A Comparison of Classification Models for Natural Disaster and Critical Event Detection from News

Tim Nugent, Fabio Petroni, Natraj Raman, Lucas Carstens and Jochen L. Leidner
Thomson Reuters, Research & Development,
30 South Colonnade, London E14 5EP, United Kingdom.
{*tim.nugent, fabio.petroni, natraj.raman, l.carstens, jochen.leidner*}*@thomsonreuters.com*

***Abstract*—We present a contrastive study of document-level event classification of a range of seven different event types, namely floods, storms, fires, armed conflict, terrorism, infrastructure breakdown and labour unavailability from English-language news.**

Our study compares different supervised classification approaches, namely Support Vector Machine (SVM), Random Forest (RF), Convolutional Neural Network (CNN) and Hierarchical Attention Network (HAN). While past systems for Topic Detection and Tracking (TDT) and event extraction have proposed different machine learning models, to date SVMs, RFs, CNNs and HANs have not been compared on this task.

Our classifiers are also informed by word embeddings trained on large amounts of high-quality agency news, which leads to improvements compared to the use of pre-trained embedding vectors.

We report a detailed quantitative error analysis.

***Keywords*-Event Extraction; Topic Detection and Tracking (TDT); Text Classification**

I. Introduction

Natural disasters such as floods and hurricanes, as well as other critical events including terrorism and strike action, tend to dominate media reporting when they occur, as news agencies seek to inform their audiences of the safety implications for those caught up in the event. Larger-scale events tend to be reported in two main phases - the initial breaking news period, followed by a longer tail of reports on the aftermath, recovery and reconstruction efforts. Examples include the reporting of Hurricane Katrina which made landfall in New Orleans in 2005, an event which dominated news in the weeks and months after the event. A decade later, new stories were still being published focusing on human stories of endurance and survival, as well as reconstruction and fortification of storm defences. The terrorist attacks which targeted London's public transport system in the same year made global headlines, but were also frequently referred to in reports covering the 2017 London Bridge attack. Journalist typically draw comparisons with historic events, and it is not uncommon for retrospective articles to reference multiple events spanning a range of times and geographies.

Detecting such breaking disaster events in news, and more recently in social media, is a key application of information extraction systems which can be used for a range of important tasks including guiding emergency response, informing policy makers and directing business decisions. Due to the high volume of event-related news that includes both breaking and follow-up stories, such a task is especially challenging for human operators who may have to analyse many hundreds of documents per day. Clearly, automated methods that include natural language processing and machine learning can come to our aid here.

To date, such approaches have been successfully applied to both topic detection [1], [2], [3], concerned with finding and following events in a stream of broadcast news stories, and First Story Detection (FSD) [4], where the task is to identify the first story to discuss a particular event. Here, our focus is on breaking news events that describe natural disasters and critical man-made events including floods, storms, fires, armed conflict, terrorism, infrastructure breakdown and labour unavailability. Rather than tracking such events or identifying the very first reports, we attempt to capture all instances of breaking news stories. In this paper, we present a comparison of a set of supervised learning methods for event type classification: Support Vector Machine (SVM), Convolutional Neural Network (CNN), Hierarchical Attention Network (HAN) and Random Forest (RF) are evaluated on a corpus of human-annotated agency news. Our findings re-affirm the importance of features and problem framing.

The contributions of this work can be summarized as follows:

- we have made a robust comparison of a range of machine learning algorithms to classify news articles into one of seven natural disaster and critical event types;
- our findings show that a Support Vector Machine classifier outperforms all other approaches, achieving a weighted f1-score score of 77.3%;
- the use of a word embedding model trained using disaster-related stories results in an improvement in performance.

978-1-5386-2715-0/17/$31.00 ©2017 IEEE

The rest of this paper is organised as follows: in Section 2, we describe related work. In Section 3 we provide details on the dataset we used in our experiments. In Section 4 we discuss a range of different machine learning models we have applied, the results of which are described in Section 5. Finally in Section 6, we summarise our findings and discuss future work.

II. Related Work

The Message Understanding Contest (MUC) was a series of US government research evaluations in information extraction, which included event extraction [5], [6]. The last evaluation, MUC-7, ran in 1996 and aimed to extract entities, relations and events from 158,000 New York Times articles. Two set of documents for training and test were extracted, to cover the aircraft accident and launch event domains, respectively.

There is a large body of literature that considers event detection as the task of "identifying instances of specified types of event in text". Progress in this area has in part been driven by the Automatic Content Extraction challenge [7], [8] (2000-2008) and associated ACE corpus. This competition aimed to improve the state of the art in the extraction of named entities, relations and events. A multitude of solutions have been suggested to solve the event detection task on the *ACE2005* data, ranging from feature-based methods [9], [10], [11], [12], [13] to neural approaches [14], [15], [16], [17]. However, this task is quite different from the one we are trying to solve in this paper. We are interested in classifying breaking news concerning natural disaster and critical event. The input to the classification method is the entire document rather than a single sentence, and the goal is to classify the news rather than detect event mentions in text, as is the case with ACE.

