2014 IEEE Joint Intelligence and Security Informatics Conference

Threat detection in tweets with trigger patterns and contextual cues

Martijn Spitters, Pieter T. Eendebak, Daniël T.H. Worm, Henri Bouma
TNO
The Hague, The Netherlands
{martijn.spitters, pieter.eendebak, daniel.worm, henri.bouma}@tno.nl

***Abstract*—Many threats in the real world can be related to activities in public sources on the internet. Early detection of threats based on internet information could assist in the prevention of incidents. However, the amount of data in social media, blogs and forums rapidly increases and it is time consuming for security services to monitor all these sources. Therefore, it is important to have a system that automatically ranks messages based on their threat potential and thereby allows security operators to check these messages more efficiently. In this paper, we present a novel method for detecting threatening messages on Twitter based on trigger keywords and contextual cues. The system was tested on multiple large collections of Dutch tweets. Our experimental results show that our system can successfully analyze messages and recognize threatening content.**

I. Introduction

Many threats to people or critical infrastructures in the real world can be related to the activity of persons on social media, blogs and forums. Open-source intelligence (OSINT) is the information collection and analysis from publicly available sources. Law enforcement OSINT aims to predict, prevent and investigate attacks and assist in profiling suspects based on information from cyber space. The information is gathered to improve situation awareness for the protection of citizens and infrastructures. However, the amount of data on the internet rapidly increases and it is time consuming, if not impossible, to monitor the continuous flow of tweets, posts and announcements on websites manually. In The Netherlands alone, millions of tweets are posted each day. Therefore, it is important to have an system that automatically selects and ranks threatening messages, enabling security operators to check the relevant messages much more efficiently.

To date, little research has been published about recognizing death threats on social media. There are several approaches to perform OSINT for disaster management [1], incident detection by handcrafted rules and semantic filtering using DBpedia or news agencies [2], to handle limited availability of annotated data with transfer or semi-supervised learning [3], [4], but they are not optimized for the detection of threats by individuals and they do not exploit the relations between words in a message. Two approaches for recognizing threats in the Dutch language have been reported [5], [6], both based on N-grams. However, their performance is poor on large test sets. Recently, the approach in [6] was improved with promising results [7]. However, a side-by-side comparison with our work is not straightforward since the datasets used are different.

In this paper, we present a novel method for detecting threatening messages on social media based on trigger keywords and context cues. The main contribution is our two-stage classification setup, which allows for an efficient and highly scalable implementation. The first stage makes a pre-selection based on a simple one-class description of threatening tweets, while the second stage classifier further reduces this selection by applying a more computationally intensive classifier based on contextual relations between multiple words in the tweets. A second advantage of our approach is that it can be applied to other languages with minimal human effort, as it does not rely on complex handcrafted rules or patterns.

The outline of this paper is as follows. Our method is described in Sec. II. The data sets, experiments and results are presented in Sec. III, and the conclusions are given in Sec. IV.

II. Threat detection method

Our threat detection pipeline comprises a preprocessor, and two classifiers which act like a funnel. Figure 1 outlines this pipeline. The preprocessing step is described in II-A. The first classifier makes a first separation of the preprocessed tweets based on a semi-automatically generated list of threat keywords, which we will call *triggers*. Creation of this trigger list is described in II-B. The second classifier is only applied to the tweets that are not filtered out by the first classifier, which makes it a fast and scalable set-up. For the second classifier, we compared two different approaches; one based on mining contextual cues for the triggers (II-C), and one based on mining more structured word patterns for the triggers (II-D).

A. Preprocessing

As a first step, incoming tweets are cleaned. Retweet indicators (RT), URL's, and usernames are removed. Emoticons are replaced by placeholders to prevent problems with subsequent tokenization. Finally the tweets are tokenized, i.e. punctuation is removed and the tokens are translated to lowercase. Furthermore, for the training stage, tweets are deduplicated in order to prevent a bias towards heavily retweeted messages.

B. First-stage classification based on threat triggers

The first-stage classifier is basically a simple filter which weeds out the bulk of the incoming tweets. This filter uses a semi-automatically generated list of *threat triggers*. Both second-stage classifiers we compare in this paper (see following two sections) are designed around this starting point. Because tweets that do not contain any of these triggers will not be sent to the second-stage classifier, it is very important that this list of triggers covers as many potential threats as possible.

978-1-4799-6364-5/14 $31.00 © 2014 IEEE
DOI 10.1109/JISIC.2014.39

IEEE computer society

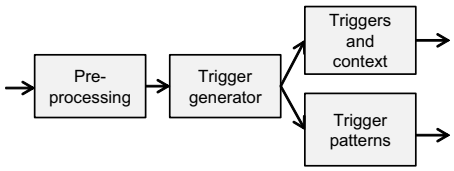

Fig. 1. Overview of our threat detection pipeline.

