaster: detecting public health rumors with a Web-based text mining system. Bioinformatics. 2008;24(24):2940–1.
14. 2017. http://www.promedmail.org. Accessed 28 July 2017.
15. Mawudeku A, Blench M, Boily L, St John R, Andraghetti R, Ruben M. The global public health intelligence network. In: M'ikanatha NM, Lynfield R, Van Beneden CA, de Valk H, editors. Infectious disease surveillance. 2nd ed. Hoboken: Wiley; 2013. p. 457–69.
16. Ghiasvand O. Disease name extraction in clinical text using conditional random fields. M.sc. (Dissertation) University of Wisconsin-Milwaukee, USA. http://dc.uwm.edu/etd/495/. Accessed 28 July 2017.

17. Pallejà A, Horn H, Eliasson S, Jensen LJ. DistiLD database: diseases and traits in linkage disequilibrium blocks. Nucleic Acid Res. 2012;40:1036–40.
18. Forbes SA, Bhamra G, Bamford S, Dawson E, Kok C, Clements J, et al. The catalogue of somatic mutations in cancer (COSMIC)., Current protocols in human geneticsHoboken: Wiley; 2008.
19. Pletscher-Frankilda S, Pallejàa A, Tsafoua K, Binder JX, Jensen J. Diseases: text mining and data integration of disease-gene associations. Methods. 2014;74:83–9.
20. Magumba MA, Nabende P, Mwebaze E. Ontology driven machine learning approach for disease name extraction from twitter messages. In: 2nd IEEE international conference on computational intelligence and applications (ICCIA). 2017. p. 68–73.
21. Jimeno-Yepes A, MacKinlay A, Han B, Chen Q. Identifying diseases, drugs, and symptoms in Twitter. Stud Health Technol Inform. 2015;216:643–7.
22. Magumba MA, Nabende P. An ontology for generalized disease incidence detection on twitter. In: International conference on hybrid artificial intelligence systems. Springer. 2017. p. 38–51.
23. Le Q, Mikolov T. Distributed representations of sentences and documents. JMLR workshop and conference proceedings. Beijing, China. 2014. pp 1188–96.
24. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. Arizona: Google Inc. Curran Associates, Inc.; 2013.
25. Hornik K. Approximation capabilities of multilayer feed forward networks. Neural Netw. 1991;4(2):251–7. https://doi.org/10.1016/0893-6080(91)90009-t.
26. Brightplanet.com. Twitter Firehose vs Twitter API: what's the difference and why you should care?. 2013. https://brightplanet.com/2013/06/twitter-firehose-vs-twitter-api-whats-the-difference-and-why-should-you-care/. Accessed 25 May 2017.
27. Cunningham H, Maynard D, Bontcheva K, Tablan V. Gate: an architecture for development of robust HLT applications. ACL. Philadelphia, USA. 2002. p. 168–75.
28. Taylor A, Marcus M, Santorini B. The Penn Treebank: an overview. In: Abeillé A, editor. Treebanks., Building and using parsed corporaDordrecht: Springer; 2003. p. 5–22.