The problem we are tying to solve is also related to First Story Detection (FSD) [4], [18], [19], [20], which aims at classifying a document as a "first story" if the document differs significantly from those published before. In our setting, however, we are interested in multiple documents describing the same new event and we additionally want to classify events in a given set of categories.

EventMiner [21] is a probabilistic framework that leverages semantic annotations to identify important events. Our solution differs in that it does not require semantic annotations and explicitly focuses on breaking rather than historic news.

More recently, social media sites like Twitter have also been a source of attempts to mine events in general [20], with some focus on disaster events [22].

III. Dataset

A significant challenge when applying machine learning to specific event detection tasks is assembling a sufficiently large and representative data set. Data for our experiments was collected by querying an ElasticSearch index containing an aggregation of news articles from over 12,000 sources [23]. Articles were selected using representative keyword lists assembled for each event type, containing up to 16 terms, and spanning a four year period from 2012 to 2016. Such keyword-based queries typically returned articles belonging to multiple event categories. For example, the following three sentences all contain keyword *fire*:

"We're experiencing rapid rates of spread on the fire" said U.S. Forest Service spokesman Chris Gaulding.

The fire service in Hamburg said trees and branches were blown across roads and three people, as well as the man killed, were trapped in their cars.

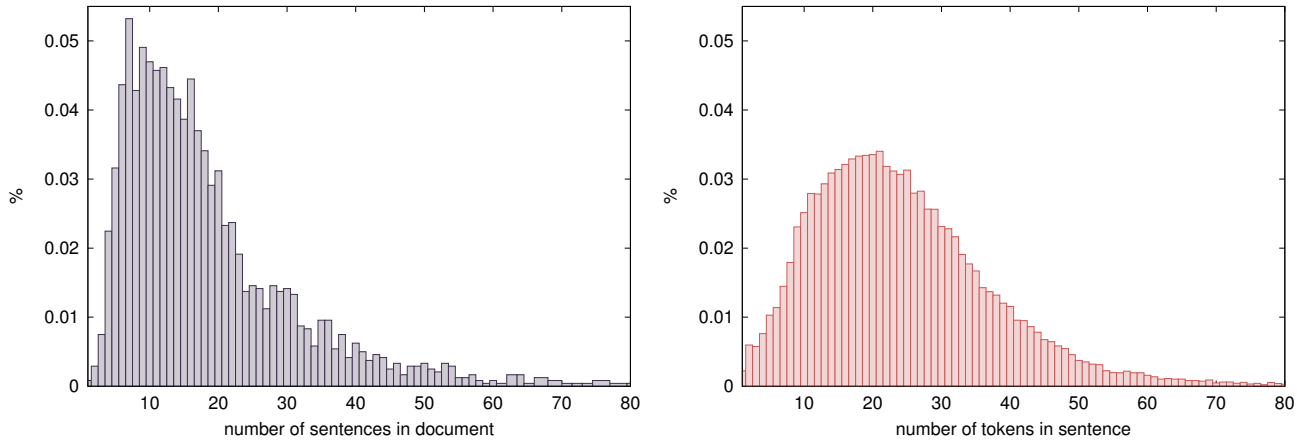
Meanwhile, at least 28 people were killed when a gunman opened fire in Sousse resort in Tunisia.

However, they correspond to event types *fire*, *storm* and *terrorism*, respectively. In order to correctly identify the event type for each article, we devised guidelines and employed annotators to perform labelling using the BRAT annotation tool [24]. Guidelines required that annotators labelled articles as events of a given type only when the article referenced concrete events that had occurred in the recent past, rather than historical events. Moreover the guidelines explicitly stated to ignore event-related terminology used in a metaphorical sense and to exclude articles whose focus was protection against such events (e.g., *flood defence programs*) but did not reference a specific event. Articles that failed these criteria were given a label of 'none'. Inter-annotator agreement metrics for a sample of 30 articles that were annotated by 12 annotators were Krippendorff's α=0.53 and Fleiss' κ=0.49, suggesting moderate agreement. In total, 2405 articles were annotated, summarised in Table I.

Table I: Article counts for each event type.

Event Type	Count
conflict	194
fire	203
flood	166
infrastructure breakdown	137
labour unavailability	61
storm	191
terrorism	107
none	1346
total	2405

The large fraction of articles labelled as 'none' gives an indication of how challenging event classification is using only a naive keyword-based approach. Clearly, natural language processing and machine learning-based approaches are required here to control the rate of false



(a) Number of sentences distribution

(b) Number of tokens distribution

Figure 1: Distributions of number of sentences in a news artice (1a) and number of tokens in a sentence (1b)

positives. Figure 1 shows the distribution of the number of sentences in a news article and the number of tokens in a sentence.

IV. Models

We have framed this problem as a multiclass supervised learning task, where the objective is to train a classifier capable of accurately predicting the discrete class label y_i, where $y_i \in$ {'conflict', 'fire', 'flood', 'infrastructure breakdown', 'labour unavailability', 'storms', 'terrorism', 'none'}, for a given news article x_i. We applied Support Vector Machine, Random Forest, Convolution Neural Network and Hierarchical Attention Network classifiers, using word embeddings as input features in each case. We experimented with a variety of embedding methods (such as word2vec [25], glove [26], fastText [27]). Moreover, we tuned each model independently in order to understand what was the optimal amount of text (i.e., number of top sentences) necessary to best capture the event information (see Section V for an empirical evaluation of this aspect). We additionally compared these methods to a baseline approach.

