Information Systems Frontiers
https://doi.org/10.1007/s10796-018-9830-2

Weakly Supervised and Online Learning of Word Models for Classification to Detect Disaster Reporting Tweets

Girish Keshav Palshikar[1] · **Manoj Apte**[1] · **Deepak Pandita**[2]

Abstract
Social media has quickly established itself as an important means that people, NGOs and governments use to spread information during natural or man-made disasters, mass emergencies and crisis situations. Given this important role, real-time analysis of social media contents to locate, organize and use valuable information for disaster management is crucial. In this paper, we propose self-learning algorithms that, with minimal supervision, construct a simple bag-of-words model of information expressed in the news about various natural disasters. Such a model is human-understandable, human-modifiable and usable in a real-time scenario. Since tweets are a different category of documents than news, we next propose a model transfer algorithm, which essentially refines the model learned from news by analyzing a large unlabeled corpus of tweets. We show empirically that model transfer improves the predictive accuracy of the model. We demonstrate empirically that our model learning algorithm is better than several state of the art semi-supervised learning algorithms. Finally, we present an online algorithm that learns the weights for words in the model and demonstrate the efficacy of the model with word weights.

Keywords Machine learning · Text classification · Weakly supervised learning · Online learning · Transfer learning · Tweet classification · Disaster management

1 Introduction

With the ever widening spread of computers, communications and the Internet, social media is becoming increasingly important as a means of social interaction. In particular, social media has quickly established itself as an important means that people, NGOs and governments use to spread information during natural or man-made disasters, mass emergencies and crisis situations. In these scenarios, social media is used to report first-hand ("ground zero") experiences including photos and videos, near-hand observations, contact relatives and friends, request help, disseminate information about available help and other services, organize local search and rescue operations, monitor situation, report status, damages or losses etc. Given this important role, real-time analysis of social media contents to locate, organize and use valuable information for disaster management is an active research area; see Imran et al. (2015) for a comprehensive survey.

In this paper, we propose *a weakly supervised* learning algorithm to automatically detect social media content having information about disasters. We include natural disasters like earthquake, flood, hurricane, famine, forest fire, volcano eruption, tsunami, land slide, disease epidemic (e.g. H1N1, swine flu, ebola) and man-made disasters like nuclear power plant accidents. We exclude crimes, accidents, insurgencies, terrorist attacks and war.

Factual information about various kinds of disasters, as expressed in, say, news stories, is often similar at a broad level: they mostly include aspects like damage, injuries, deaths brought about by a disaster, search, rescue, relief, recovery, rehabilitation etc. We generalize this informal observation to a natural principle, which might be called as the **principle of information correspondence**: *in a given*

✉ Girish Keshav Palshikar
gk.palshikar@tcs.com

Manoj Apte
manoj.apte@tcs.com

Deepak Pandita
dpandit2@ur.rochester.edu

[1] TCS Research, Tata Consultancy Services Limited, 54B Hadapsar Industrial Estate, Pune 411013, India

[2] Department of Computer Science, University of Rochester, Rochester, NY 14623, USA

Published online: 22 February 2018

type of documents, similar events are expressed similarly. The similarity of expression of information is essentially at the level of semantics. Clearly, news and tweets are two distinct types of documents. Thus we may expect that different news items will broadly express information about disasters in a similar manner; and different tweets will broadly express information about disasters in a similar manner, although information expression across news and tweets need not be similar. Example text from news and tweets related to two types of disasters: earthquake and flood is shown in Table 1. Crimes, accidents, acts of terrorism, and even weather reports, may sometimes contain similar information as a disaster. Thus any technique for automatically detecting disaster-related information should reject such confounding pieces of information.

In this paper, our main goal is to build a model, with minimal supervision, that can be used to quickly classify tweets in an incoming stream as DISASTER-RELATED(+1) or NOT-DISASTER-RELATED(−1). We require the model to be simple, human-understandable (and even human-modifiable) and usable in a real-time scenario. A reason for this requirement is that the users of such a model (i.e., users who read and analyze disaster related tweets for further actions) find it easier to deal with word-based models in order to act quickly and clearly. Our word-based models have the advantage that these users can dynamically change and update them without too much efforts, in order to align the models with the drifts in the nature and content of tweets.

Toward this end, we propose self-learning algorithms that, with minimal supervision, construct models of information expressed in news about various natural disasters. The constructed model is extremely simple and basically consists of a set (bag) of characteristic words that are often present in information expressed about disasters.

Table 1 Some example news and tweets related to earthquakes and floods

A moderate earthquake with a magnitude of 5.5 struck Pakistan on Saturday, but no loss to life or property has been reported so far.
Record flooding from rain-swollen rivers has washed out hundreds of structures in Missouri, Illinois, Arkansas and eastern Oklahoma, forcing thousands to flee their homes, and 9.3 million Americans still face flood warnings. At least 28 people have died in the U.S.
Two tremors jolt central, southeastern Iran #Earthquake
#France - Seine River peaked on Saturday. Flood killed four. Thousands forced from homes

The algorithm constructs a single model for all disasters i.e., it does not differentiate among different classes of disasters, although our approach is easy to use for learning a model for a specific disaster type.

Since tweets are a different type of documents than news, we next propose a *model transfer* algorithm, which essentially refines the model learned from news by analyzing a large unlabeled corpus of tweets. We show empirically that model transfer improves the predictive accuracy of the model. We demonstrate empirically that our model learning algorithm is better than several state of the art semi-supervised learning algorithms. Once the word model is deployed to classify a stream of tweets, we often find that the model needs to be adjusted to handle drifts in the vocabulary used to report disasters. As a first step toward dynamically adjusting the word model for tweet classification, we propose an online algorithm that learns and automatically adjusts weights for each word in the initial word model.