Therefore, while constructing this list, we focused on recall rather than on precision.

The trigger list is generated in a semi-automatic way from a labeled training set of threatening and non-threatening tweets (see III-A for dataset details). First, all tweets in the training set are *lemmatized*, thus all different inflected forms of a word are normalized to their lemma (e.g. stab, stabbing, stabbed all become *stab*). We used the lemmatizer of the Frog parser for Dutch [8]. Second, the most distinctive lemmas for the threatening tweets w.r.t. the non-threatening tweets are selected by computing Matthews correlation coefficient (MCC), optionally with Laplace smoothing (K) to guarantee that it is well-defined:

$$\frac{(\mathbf{tp}+K)\cdot(\mathbf{tn}+K)-(\mathbf{fp}+K)\cdot(\mathbf{fn}+K)}{\sqrt{(\mathbf{tp}+\mathbf{fp}+2K)(\mathbf{tp}+\mathbf{fn}+2K)(\mathbf{tn}+\mathbf{fp}+2K)(\mathbf{tn}+\mathbf{fn}+2K)}} \tag{1}$$

where $\mathbf{tp}$, $\mathbf{fp}$, $\mathbf{tn}$ and $\mathbf{fn}$ refer to the number of true/false positives/negatives. The MCC has successfully been applied as a feature selection metric for various text classification tasks [9]–[11]. The $\mathrm{MCC}(L)$ is computed for each lemma L with $K = 0$. Lemmas occurring in less than 3 tweets were ignored.

Subsequently, the resulting list of lemmas, ordered by their correlation to the class of threatening tweets, was manually cleaned. This boiled down to removing lemmas which *on their own* are not evidential for a potential threat. E.g., *dood* (dead), *steken* (stab), *schieten* (shoot), *mes* (knife), *bom* (bomb) were kept, and *jou* (you), *maken* (make), *moeder* (mother), *hoofd* (head), *slet* (slut) were removed. Lemmas with a negative correlation are ignored in this stage.

Finally, because for the first filter recall is much more important than precision, the selected lemmas were expanded again with all their inflected forms found in the full training set. This expanded list contains 172 triggers in total, and it forms the final list of threat triggers t_j used in the first stage of our classification process.

C. Second-stage classification based on contextual cues

Our first second-stage classification approach is based on mining positive and negative contextual cues for the triggers. The intuition behind this approach is that words which have a relatively high occurrence in the trigger's contexts in threatening tweets compared to the contexts in non-threatening tweets *reinforce* the chance of actual threatening content, and that words for which the opposite is true, *weaken* that chance.

1) Context word mining: Similar to the selection of our trigger list, we use MCC (see Eq. 1) to compute the correlations of the context words to the threatening and non-threatening contexts of the triggers in our labeled training set. A combined correlation score $\mathrm{MCC}(t_j, c_{i,j})$ is computed for each context word $c_{i,j}$ for each trigger t_j. In this computation we applied Laplace smoothing with $K = 0.1$. Note that in this approach all positional information is ignored, e.g. for the trigger word 'destroy' the tweets 'I will you destroy' and 'I will destroy you' are assigned the same scores. We only selected context words with correlation coefficient $|MCC| > 0.05$. For each trigger the number of positive and the number of negative context words were empirically restricted to a maximum of 18. Table I and Table II show examples of respectively reinforcing and weakening contextual cues, automatically mined for the triggers given in the first columns.

TABLE I. EXAMPLES OF REINFORCING CUES AUTOMATICALLY DISCOVERED BY THE CONTEXT METHOD.

Trigger	Positive cues
snijden (cut)	*bek* (mouth), *door* (through), *kanker/kkr/kk* (cancer), *ik* (I), *je* (you/your), *strot* (throat), *keel* (throat), *open* (open)
zien (see)	*homo* (gay), *jou* (you), *gaan* (will/go), *hoer* (whore), *maken* (make), *als* (if), *kanker/kkr/kk* (cancer), *dood* (death/dead)

TABLE II. EXAMPLES OF WEAKENING CUES AUTOMATICALLY DISCOVERED BY THE CONTEXT METHOD.