***Baseline**:* Our baseline approach uses matches to positive and negative gazetteer lists assembled for each event type. For instance, for event type *flood*, a phrase from the positive list might be *heavy rain fall*, while a negative term might be *flood the market*. By subtracting the count of negative from positive matches, the baseline returns the event type with the modal score. Moreover we weight each match according to its inverse distance (in tokens) from the beginning of the article before performing the summation, reflecting how the key descriptive aspects that define such events are typically found towards the beginning of news articles.

***Word Embeddings**:* A word embedding is a function mapping words to a fixed-dimensional vector of real numbers such that $W : words \rightarrow \mathbb{R}^n$ [28]. Embedding approaches attempt to preserve semantic and syntactic relationships within the resulting vector space. They have proved useful in a variety of natural language processing tasks [26] and perform especially well in text classification problems, where they alleviate issues associated with high-dimensional feature vectors. Here, we used the fastText library [27] which is able to efficiently learn word representations by exploiting subword information. The continuous skip-gram model is applied in order to represent words as the sum of character n-gram vectors, allowing morphologically rich languages to be efficiently modelled while also providing representations for unseen words. Resulting word vectors can also be averaged together to form good sentence representations [29].

We trained a fastText model using a combination of two data sources. As well as the filtered English Wikipedia dump from 2006, we included tokens extracted from Reuters New Archive (RNA) articles tagged with disaster or accident Reuters topic codes, allowing the model to capture the semantic structure of event-related news. In addition to the 124 million tokens in the Wikipedia set, 190 thousand articles from RNA provided a further 47 million tokens. The skip-gram model was trained with default parameters and a fixed dimensionality of size 200.

***Support Vector Machines**:* Support vector machines (SVMs) are supervised learning algorithms used for classification and regression analysis. They attempt to optimally

position a hyperplane in a high dimensional feature space such that it maximally separates the data points belonging to various classes, projected into this space from their original inputs using kernel functions [30]. We trained multiclass SVMs under a one-vs-rest setting in combination with a grid search of the hyperparameter space. We used the non-linear radial basis function (RBF) kernel, defined as:

$$K(\vec{x_i}, \vec{x_j}) = e^{-\gamma|\vec{x_i}-\vec{x_j}|^2} \quad (1)$$

where $\vec{x_i}$ and $\vec{x_j}$ are embedding vectors representing news articles a_i and a_j, and γ is a free parameter. A grid search over the hyper-parameter space (see Section V) confirmed that RBF kernel outperforms a selection of standard kernel functions.

Embedding vectors representing new articles are obtained by averaging together the word vectors in the input text [31]. In particular, given the tokenized text $T = \{t_1, ..., t_Z\}$ representing document d and given a pre-trained word embedding model that maps each token t_i to a word vector $\vec{w_i}$, the embedding vector representing d is computed as:

$$\vec{d} = \frac{\sum_T \vec{w_i}}{|T|} \quad (2)$$

We considered two variants of the SVM model: SVM$_{doc}$ and SVM$_{t+b}$. In SVM$_{doc}$ the article headline and the article body are condensed together in the input text. However, it has been shown in the context of information retrieval [32], [33] that the article headline by itself carries valuable information and should be considered in an independent fashion from the body. Following this idea in SVM$_{t+b}$ we adopted the following approach: we first derive two vector representations for the document, one for the title $\vec{d_t}$ and another for the body $\vec{d_b}$.; we then concatenate together these vectors $\vec{d_t} \frown \vec{d_b}$ to form the input to the SVM classifier. In practise, we observed that the vector obtained by concatenating the average of word embeddings corresponding to the headline sentence with the average of word embeddings derived from the sentences in the text provide the best results (see Section V).

Random Forests: Random forests [34] (RF) are an ensemble learning model for classification and regression that construct multiple decision trees and combine the output of the individual trees to output a result (for a classification task it outputs the mode of the classes). Also in this case, we considered two variants of the RF model, one that considers in input the concatenation of the title and body vectors (i.e., RF$_{t+b}$), and another the consider the document as a whole (i.e., RF$_{doc}$)

Convolutional Neural Networks: Convolutional neural networks (CNN) [35], [36], [37], [38] are a family of neural algorithms that apply layers of convolving filters to local features. In the context of natural language processing, the features associated with the input text usually take the form of distributed representations of words. In particular, given a tokenized text $T = \{t_1, ..., t_L\}$, the input to a CNN is represented as a text matrix A where the i-th row is the word vector representations of the i-th token in the input text. We denote the dimensionality of the word vectors by d, hence $A \in \mathbb{R}^{L \times d}$. Intuitively, the sentence matrix can be treated as the input image in computer vision, where convolution has been firstly applied. Convolution is performed through a set of filters. In contrast with computer vision, where a convolution filter might assume an arbitrary width, in the context of natural language processing filters slide over full rows of the matrix (i.e., they have a width equal to d). Their heights, instead, are an hyperparameter of the model and represent the number of adjacent rows (i.e., of adjacent tokens) considered jointly by the filter. Consider a filter $W \in \mathbb{R}^{d \times h}$ (i.e., with height h), its output can be computed as:

$$c_i = f(W \cdot A[i:j] + b) \quad (3)$$

where $A[i:j]$ denotes the sub-matrix of A from row i to row j, $b \in \mathbb{R}$ is a bias term and f is a non-linear function such as the hyperbolic tangent. This filter is applied to each possible window of words in the text $\{A[1:h], A[2:h+1], ..., A[s-h+1:s]\}$ to produce a convolutional feature map $c = [c_1, c_2, ..., c_{sh+1}]$. A pooling function is then applied to feature map c to extract a single scalar $\hat{c}$ from it. There are several non-linear functions to implement pooling among which max pooling is the most common, that is $\hat{c} = max\{c\}$. Intuitively this approach should capture the most important feature (i.e., with the highest value) as output from the filter. The CNN architecture is actually composed of multiple filters (with varying width) to obtain multiple features. These features are combined in the penultimate layer and are passed to a fully connected softmax layer whose output is the probability distribution over the classification labels. Figure 2 shows a graphical representation of the CNN architecture.

Hierarchical Attention Networks: Hierarchical Attention Networks (HAN) for document classification have been proposed in [39]. The idea is to combine several recurrent neural networks (RNNs) in a hierarchical architecture that reflects the hierarchical document structure (i.e., document/sentences/tokens). Assume that a document has N sentences $S = \{s_1, ..., s_k, ..., s_N\}$. Without loss of generality assume that each sentence contains M tokens.

The first stage of the architecture consist on a word encoder, that uses a bidirectional gated recurrent unit (GRU) [40]. Since not all words contribute equally to the representation of the sentence meaning, an attention mechanism is introduced to weigh the importance of the words. In particular, each state of the bidirectional GRU is