The paper is organized as follows. Section 2 contains related work, Section 3 contains learning algorithms, Section 4 contains baselines, Section 5 contains experiments, Section 6 presents an online algorithm that learns weights for words in a model, and Section 7 outlines conclusions and further work.

2 Related Work

In this section, we summarize the work related to disaster detection, focusing primarily on semi-supervised learning and transfer learning. Imran et al. (2015) surveyed the computational methods to process social media messages and map them to the problems like detection of events, creation of actionable and useful summaries etc. Zhao et al. (2007) has used textual, social and temporal characteristics to detect events on social streams. Social text streams are represented as multi-graphs with nodes as social actors and information flows as edges. Events are detected by combining text-based clustering, temporal segmentation, and information flow-based graph cuts of the dual graph of the social networks. Sakaki et al. (2010) has built an earthquake reporting system based on event detection. They have used a classifier based on the keywords in tweet, the size of tweet based on number of words and the context to detect earthquake like event. Once an event is identified a probabilistic spatio-temporal model is built for finding the center and trajectory of the event. LITMUS (Musaev et al. 2014) is a landslide detection system which integrates USGS seismic data, NASA TRMM Rainfall network with Twitter, Instagram and Youtube. Social media data is filtered using keyword-based filtering, geo-tagging, classification and relevance score is computed to detect landslides.

Ritter et al. (2015) cast seed-based event extraction as a weakly supervised learning problem where only positive and unlabeled data is available. They regularize the label distribution over unlabeled examples toward user-specified expectation of the label distribution for the keyword. Zhou et al. (2012) presented a self-training algorithm that decreases the disagreement region of hypotheses. The algorithm supplements the training set with self-labeled instances. The instances that greatly reduce the disagreement region of hypotheses are labeled and added to the training set. Yang et al. (2009) developed a technique to identify evolution of relationships between the news events within a same topic. Event evolution identification technique automatically identifies event evolution relationships and represents as Event Evolution Graph (EEG). EEG helps to understand how events evolve along the timeline.

Our work is similar in spirit to query expansion, where the idea is to add suitable words to the user's initial query so as to improve the retrieval results. We do not survey those works here, except for Zhao et al. (2014), who present a query expansion algorithm, which they use to create a tweet graph and then present an anomaly detection method to specifically for detecting civil unrest related tweets in this graph.

There is a large amount of work in semi-supervised classification which uses a large amount of unlabeled data and a small amount of labeled data to build better classifiers. Nigam et al. (2000) introduced an algorithm for learning from labeled and unlabeled text documents based on the combination of Expectation-Maximization (EM) and a naive Bayes classifier. A classifier is trained using the labeled documents and probabilistically labels are predicted for unlabeled documents. Then a new classifer is trained using the labels for all the documents and iterates to convergence. We use a simplified version of this approach as a baseline. Davidov et al. (2010) utilized the semi-supervised sarcasm identification algorithm of Tsur et al. (2010) and proposed SA SI algorithm that successfully captures sarcastic sentences in twitter and other domains. The algorithm employs two modules: semi supervised pattern acquisition for identifying sarcastic patterns that serve as features for a classifier, and a classification stage that classifies each sentence to a sarcastic class.

Transfer learning involves leveraging knowledge learnt from source domain/task to improve learning in target domain/task. Dai et al. (2007) has proposed a transfer-learning algorithm for text classification based on an EM-based Naive Bayes classifier. First the initial probabilities under a distribution of labeled data set are estimated and then an EM algorithm is used to revise the model for a different distribution of the unlabeled test data. This approach has been used by Zhao et al. (2013) for crowd-selection on twitter. Guerra et al. (2011) analyzes sentiments by using opinion holder bias prediction. First, the bias of a social media user toward a specific topic is measured by solving the relational learning task over a network of users connected by endorsements. The sentiments are analyzed by transferring user biases to textual features. They show that even when the topic changes its profile as new terms arise and old terms change their meaning, the user bias helps in building more accurate classification models due to consistency over time.

Roy et al. (2012) have proposed SocialTransfer - a cross domain real time transfer learning framework. SocialTransfer uses topic space learned in real time via Online Streaming Latent Dirichlet Allocation and real time cross domain graph spectra analysis based transfer learning method. The Transfer Graph captures the relationships between videos and topics which cannot update itself due to the data constraints. The social social streams relationships with the topics are updated using the streaming social media data to mark the change in the topic profile. The updated transfer graph is then used for video recommendation and query suggestion for video search.

Online learning of classification is fast emerging as a new and practically useful setting for classification. Many online learning algorithms for classification assume an ensemble (or committee of experts setting) (Mohri et al. 2012). In our paper, we have followed and modified the classic perceptron algorithm for online learning, although our classification model is not a linear model.

Each tweet is a very short and a rather noisy document. Computing similarity (say, for clustering) between short text segments is a challenging problem, because simple document representations such as TF-IDF suffer due to the short sizes of the documents. Even semantic representations, such as word embeddings, suffer from similar problems, when constructed from a corpus of short documents. De Boom et al. (2016) propose a specialized representation learning method for short texts that creates a more semantic representation by weighing word embeddings created from Wikipedia and Twitter. A common way to measure similarity between two texts is to measure the similarity between their mean vectors - where the mean vector of document is computed as the mean of the embeddings of the words in that text. This simple approach has several limitations, and Kenter and de Rijke (2015) proposes a way to address some of them, by creating bins of the dimensions and measuring similarity over these bins.

3 Learning Algorithms

3.1 Weakly Supervised Model Learning from News

Our approach is two-fold. In the first *learning phase*, we learn a one-class classification model for identifying

documents of class +1 (which are disaster reporting news), as against any other kind of document (class = −1). The learnt model is in the form of a word set W, consisting of words which characterize only the class +1. We are *not* interested in characterizing class −1, and in that sense this is a one-class classification problem. The model W is learnt in a weakly supervised manner - the only "help" given to the model learning algorithm is in the form of a small labeled *seed set* D, where each document in D is labeled with class = +1 (i.e., each document is a known to be related to disaster), and a small set W_0 of known seed words which partially "characterize" class +1 (i.e., are related to disasters). In addition, a large corpus U of unlabeled documents is given i.e., none of the documents in U are labeled with any class label (either +1 or −1). Since the text in news is significantly different than that in tweets, we need to transfer this model to learn to classify tweets. That part is discussed in Section 3.2. Finally, in the *prediction (operation) phase*, we use the final model (i.e., the news domain model transferred to the tweets domain) to dynamically classify any incoming tweet as having class = +1 or not.

We represent each input document U_i as a *word tuples sequence (WTS)* σ_i, which is an ordered sequence of word tuples: $\sigma_i = \langle wt_1, wt_2, \ldots, wt_{N_i} \rangle$, where N_i is the number tokens in the document U_i and each wt_j, $1 \leq j \leq N_i$, is a word tuple. Each *word tuple* has the form $wt_j = (w_j, t_j, c_j, f_j)$, where w_j is a word token, t_j is its POS tag (using Stanford POS tagger), c_j is the *term frequency* i.e., the number of times this token occurs in the current input document irrespective of its POS tag, f_j is its *document frequency* i.e., the number of documents in the entire corpus U in which this token occurs irrespective of its POS tag. Thus each word tuple is essentially a feature vector for each word token in the document. Word tokens are considered after stop-word removal and stemming. We insert a dummy word tuple to mark the sentence boundaries. Note that if a word w occurs multiple times in the same document, σ_i will contain multiple word tuples corresponding to w; these word tokens may differ in the POS tag component, but otherwise they would be identical. If N denotes the number of documents in the corpus U, then one way to to compute the TFIDF for a word w_j is: $c_j \cdot log\frac{N}{f_j}$, where $N = |U|$ is the total number of documents.

The simplest way to select a set of words characterizing disasters from a corpus U would be based on TFIDF. Unfortunately, since the documents in U are not labeled, this method selects all sorts of words, very few of which are disaster related. Hence we need a different method. The algorithm $learn\_disaster\_model$ (Fig. 1) initializes the model W with the given set W_0 of characteristic keywords, plus words that frequently occur "around" words in W_0

```
algorithm learn_disaster_model
input D = {D_1, ..., D_k}; // k known disaster reporting news
input W_0 = {w_1, ..., w_n}; // n seed words related to disasters
input U = {U_1, ..., U_m}; // m documents with unknown class
output W; // output model (words related to disasters)
input n_D, n_0, n_1, n_inv, c_noun, c_verb, θ_0, window; // hyper parameters of the algorithm
W := W_0 ∪ GetContextWords(W_0, n_D, window, D);
flag := true; iter := 1;
while (flag and iter ≤ MaxIter)
    flag := false; // reset flag
    for (i = 0; i ≤ m; i++) // do for each document in U
        if (U_i is already marked as +1) then continue; endif
        W_1 := GetFrequentWords(n_0, {U_i}); // words in U_i with frequency ≥ n_0
        if |W_1 ∩ W_0| = ∅ then continue; endif
        sim := WordsetSim(W, W_1);
        if (sim ≤ θ_0) then continue; endif
        W_2 := ∅;
        foreach (word u ∈ W_1) do
            if (GetTokenFreq(u, {U_i}) ≥ n_1 and WNHasWord(u) and
                WNFreq(u, noun) ≤ c_noun and WNFreq(u, verb) ≤ c_verb and
                GetTokenFreq(u, U − {U_i}) ≤ n_inv) then W_2 := W_2 ∪ {u}; endif
        end foreach
        W := W ∪ W_2; Mark U_i as +1; flag := true;
    end for
    iter++;
end while
```

Fig. 1 Algorithm to learn the disaster word model

in the known disaster reporting documents D. Initially, all documents in U are marked as −1. The algorithm iteratively examines each document in U (among those which are still marked as −1), and checks if the label for this document can be changed to +1, as follows. If the set W_1 of frequently occurring words in this document does not have a significant overlap with the current model W, then this document is ignored currently i.e., its label continues to be −1. If the set W_1 does contain a significant overlap with the current model W, then this document is marked as +1 and is never considered again. But before going to the next document, the algorithm selects those words (if any) from W_1, which occur in WordNet but whose corpus count in WordNet is not "too high", and adds them to the current model. After finishing the examination of all documents in U, the algorithm continues to the next iteration, because the labels of some more documents may now change, if new words were added to W in the previous iteration. The algorithm stops after a user-specified number of iterations or if no words were added to W in the previous iteration.

The subroutine $GetContextWords(W_0, n_D, window, S)$ works as follows. For every word w in the given seed list W_0, compute the set X of all words (nouns or verbs only) which occur before or after w in a window of given size in any sentence in the documents in the given set of documents S. For each word x in X, find the number of documents in S in which x occurs and remove x from X if this frequency is less than the given threshold n_D. So far,

the output model is just a set of words. We later present another algorithm that "learns" a weight for each word w_i in the model. Alternatively, for the weight of a word w_i in the model, we could use its TFIDF score, or the conditional probability $P(w_i|class = +1)$.

3.2 Model Transfer Algorithm

Suppose we have a model W_{news} learned from news. The simplest approach would use the model learned from the news corpus as it is on tweets, to predict which tweets are related to disasters. We will show in Section 5.2 that this approach has less accuracy, because news and tweets are different types of documents in terms of the vocabulary and style of writing. We need to refine the model W_{news}, by removing and adding words, to construct a new model W_{tweet} (we focus on adding new words only). We propose a new transfer learning algorithm, $augment\_model$ (Fig. 2), to augment the model from a source domain (e.g., news) by examining the unlabeled corpus from the target domain (e.g., tweets). For this, we assume that we have an unlabeled corpus of tweets available (dataset $D4$).