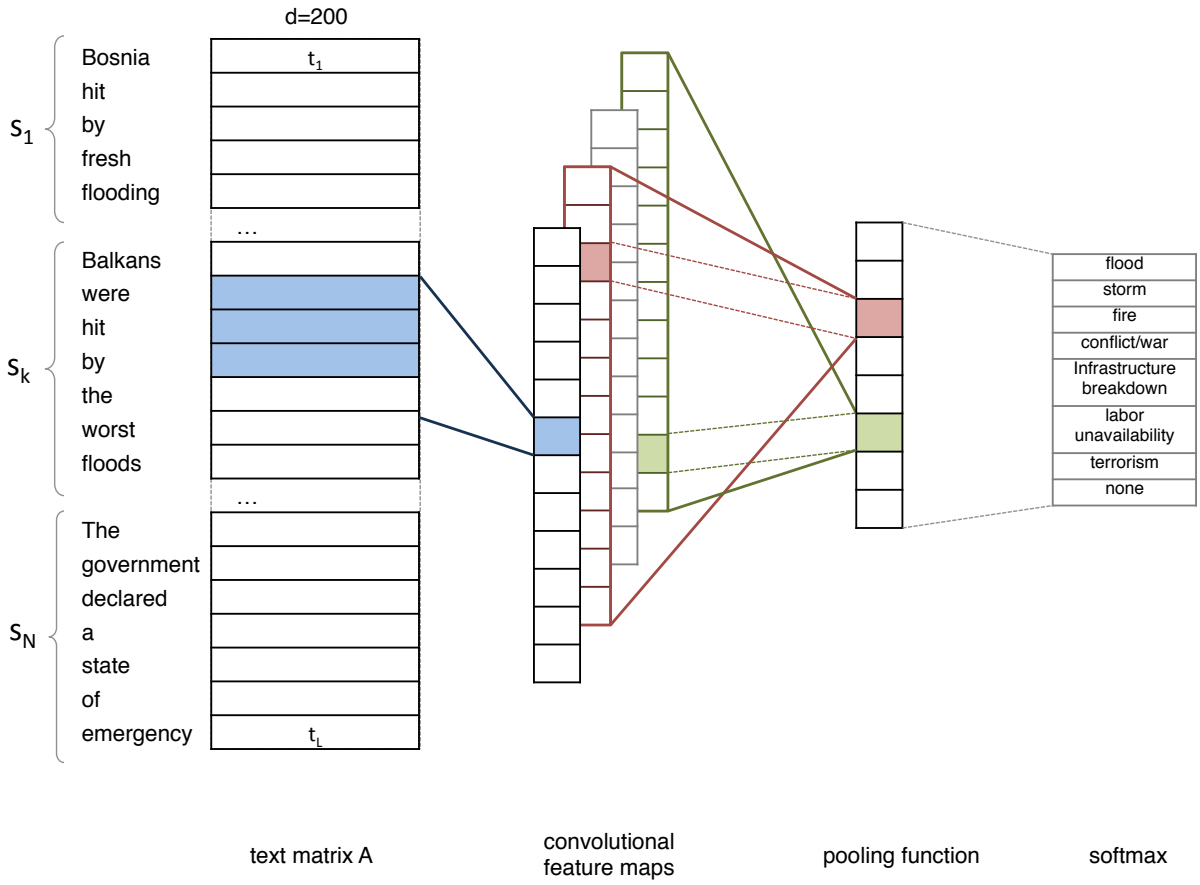


Figure 2: Convolutional neural network architecture

combined with a word level context vector u_w to return a normalized importance weight. The final sentence vector is computed as a weighted sum of the GRU states based on those weights. The word context vector u_w is randomly initialized and jointly learned during the training process. All the sentences vectors computed are then exposed to a sentence encoder stage, powered again by a bidirectional GRU. After that, a sentence-based attention mechanism is applied to reward those sentences that convey more information in order to correctly classify a document. In particular, a document vector v is computed by weighting the state of the GRU in a similar way as before (using this time a sentence level context vector u_s). The document vector v is a high dimensional representation of the document, that can be used as a feature for document classification, for instance through a softmax layer (as in the CNN case). Figure 3 shows a graphical representation of the HAN architecture.

V. Evaluation

We conducted an experimental study to compare the performance achieved by the considered classification models.

A. Methodology

We tested the performances of competing classifiers using a 5-folds cross validation approach. This technique consists of splitting the dataset into 5 folds of equal size (we additionally took care that the distribution of the classes were the same across all folds), and running 5 independent runs for each classifier, using one of the folds as test set and the remaining 4 folds as training set. The final result is computed by averaging the results of the 5 independent runs. The main advantages of cross-validation are that: (1) it reduces the variability of the result; (2) it mitigates the negative effect of potential noise in the data; (3) it allows to produce output prediction scores for the whole dataset. Moreover, we reported the average results over three independent runs for all models.

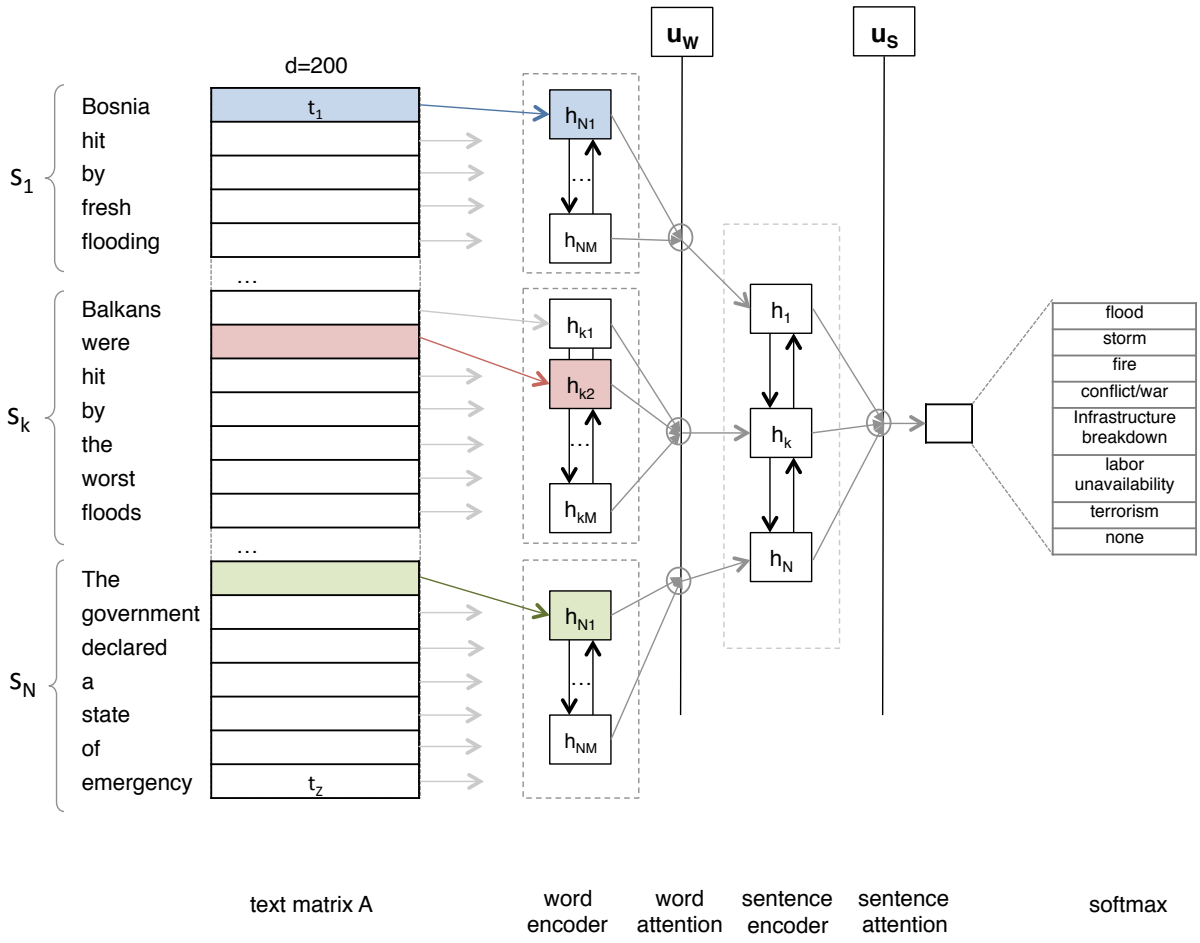


Figure 3: Hierarchical attention network architecture

As evaluation metric we used weighted precision, weighted recall and weighted F1-score.