A new word is added to the model if it co-occurs with "sufficient" frequency with a word in W_{news}. *Pointwise mutual information (PMI)* between two words u and v is defined as $PMI(u, v) = log\left(\frac{p(u,v)}{p(u)p(v)}\right)$ where, $p(u, v)$ is the probability of co-occurrence of u and v, $p(u)$ and $p(v)$ is the probability of co-occurrence of u and v respectively. Out of several available alternatives, we use PMI as a measure of similarity between a word in the model and any other word in the unlabeled corpus. We select top N_0 (we used $N_0 = 25$) having the highest PMI with any word in the model W_{news} and remove words not present in WordNet (unless they begin with #), or are named entities person, location, organization etc. We add the remaining words from this list to W_{news} to get W_{tweet}.

One issue with tweets is the prevalence of informally written words. We have found that most of the important disaster related words are not written informally in a tweet. Also, informally written words in a tweet (such as `plz, pls, lol, thx` etc.) do not tend to come into the model for the following reason. While we do remove some stopwords from tweets, we have used the presence of a word in WordNet as a major constraint for words in a model, which rejects such informal words creeping into the model.

algorithm augment_model
input $W = \{w_1, w_2, \ldots, w_m\}$; // model learned from news
input $T = \{t_1, t_2, \ldots, t_N\}$; // unlabeled corpus of tweets
input N_0; // number of words to add to the model
for $i = 1$ **to** N **do** // examine each tweet in T
  for each word token $u \in t_i$ **do**
    $count(u) + +$; // increment occurrence count of word u
    if $u \in W$ **then continue;**
    for each word $w \in W$ **do**
      if $w \notin t_i$ **then continue;**
      $count(w, u) + +$; // increment co-occurrence count
    end foreach
  end foreach
end for
$S :=$ Top N_0 words having the highest PMI value;
Remove non-WordNet words (except hashtags) and named entities from S;
return $W \cup S$;

Fig. 2 Transfer learning algorithm to augment a source model

3.3 Model-based Classification

Let a given document (a news or a tweet) d contain words (nouns or verbs) $W_d = \{v_1, v_2, \ldots, v_k\}$. We have a simple and efficient real-time algorithm $identify\_disaster\_tweet$ that can use the given model W to predict the class label for any given document d. Basically, if the similarity between the set W_d and the given model W is more than a user-specified threshold θ_1 then the algorithm predicts $class = +1$ (disaster related) else it predicts $class = -1$ (not disaster related). We use the Jaccard similarity between W and W_d, which is just $\frac{|W \cap W_d|}{|W \cup W_d|}$.

We have found that disaster related documents sometimes look similar to those related to crime, accidents, war, terrorism or weather forecasts. To reduce this confusion, we can use the corpus to create separate models for each of these class of documents. We then modify our model-based classification algorithms to use these *negative models* as follows. If the similarity between the set W_d and any given negative model W_{neg} is more than a user-specified threshold θ_2 then predict $class = -1$ for d. If d is not similar to any of the negative models, only then we use the previous rule to predict whether d is disaster related or not.

4 Baseline Methods

We have created some baseline methods to compare our approach with. Starting with a given set W_0 of "seed" keywords characterizing disasters, the algorithm $wordset\_expansion$ (Fig. 3) detects and adds other words (only nouns or verbs) in a given unlabeled corpus, which are very similar to those in W_0 i.e., it creates a single "cluster" of words, starting with a set of cluster prototype or representative words. The algorithm does not use the set of known disaster documents, nor does it impose any restrictions on the frequencies of the words to be added. The cosine similarity uses the word embeddings produced by GloVe (Pennington et al. 2014).

algorithm wordset_expansion
input $W_0 = \{w_1, \ldots, w_n\}$; // n seed words related to disasters
input $U = \{U_1, \ldots, U_m\}$; // m documents with unknown class
$C := W_0$; // initial set of words
$s_0 := \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} cosine(embed(w_i), embed(w_j))$;
$sim_0 := 2 \cdot \frac{s_0}{|C|*(|C|-1)}$; // avg similarity between model words
$h := \emptyset$; //hash-table of all words processed so far
for $(i = 0; i \leq m; i++)$ // do for each document in U
  Let $U_j = \langle u_1, u_2, \ldots, u_{j_a} \rangle$; //document as a sequence of word tokens
  for $j = 1$ **to** j_a **do**
    if u_j is present in h **then continue**; //already processed
    if u_j is not a noun **or** u_j is not a verb **then continue**;
    $s_1 := \sum_{p=1}^{|C|} cosine(embed(w_p), embed(u_j))$;
    $sim_1 := 2 \cdot \frac{s_0+s_1}{|C|*(|C|+1)}$;
    if $sim_1 > sim_0$ **then**
      $C := C \cup \{u_j\}; s_0 := s_1; sim_0 := sim_1$;
    end if
  end for
end for
return C;

Fig. 3 Algorithm to expand a set of given seed words

We designed a second baseline method which uses topic modeling ($topic\_based$). We extracted 300 topics using the mallet toolkit (McCallum 2002) on dataset $D1$ and manually labeled the topics as disaster-related or not. We found 7 topics (out of 300) to be related to disasters. For example, following are some words in one of the topics: `earthquake, ocean, warning, quake, tsunami, tremor, magnitude, strike, damage`. For any given document, mallet gives a topic distribution within that document. We used a simple classification rule that if the most frequent topic within a document D is one of the disaster related topics, then label D as $+1$ else as -1. We use this topic-based classification scheme as a baseline because, it is also weakly supervised, like our approach.

We used the semi-supervised classification method of Transductive SVM (Joachims 1999) as the third baseline; we used the SVM-Lite tool to train a Transductive SVM using dataset $D1$.

Our fourth baseline algorithm is a self-training algorithm $NB\_iterative$, which takes the same set of unlabeled and positive examples as given to $learn\_disaster\_model$, along with a small set $NegSet$ of known negative examples. In each iteration, it trains a simple Naive Bayes classifier on the current sets of positive and negative examples, and predicts a class label L_d for each document $d \in U$ with confidence c. If $L_d = +1$ and c is "sufficiently high" then it adds d to D and removes it from U; else if $L_d = -1$ and c is "sufficiently high" then it adds d to $NegSet$ and removes it from U; otherwise, d remains in U without any class label. After a specified number of iterations, the final Naive Bayes model is tested. This algorithm is also weakly supervised and it does not use the user-specified set of seed keywords.

5 Experimental Studies

5.1 Datasets

Training News Dataset D1 This dataset contains 9983 documents. Out of these, 10 earthquake related documents are explicitly labeled as $+1$ (DISASTER-RELATED), and are used as seeds. Among the 9973 remaining unlabeled documents, we know that there are 50 documents related to other disasters (e.g., hurricanes and floods), 60 each for crime, accidents and weather, although this information is *not* passed to the disaster model learning algorithm. The remaining 9743 unlabeled documents are randomly selected news items from the FIRE corpus, some of which may be related to disasters, crime, accidents or weather, but we do not know which ones. The FIRE (Forum for Information Retrieval Evaluation) corpus[1] contains 392,577 English news items from Indian newspapers such as The Telegraph and BDNews.

Test News Dataset D3 This fully labeled dataset consists of 2537 news items, out of which 162 are labeled $+1$ (they are related to several natural disasters) and 2375 are labeled -1. Among the latter, 2225 news items (labeled -1) are from the BBC news website (Greene and Cunningham 2006) corresponding to stories in five topical areas (business, entertainment, politics, sports and technology) from 2004-2005, and 50 each are news related to crime, accidents and weather.

Labeled Tweets Dataset D2 This dataset contains 1344 disaster related tweets (labeled $+1$) and 2696 non-disasters tweets (labeled -1), among the latter 100 tweets related to crime, 100 tweets related to accidents and 100 tweets related to weather. The tweets were labeled manually by us. The tweets labeled $+1$ were related to a variety of natural disasters, such as avalanche, cyclone, drought, flood, forest fire, landslide, tsunami, volcano as well as nuclear accidents and biological disasters.

Unlabeled Tweets Dataset D4 Using the Twitter Streaming API, we downloaded a corpus of 7,555,000 English language tweets from 12 to 19 September 2016. All tweets are unlabeled.

[1] http://fire.irsi.res.in/fire/data

5.2 Results

We trained the algorithm *learn_disaster_model* on dataset $D1$. We started with the following seed words: `disaster, die, death, ruin, earthquake, quake, avalanche, landslide, cyclone, famine, flood, forestfire, fire, tsunami, volcano, H1N1, flu, ebola, epidemic, outbreak, radiation, nuclear`. We fixed the values of the parameters as follows: $n_0 = 2, n_1 = 4, n_D = 5, n_{inv} = 10000, c_{noun} = 6, c_{verb} = 39, window = 10, \theta_0 = 0.085, \theta_1 = 0.025$. This specific model (M1) contained 40 words. Some of the words in the model (not present in the seed words) were: `aftershock, tremor, magnitude, rain, storm, damage, kill, collapse`. We used this model to predict disaster related tweets on dataset $D2$, which gave $F = 0.703$ (entry M1 in Table 2).

So far, these results *do not* use our transfer learning algorithm. Next, we started with the model $M1$ (as created by *learn_disaster_model* on $D1$) and used our transfer learning algorithm *augment_model* on $D4$, which led to the addition of these words to the model `erupt, lava, #prayforkorea`. We tested this augmented (transferred) model on $D2$ (with $\theta_1 = 0.025$), which gave a higher F-measure of 0.734, indicating the advantage provided by the transfer learning even in the unsupervised setting (entry M2 in Table 2).

Note that the model M2 does not use any negative models, such as the one for Accident. Hence, we started with an initial seed list of 29 words for Accident (e.g., `accident, crash, wreck, collide, sink, drown, injure, die, capsize`), trained on $D1$ with $\theta_0 = 0.07$, which resulted in a new model for Accident containing 38 words. Then we transferred this model to the tweets domain, using dataset $D4$, which resulted in a model containing 43 words. Finally, we used the model M2, along with this *negative model* for Accident with $\theta_1 = 0.07$, to modify the predictions of M2 as mentioned earlier on dataset $D2$ (entry M2b in Table 2). Since this model M2b has a better performance than M2, albeit only slightly, this validates our proposition that *negative models* have the potential to improve the prediction accuracy by reducing false positives of the Disaster model. As an example, the model M2 classifies the tweet `pakistan train crash deaths and injuries reported collision kills at least six people and injures more than` as class +1 (Disaster), but the model for Accident correctly recognizes this tweet as belonging to the Accident class, and hence it is not classified as Disaster by the model M2b. Finally, Table 2 also shows the results obtained to predict disaster related tweets on $D2$ using the various baselines discussed earlier.

Table 2 Experimental results on dataset $D2$

Algorithm	P	R	F
wordset_expansion	0.535	0.051	0.092
NB_iterative	0.753	0.434	0.551
transductive_SVM	0.485	0.598	0.536
topic_based	0.893	0.474	0.619
M1	0.804	0.625	0.703
M2	0.812	0.670	0.734
M2b	0.815	0.670	0.735

6 Online Modification of Word Model

6.1 Online Learning of Weights of Words

The word model obtained after the Model Transfer from the news domain to the tweets domain is static in terms of the words and also, no weight (i.e., importance) is associated with the words in the model. There is a need for an online learning algorithm that can dynamically adapt and modify this model, to cope with the wide variety of text in the incoming real-life tweets text, far wider than the limited corpus from which the word model is learnt. The changes to the word model are of three kinds: add new words, remove older words or change the weights of the words already in the model. In this section, we only examine the problem of dynamically adapting the weights of the words in the model.