B. System setup

We used the scikit-learn[1] Python library to train the SVM and the RF models, while we implemented both the CNN and the HAN models in TensorFlow[2].

All models have been tuned with a fine-grained grid search over the space of the hyperparameters. In particular, the SVM model has been trained with Radial Basis Function (RBF) kernel, C equal to 71.9 and automatic gamma. For RF we used 200 trees, no bootstrap, automatic number of features, 1 sample required to be at a leaf node. The CNN model has been trained using 400 filters of size 2, 3, 4, 5 and 6, considering the top 4 sentences in the document, the hyperbolic tangent (tanh) as non-linear activation function, a batch size of 64 training points and a learning rate of 0.001. Moreover, in order to regularize the model we used a dropout of 0.5 and we limited the norm of the convolution matrices to be smaller than 12. To train the HAN model we used Gated Recurrent Unit (GRU) cells of size 200, dropout probability of 0.6, limit over matrix norm of 10, learning rate of 0.001. Moreover we considered the top 21 sentences in the document and the first 50 tokens in each sentence.

For neural models, training was performed on a Nvidia Tesla K80 GPU.

[1]http://scikit-learn.org
[2]https://www.tensorflow.org

C. Results

Table II reports the results for the different algorithms considered. The first consideration is that the baseline approach performs poor, achieving a weighted F1-score of just 34.3%. This result is due to the simple rule-based approach of the baseline solution, and highlights the fact that considering just gazetteer lists of positive and negative keywords is inefficient to correctly classify news articles. The second consideration is that, among the non-neural

Table II: Classification results (expressed in percentages).

algorithm	weighted precision	weighted recall	weighted F1-score
baseline	67.9	40.0	34.3
RF_{doc}	70.1	71.7	69.0
RF_{t+b}	75.2	73.1	70.6
SVM_{doc}	74.8	75.6	74.7
SVM_{t+b}	**77.5**	**77.8**	**77.3**
CNN	76.1	76.3	75.9
HAN	76.2	75.4	75.6

models, SVM consistently outperforms Random Forest by a large margin for both the *doc* (+8%) and the $t+b$ (+9.5%) variants. Next we discus results for the neural models. If we focus on algorithms that consider the entire document (without differentiating between title and body) the two neural model considered (i.e., CNN and HAN) outperform both SVM_{doc} and RF_{doc}, achieving a weighted F1-score respectively of 75.9% (CNN) and 75.6% with respect to 69.0% of RF_{doc} and 74.7% of SVM_{doc}. However, CNN and HAN achieve those results by considering different portion of text. In particular, Figure 4 reports the performance in weighted F1-score for the considered algorithms by varying the amount of text considered in the document, expressed as the top-k sentences considered (note that in our convention the first sentence is the title). This figure clearly shows that the best performance for CNN (red line with up-facing triangles) and HAN (yellow line with pentagones) are achieved with different settings: CNN achieves its best weighted F1-score considering just the first 4 sentences, and after that point results decrease with increasing text size; the performance of HAN instead improves when considering more sentences, achieving its best weighted F1-score with 21 sentences in input. This behaviour is connected with the different architecture of the two neural models. In particular, the HAN architecture represents the input text in a tokens/sentences/document hierarchical structure, and this allows it to naturally deal with long documents. CNN instead collapses all the text in input into a flat representation (i.e., a really long sentence), and underperforms when this text is really long.

The best performing model in our evaluation is SVM_{t+b}, that achieves a weighted F1-score of 77.3%. The idea to explicitly separate the information contained in the title and in the body has already been successfully explored in the context of information retrieval [32], [33], with the intuition that the title is expected to provide stronger evidence on the task (in this case document classification) than the body. We have tried to exploit the same idea in CNN, by defining an architecture with two parallel convolutional layers, one for the text and one for the body, and concatenating the vectors after the pooling function just before the final softmax layer. However, the results for this latter approach (not reported in the paper) were worse than the original CNN version.

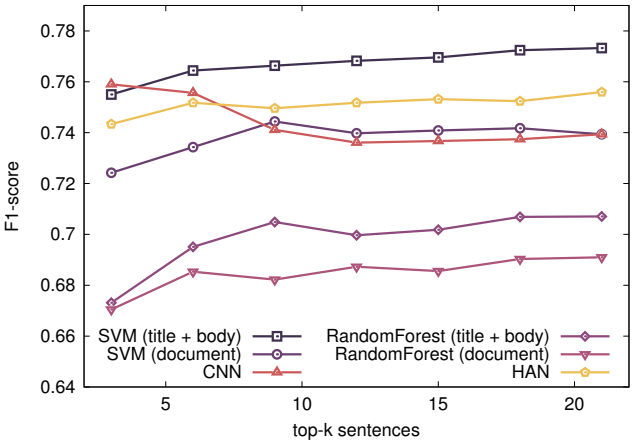


Figure 4: Performance in weighted F1-score varying top-k sentences considered in the model.