We define a *weighted word model* as a set of words with a real number as a weight for each word: $M_{wt} = \{u_1 : w_1, u_2 : w2, \ldots, u_n : w_n\}$; here, $w_i \in \mathbf{R}$, and can be positive, negative or even 0. Note that the model learnt using the previous algorithm is *not* a weighted word model i.e., it has no weights attached to any word. So we need to first convert this model to a weighted word model. And then we need to adapt this initial weighted word model to cope with the variations in the text in the incoming tweet stream. We use a common algorithm (algorithm *learn_weights_online*) for both these purposes (Fig. 4), which is a slightly modified version of the perceptron algorithm.

The algorithm *learn_weights_online* takes a weighted word model M_{wt} as input. It examines tweets in an incoming stream in a sequential manner: if the incoming tweet has a true label, then it modifies the weights of words in the model (explained shortly), else it leaves the model unchanged. For every incoming tweet t (which has a true label y), the algorithm computes S which is the intersection of t and M_{wt}. It then computes the sum of the weights of

algorithm learn_weights_online
input $M_{wt} = \{u_1 : wt_1, \ldots, u_n : wt_n\}$; // input model with weights
input $T = \{T_1 : y_1, \ldots, T_k : y_k\}$; // tweets with ground-truth labels
input $\gamma, max\_iter, min\_error\_count$; // hyper parameters
$iter := 0$; $bias := 0$; $error := 0$
while $(iter \leq max\_iter)$
  $error := 0$; // number of updates done in this iteration
  foreach $(t \in T)$ // do for each tweet in T
    sum := bias;
    foreach (word $v \in t$) **do**
      if $v \in M_{wt}$ **then** sum += weight of v in M_{wt} **endif**
    end foreach
    if $sum > 0$ **then** $\hat{y} := 1$ **else** $\hat{y} := 0$; **endif**
    if $y \neq \hat{y}$ **then**
      foreach (word $v \in t$) **do**
        if $v \in M_{wt}$ **then** weight of $v+ = \gamma \times (y - \hat{y})$ **endif**
      end foreach
      $bias- = \gamma \times (y - \hat{y})$;
      $error++$;
    endif
  end foreach
  if $error < min\_error\_count$ **then break**; **else** $iter++$; **endif**
end while

Fig. 4 Algorithm to learn weights of model words online

the words in S, as per M_{wt}. If this sum is positive then the predicted label $\hat{y}$ for t is $+1$, else it is 0. If the predicted label does not match with the true label of t (i.e., $y \neq \hat{y}$), then the weights of only the words in S are modified by the magnitude of learning rate γ (γ is constant and is given as an input to the algorithm). If the weight of a word in M_{wt} becomes negative, it indicates a word that whose presence indicates some evidence of the tweet being a non-disaster tweet.

Let $M2$ denote the word model learnt using the model transfer algorithm. Since words in this model have no weights, we assume that each word in it has weight 0. We pass this initial weighted word model (with all weights 0) as input to the algorithm $learn\_weights\_online$, along with a finite dataset **D5** of labeled tweets, where the ground truth label for each tweet is the one predicted by $M2$ (along with the negative models for Accident and Crime). Then the output of this algorithm is a new weighted word model $M3$, in which each word has a sensible weight. In this case, each iteration of the algorithm processes all tweets in **D5**, and the algorithm stops either after a maximum number of iterations is reached or when there are very few weight updates. Note that the algorithm includes an extra feature to capture bias i.e., the intercept.

In the next stage, we use the same algorithm in the true online setting, where the weights of the words in $M3$ keep getting updated whenever it receives a tweet with a ground-truth label. We could use some active learning strategy (such as uncertainty sampling) to pick tweets for labeling by the user at run time.

6.2 Baselines

For experimental validation, we will use the new weighted word model (modified using the algorithm $learn\_weights\_online$) to predict the class label for each tweet in a test dataset. For comparison, we have created several baseline methods, which also produce a class label for each tweet in our test dataset.

The first baseline method $wordpair\_similarity$ computes the pairwise cosine similarity between each word in a tweet and each word in the model $M2$, using the word embeddings from GloVe (Pennington et al. 2014). Then it selects top 5 word pairs, computes the average of their cosine similarity and predicts class label $+1$ if this average is above a given threshold and 0 otherwise. We searched for several values of the threshold and report the best result.

We used Generalized Expectation Feature Labeling ($GEFL$) (Druck et al. 2008) as another baseline. This approach treats words in a document as features, and trains the maximum entropy document classifier with expectation constraints that specify affinities between words and class labels. For each word in our model $M2$, we provided a high probability value (0.9999) for that word belonging to the class +1 and a low probability value (0.0001) for that word belonging to the class 0. We then trained the MaxEnt classifier using our labeled dataset, with these probability values as constraints. To be fair, note that our constraints refer to only the class $+1$; we have not provided probabilities for any words outside the model (being too numerous). We are not using any standard classifier as a baseline, since we are working in an online setting.

As another baseline, we used the semi-supervised Label Propogation algorithm (LPA) (Zhu et al. 2003). We constructed a directed graph, where each word in a corpus (only if it was present in WordNet) formed a vertex and two words u, v were connected by two directed edges (uv and vu), each labeled with the pairwise mutual information (PMI) value between the two words computed from the corpus. We added an edge between a word pair only if the two words co-occurred in at least 5 tweets. We labeled the vertices (corresponding to the 40 words from the model M2) as +1 and we manually selected 40 non-disaster words and labebeled the corresponding 40 vertices as -1. We used the $D5$ corpus (explained below) to construct this graph, which had a total of 11,755 vertices and 2,11,444 edges.