Regarding HAN, its architecture is supposed to learn the importance of each sentence with a dedicated attention layer, and should be able to automatically assign an higher score to the title if this conveys a stronger evidence. We believe a further reason for the advantage of SVM_{t+b} over neural models is the moderate amount of training points available. Neural networks approaches require a large amount of data in order to be able to learn and generalize due to the complexity (i.e., number of parameters) of their architecture. SVM instead gives good performance even with a relatively small dataset.

Finally, we investigated the impact of different embedding vectors on the performance of the models. Table III reports the performance in weighted F1-score achieved by the different considered algorithms with varying embedding vectors. In particular, we considered word2vec vectors pre-trained on Google News (*word2vec-GoogleNews*)[3], glove vectors of 200 and 300 dimension pre-trained on Wikipedia and Gigaword (*glove-glove.6B.200d* and *glove-glove.6B.300d*)[4], fasttext vectors trained on Wikipedia (*fasttext-wiki.en*)[5] and fasttext vectors on Reuters data as described in Section IV

[3]available at https://code.google.com/archive/p/word2vec
[4]available at https://nlp.stanford.edu/projects/glove
[5]available at https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

Table III: Performance with different embeddings models (expressed in percentages).

embeddings	dim	RF_{t+b}	SVM_{t+b}	CNN	HAN
word2vec-GoogleNews	300	69.3	76.9	75.5	75.2
glove-glove.6B.200d	200	69.1	75.8	74.0	73.5
glove-glove.6B.300d	300	68.9	76.2	74.1	73.9
fasttext-wiki.en	300	69.1	75.9	75.4	75.1
fasttext-reuters	200	**70.6**	**77.3**	**75.9**	**75.6**

(*fasttext-reuters*).

For comparison, we also measured the performance that the SVM_{doc} model achieves by using *tf-idf* (term frequency-inverse document frequency) document vectors as input to the classifier [41]. Note that with this approach each document is represented as a ~ 40k dimensional vector (one dimension for each word in the vocabulary), obtained by computing statistics over the corpus. The weighted F1-score achieved using *tf-idf* document vectors as input to SVM_{doc} is 57.3%. This result is considerably worse than those obtained by using pre-trained word vectors.

From Table III we can see that the best performance is achieved when the models use in input embeddings vectors pre-trained on news data (*word2vec-GoogleNews* and *fasttext-reuters*) rather than other sources (e.g, Wikipedia and Gigaword). This advantage is due to the common nature of the classification and embedding corpus (i.e., news articles both to pre-train embeddings and for the classification task): pre-trained vectors are able to capture domain-specific (news-specific) syntactic and semantic information that are beneficial for the classifiers. A further improvement can be achieved by training the model on fine-grained domain data: *fasttext-reuters* slightly outperforms *word2vec-GoogleNews* being trained on Reuters articles tagged with disaster or accident Reuters topic codes.

VI. Summary, Conclusions & Future Work

In this paper, we have compared several approaches to event type classification, and important part of first story detection in the context of Topic Detection and Tracking in information retrieval and event extraction in natural language processing. Our best SVM model (i.e., SVM_{t+b}) outperforms all other models, but with some interesting twists: CNNs perform more poorly the longer the stories are, whereas SVMs benefit from basing their decision on more textual evidence. RFs, popular in image/video processing, and also often used in text classification do not nearly perform at the same level as either CNNs or SVMs. Most strikingly, our simple "bag of words" SVM is beaten by a CNN, but the CNN is itself outperformed by an improved SVM model that separates titles and story bodies in separate spaces. Our HAN model outperforms the CNN model for news stories longer than seven sentences (or prefixes of longer stories of that length); the HAN also increases performance when the inputs get longer, whereas CNNs seem to do well for short stories up to six sentences.

Overall, our findings suggest that at moderate training set sizes, the received wisdom "clever feature engineering is more important than the model type" could be extended to "clever feature engineering, together with the right model type that benefits from it, matters". We were unable to find a similar benefit for the CNN from the same design, and would suggest for future work to explore a systematic theory of best practices in feature design – what works best for each model type? In future work, we also plan to explore new ways for combining custom-trained embeddings with off-the-shelf embeddings.

Acknowledgment

The authors gratefully acknowledge the support of Khalid Al-Kofahi, Mona Vernon and Shaun Sibley. Thanks to Thomson Reuters Global Resources for funding this work. We acknowledge the contributions, annotation work and/or feedback of: Zarko Panic, Leigh Henson, Simon Schmid, Vassilis PLachouras, Wandee Lee and Lynsey K. Lawrence. We would also like to thank 12 anonymous annotators.

References

[1] Y. Yang, T. Pierce, and J. Carbonell, "A study of retrospective and on-line event detection," in *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '98. New York, NY, USA: ACM, 1998, pp. 28–36. [Online]. Available: http://doi.acm.org/10.1145/290941.290953

[2] J. Allan, Ed., *Topic Detection and Tracking: Event-Based Information Organization*. Dordrecht, The Netherlands: Klouwer, 2002.