6.3 Dataset for Online Learning

Unlabeled Tweets Dataset D5 This is a subset of 755,427 tweets randomly selected from $D4$. $D5$ is used for learning the weights of words in the model $M2$ (phase 1 as discussed above). For each tweet t in $D5$, we produce a ground-truth label as follows: use model M2 to predict the label of t; if any of the negative models (say, for Accident) predicts label +1 for t then we make the label of t as 0; else we keep the same label as predicted by $M2$. Thus $D5$ is now a labeled dataset (the labels are not ground truth, but are produced by our algorithms). Since $D5$ is only used for initializing the weights in the model, we have chosen not to use the entire $D4$ dataset for this purpose.

Labeled Tweets Dataset D6 This is a separate, manually labeled, dataset containing 2250 tweets, out of which 658 are disaster related tweets (labeled +1) and 1592 are non-disaster tweets (labeled 0).

6.4 Experiments with Online Learning

As explained earlier, we start with the model $M2$ (in which each each word has weight 0), and use the algorithm $learn\_weights\_online$ on $D5$ to create a new weighted word model $M3$, with hyperparameter values as $\gamma = 0.001$, $max\_iter = 200$, $min\_error\_count = 1$. In $M3$, some of the top and bottom words (with highest and lowest weights) are: `earthquake, damage, death, disaster`, and `#prayforkorea`. Thus $M3$ is the same as $M2$. except that each word now has a weight.

Next, we use the algorithm $learn\_weights\_online$ with $D6$ and model $M3$ as input, with the same hyperparameter values as above, to learn another model $M4$, in which the weights of $M3$ are modified. We now use model $M4$ to predict the class labels for the test dataset $D7$, using the same prediction formula mentioned in algorithm $learn\_weights\_online$. Baseline $wordpair\_similarity$ is used directly to predict class labels for tweets in $D2$ (we report the best result for among various threshold values). We train the GEFL baseline on dataset $D6$ (as explained earlier) and use the trained MaxEnt classifier to predict the class labels for the tweets in test dataset $D2$. We also prepared the LPA model using the Label Propagation algorithm, as explained earlier. Running the LPA algorithm for 100 iterations and with $\epsilon = 0.00001$, on the graph constructed on $D5$ corpus, resulted in 192 vertices labeled +1 (including the initial 40 vertices); we call this model LPA. We used this word model to classify the tweets, as explained earlier. Note that the performance of $M3$ and $M4$, is better than all the three baselines (see Table 3). Note also that $M4$ performs better than all models, $M1$, $M2$, $M3$, indicating that the weighted version of the word model works better. It also outperforms all the baselines. As seen, there is a big jump in the accuracy of the models from $M3$ to $M4$, justifying our step of learning of the weights of the words in the model.

Table 3 Experimental results for Online Learning on dataset **D2**

$Model$	P	R	F
$wordpair\_similarity$	0.734	0.867	0.795
$GEFL$	0.333	1.0	0.499
LPA	0.647	0.464	0.541
$M3$	0.778	0.571	0.658
$M4$	0.901	0.810	0.853

So far, we have only tried to adapt the weights of the words in the model, but we did not add any new words to the model. We have also experimented with addition of new words to the model using the following strategy. Suppose we get a tweet t with a known ground truth (class label y), which is misclassified by the current model. Then we add all those words from t to the current model, with initial weight $= (y - \hat{y}) \cdot \gamma$, provided the words are not currently in the model and each word satisfies our constraints as specified in $learn\_disaster\_model$. The weights of these newly added words then gets updated (as per the new model's predictions for subsequent tweets) just like any other word in the model. We start with model $M4$ and simultaneously add new words to the model and modify the weights of words in the model to get another model $M4b$. Unfortunately, we found that the word addition quickly diverges, and the model keeps getting larger and larger. We are looking for a strategy (e.g., based on word embeddings) which imposes a tight check on the words before they get added to the model. One possibility is to use similarity of a new word with the words already in the model (e.g., using word embeddings). Another possibility is to change the strategy for adding new words: add new words only if $y = 1$ and $\hat{y} = 0$ i.e., focus on improving recall.

7 Conclusions and Further Work

In this paper, we proposed a weakly supervised algorithm to learn a bag of words model for various disasters from news corpus and then proposed a simple model transfer algorithm that augments the news-based model from a corpus of unlabeled tweets. Then we proposed a simple online learning method to enhance this model in a dynamic run-time setting. The proposed algorithms perform better then several baselines based on semi-supervised learning approaches. We also demonstrated the effectiveness of this approach on completely unseen stream of tweets. The learned model, being simply a bag of words, is easy for humans to understand and modify.

We are planning to use this approach to detect other kinds of information, such as HIV/AIDS related tweets. We are currently not allowing non-WordNet words to enter into the model, which restricts the effectiveness of the model, because tweets often contain domain-specific words not present in WordNet including hashtags (e.g., `H1N1, forestfire, ebola` etc.). We need to lift this restriction for wider applicability of this technique. Another limitation of this approach is that the model structure is very simple; we are investigating more complex model structures to improve the performance. For example, one can include phrases, rather than single words, in the model. We have developed techniques (based on χ^2-test and test of proportions) to detect any decay in the importance of words in the model (e.g., hashtags) over time. Online learning setting here only modifies the word weights; we are exploring an optimal way to add new words or drop less useful words, based on emerging vocabulary. We are also exploring other model modification mechanisms as well as other online learning algorithms to modify the model on the fly during operational deployment.

References

Guerra, P.H.C., Veloso, A., Meira, W.Jr., Almeida, V. (2011). From bias to opinion: a transfer-learning approach to real-time sentiment analysis. In *Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 150–158): ACM.

Dai, W., Xue, G.-R., Yang, Q., Yong, Y. (2007). Transferring naive bayes classifiers for text classification. In *Proceedings of the national conference on artificial intelligence 1999* (Vol. 22, p. 540). Menlo Park, CA; Cambridge, MA; London: AAAI Press; MIT Press.

Davidov, D., Tsur, O., Rappoport, A. (2010). Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the fourteenth conference on computational natural language learning* (pp. 107–116). Association for Computational Linguistics.

De Boom, C., Van Canneyt, S., Demeester, T., Dhoedt, B. (2016). Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognition Letters*, *80*(C), 150–156.

Druck, G., Mann, G., McCallum, A. (2008). Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval* (pp. 595–602). ACM.

Greene, D., & Cunningham, P. (2006). Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings 23rd international conference on machine learning* (ICML06) (pp. 377384). ACM Press.

Imran, M., Castillo, C., Diaz, F., Vieweg, S. (2015). Processing social media messages in mass emergency: a survey. *ACM Computing Surveys*, *47*(4), 67:1–67:38.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the sixteenth international conference on machine learning (ICML 99)* (pp. 200–209).

Kenter, T., & de Rijke, M. (2015). Short text similarity with word embeddings. In *Proceedings of the 24th ACM international on conference on information and knowledge management, CIKM '15* (pp. 1411–1420).

McCallum, A.K. (2002). Mallet: a machine learning for language toolkit. http://mallet.cs.umass.edu.

Mohri, M., Rostamizadeh, A., Talwalkar, A. (2012). Foundations of machine learning the MIT press.

Musaev, A., De, W., Litmus, C.P. (2014). Landslide detection by integrating multiple sources. In *11th international conference information systems for crisis response and management (ISCRAM)*.

Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T. (2000). Text classification from labeled and unlabeled documents using em. *Machine Learning*, *39*(2-3), 103–134.

Pennington, J., Socher, R., Manning, C.D. (2014). Glove: global vectors for word representation. In *Empirical methods in natural language processing (EMNLP)* (pp. 1532–1543).

Ritter, A., Wright, E., Casey, W., Mitchell, T. (2015). Weakly supervised extraction of computer security events from twitter. In *Proceedings of the 24th international conference on world wide web* (pp.896–905). ACM.

Roy, S.uman.D., Mei, T., Zeng, W., Li, S. (2012). Socialtransfer: cross-domain transfer learning from social streams for media applications. In *Proceedings of the 20th ACM international conference on multimedia* (pp. 649–658). ACM.

Sakaki, T., Okazaki, M., Matsuo, Y. (2010). Earthquake shake s twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on world wide web* (pp. 851–860). ACM.

Tsur, O., Davidov, D., name, A.R. (2010). Icwsm-a great catchy Semi-supervised recognition of sarcastic sentences in online product reviews. In *ICWSM*.

Yang, C.C., Shi, X., Wei, C.-P. (2009). Discovering event evolution graphs from news corpora. *IEEE Transactions on Systems, Man, and cybernetics-Part A: Systems and Humans*, *39*(4), 850–863.

Zhao, Q., Mitra, P., Bi, C. (2007). Temporal and information flow based event detection from social text streams. In *AAAI* (Vol. 7, pp. 1501–1506).

Zhao, Z., Da, Y., Ng, W., Gao, S. (2013). A transfer learning based framework of crowd-selection on twitter. In *Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1514–1517). ACM.

Zhao, L., Chen, F., Dai, J., Hua, T., Lu, C.-T., Ramakrishnan, N. (2014). Unsupervised spatial event detection in targeted domains with applications to civil unrest modeling. *PLOS ONE, 9*(10).

Zhou, Y., Kantarcioglu, M., Thuraisingham, B. (2012). Self-training with selection-by-rejection. In *2012 IEEE 12th international conference on data mining* (pp. 795–803). IEEE.

Zhu, X., Ghahramani, Z., Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *ICML* (pp. 912–919).

Girish Keshav Palshikar obtained M.Sc. (Physics) from Indian Institute of Technology, Bombay in 1985 and M.S. (Computer Science and Engineering) from Indian Institute of Technology, Madras in 1988. Since 1992, he is working in TCS Research, Tata Research Development and Design Centre (TRDDC), Pune, India, where he is now a principal scientist and leads the Machine Learning research group. In 2012, he was honoured with the title of TCS Distinguished Scientist. TCS Research is a part of Tata Consultancy Services. He is also a visiting lecturer at the Computer Science Department of University of Pune and Government College of Engineering, Pune. His areas of research include machine learning, data mining, text mining, natural language processing and their applications to various domains, including fraud detection and human resource management.

Manoj Apte received B.E. in Industrial Engineering from University of Pune, India in 1994 and M.Tech. in Industrial Management from Indian Institute of Technology, Madras, in 1997. He is working with Tata Consultancy Services (TCS) after completing his Masters and currently working with TCS Research, Tata Research Development and Design Centre (TRDDC) as a scientist. His research areas include applications of Data Analysis and Mining to real life situations in the industry including fraud detection, HR and social media analytics. He has worked on research problems in domains such as automotive, utilities, medical devices, finance, human resource and defence.

Deepak Pandita received B.E. in Information Technology from Govt. College of Engineering, Aurangabad, India in 2014. After completing his Bachelors, he worked with Tata Consultancy Services (TCS) in TCS Research, Tata Research Development and Design Centre (TRDDC) as a research and development engineer. Currently, he is pursuing M.S. in Computer Science at the University of Rochester, NY. His research interests include applications of Data Mining, Machine Learning, and Natural Language Processing (NLP). He has worked on research problems in social media analysis, human resource, and NLP.