[3] D. Margineantu, W.-K. Wong, and D. Dash, "Machine learning algorithms for event detection," *Machine Learning*, vol. 79, no. 3, pp. 257–259, Jun 2010. [Online]. Available: https://doi.org/10.1007/s10994-010-5184-9

[4] J. Allan, V. Lavrenko, and H. Jin, "First story detection in tdt is hard," in *Proceedings of the Ninth International Conference on Information and Knowledge Management*, ser. CIKM'00. New York, NY, USA: ACM, 2000, pp. 374–381.

[5] N. A. Chinchor, "Overview of MUC-7," in *Proceedings of the Seventh Message Understanding Contest*. [Online]. Available: http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/muc_7_proceedings/overview.html

[6] B. M. Sundheim, "Overview of the third message understanding evaluation and conference," in *Third Message Understanding Conference (MUC-3): Proceedings of a Conference Held in San Diego, California, May 21-23, 1991*. Morgan Kaufmann, 1991, pp. 3–16. [Online]. Available: http://aclweb.org/anthology/M/M91/M91-1001.pdf

[7] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. Strassel, and R. M. Weischedel, "The automatic content extraction (ace) program-tasks, data, and evaluation." in *LREC*, vol. 2, 2004, pp. 837–840.

[8] "Annotation tasks and specifications," 2008, (online, cited 2017-09-26). [Online]. Available: https://www.ldc.upenn.edu/collaborations/past-projects/ace/annotation-tasks-and-specifications

[9] D. Ahn, "The stages of event extraction," in *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*. Association for Computational Linguistics, 2006, pp. 1–8.

[10] S. Liao and R. Grishman, "Using document level cross-event inference to improve event extraction," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 789–797.

[11] H. Ji, R. Grishman *et al.*, "Refining event extraction through cross-document inference." in *ACL*, 2008, pp. 254–262.

[12] P. Gupta and H. Ji, "Predicting unknown time arguments based on cross-event propagation," in *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. Association for Computational Linguistics, 2009, pp. 369–372.

[13] Y. Hong, J. Zhang, B. Ma, J. Yao, G. Zhou, and Q. Zhu, "Using cross-entity inference to improve event extraction," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011, pp. 1127–1136.

[14] T. H. Nguyen and R. Grishman, "Event detection and domain adaptation with convolutional neural networks." in *ACL (2)*, 2015, pp. 365–371.

[15] Y. Chen, L. Xu, K. Liu, D. Zeng, J. Zhao *et al.*, "Event extraction via dynamic multi-pooling convolutional neural networks." in *ACL (1)*, 2015, pp. 167–176.

[16] T. H. Nguyen, K. Cho, and R. Grishman, "Joint event extraction via recurrent neural networks." in *HLT-NAACL*, 2016, pp. 300–309.

[17] T. H. Nguyen, N. Fauceglia, M. Rodriguez-Muro, O. Hassanzadeh, A. M. Gliozzo, and M. Sadoghi, "Joint learning of local and global features for entity linking via neural networks." in *COLING*, 2016, pp. 2310–2320.

[18] S. Moran, R. McCreadie, C. Macdonald, and I. Ounis, "Enhancing first story detection using word embeddings," in *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR'16. New York, NY, USA: ACM, 2016, pp. 821–824.

[19] N. Panagiotou, "First story detection using entities and relations," in *Proc ACL*, 2016. [Online]. Available: http://aclweb.org/anthology/C16-1305

[20] S. Petrović, M. Osborne, and V. Lavrenko, "Streaming first story detection with application to twitter," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, ser. HLT'10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 181–189. [Online]. Available: http://dl.acm.org/citation.cfm?id=1857999.1858020

[21] D. Gupta, J. Strötgen, and K. Berberich, "Eventminer: Mining events from annotated documents," in *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval*. ACM, 2016, pp. 261–270.

[22] C. Castillo, *Big Crisis Data: Social Media in Disasters and Time-Critical Situations*. Cambridge, England, UK: Cambridge University Press, 2016.

[23] J. G. Conrad, X. S. Guo, and C. P. Schriber, "Online duplicate document detection: signature reliability in a dynamic retrieval environment," in *Proceedings of the twelfth international conference on Information and knowledge management*. ACM, 2003, pp. 443–452.

[24] P. Stenetorp, S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. Tsujii, "Brat: a web-based tool for nlp-assisted text annotation," in *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2012, pp. 102–107.

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

[26] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[27] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *arXiv preprint arXiv:1607.04606*, 2016.

[28] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[29] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[30] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[31] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," 2016.

[32] S. Robertson, H. Zaragoza, and M. Taylor, "Simple bm25 extension to multiple weighted fields," in *Proceedings of the thirteenth ACM International Conference on Information and Knowledge Management*. ACM, 2004, pp. 42–49.

[33] S. Robertson, H. Zaragoza *et al.*, "The probabilistic relevance framework: Bm25 and beyond," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[34] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[36] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[37] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[38] Y. Zhang and B. Wallace, "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification," *arXiv preprint arXiv:1510.03820*, 2015.

[39] Z. Yang, D. Yang, C. Dyer, X. He, A. J. Smola, and E. H. Hovy, "Hierarchical attention networks for document classification." in *HLT-NAACL*, 2016, pp. 1480–1489.

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

[41] G. G. Chowdhury, *Introduction to Modern Information Retrieval*. Facet publishing, 2010.