Trigger	Negative cue	Meaning of common expression
9mm (9mm)	*neerslag* (rainfall)	9mm rainfall
breken (break)	*hart* (heart)	broken heart
dood (dead)	*moe* (tired)	very tired
opblazen (blow up)	*ballon* (balloon)	blow up balloon
slaan (beat/hit)	*nergens* (nothing)	makes no sense

2) Context-based threat classification: Given the original correlation scores of the triggers, and the correlation scores of the context words, we can compute a final threat score for an unseen tweet in various ways. We have tried several heuristic combinations of the different trigger and context scores, of which the version given in Equation 2 generally yielded the best performance. Given a tweet T, we denote the trigger words contained in T by t_j, $j = 1, \ldots, n_T$, the corresponding context words contained in T by $c_{i,j}$, $i = 1, \ldots, n_j$ and the number of tokens in the tweet by $\mathrm{len}(T)$. The threat score of a tweet T is computed as follows:

$$S_{\text{context}}(T) = \frac{\sum_{j=1}^{n_T}\sum_{i=1}^{n_j}\mathrm{MCC}(t_j, c_{i,j})}{1+\mathrm{len}(T)} \tag{2}$$

For the sake of completeness we also give the basic score computation which does not take any context information into account in the following equation, as it is used as a baseline for our experiments reported later in Section III:

$$S_{\text{trigger}}(T) = \frac{\sum_{j=1}^{n_T}\mathrm{MCC}(t_j)}{\sqrt{1+\mathrm{len}(T)}} \tag{3}$$

D. Second-stage classification based on patterns

In our second approach we mine *threat patterns* from our training set of death threats using a sequence alignment technique which has its roots in the field of bioinformatics. We used the Needleman-Wunsch (N-W) algorithm [12], which is widely used for aligning protein and nucleotide sequences. It uses dynamic programming to find the optimal global alignment between two sequences, admitting gap insertions at a certain cost. For our purpose, we look at a tweet as a sequence of

words, and apply N-W to discover the typical word patterns used in threatening tweets. As far as we know, biological sequence alignment techniques have not been used for mining text classification patterns before. However, some work has been done for paraphrase alignment and sentence compression [13]. In the continuation of this section we will explain how we mine the patterns from our training set, and how we use them to rank unseen tweets by their threat potential.

1) Pattern mining: Our pattern-mining process consists of the following three steps. First, every tweet in our set of threatening training examples is aligned with every other tweet from the same set, using the N-W algorithm. Because our training set contains 3096 threatening examples, this initial step generates over 9 million alignments. As an illustration Table III shows two alignment examples, one for the tweet *'ik ga een bom in mijn school plaatsen'* (I'm going to place a bomb in my school), and one for the tweet *'ik schiet je door je hoofd bitch'* (I'll shoot you through your head bitch).

TABLE III. TWO ALIGNMENT EXAMPLES, DERIVED USING THE NEEDLEMAN-WUNSCH (N-W) ALGORITHM.

Seq1	ik	ga	een	**bom**	in	mijn	school plaatsen
Seq2	ik	ga morgen	een	**bom**	op	mn	school gooien
Pattern	ik	ga *	een	**bom**	*		school *
Seq1	ik	**schiet** je	door	je	hoofd	bitch	
Seq2	ik	**schiet** die Wilders	door	zn	kop	wacht maar	
Pattern	ik	**schiet** *	door	*			

In the second step of the mining process, the huge set of alignment patterns is radically reduced by (i) compressing sequences of multiple gaps and/or mismatches to a single matching expression which can be configured to match a sequence of n words (see II-D2), (ii) filtering out all patterns which do not contain at least one of the triggers, and (iii) deduplication.

The final step of the mining process creates clusters of patterns based on the threat trigger they contain. Each of these clusters is then expanded by generating 'sub-patterns'. For a given trigger-specific pattern we want to generate all possible sub-patterns still containing the trigger, as not all of them necessarily arise during the alignment procedure, but may still be useful for threat detection. As an illustration, Table IV shows the result of generating the sub-patterns for pattern *'jij bent * dood * maar * jij'* (you are * dead * but * you).

TABLE IV. GENERATED SUB-PATTERNS FOR A COMPLEX PATTERN.

Main pattern	jij bent * **dood** * maar * jij
Sub-patterns	jij bent * **dood** * maar
	jij bent * **dood** * jij
	dood * maar * jij
	jij bent * **dood**
	dood * maar
	dood * jij
	dood

2) Pattern-based threat classification: The threat patterns can be matched on unseen tweets in a real-time fashion. As described before, for a new tweet our classifier first checks if it contains one of the threat triggers. If it does not, the tweet is let through without further analysis. If the tweet contains a trigger, all patterns containing that trigger are matched on the tweet, starting with the longest one(s). If none of the longest patterns match, the second longest pattern(s) are matched, and so on. As soon as a pattern of a certain length matches, its length is used in the threat score computation, and no shorter patterns for the same trigger are matched anymore. If multiple matching patterns of the same trigger have equal length, the highest score is used. If the tweet contains multiple different triggers, the pattern matching process is repeated for each of the triggers. The '*' expression in a pattern can be configured to match one or more words. In our experiments we set it to match 0-5 words. The final scoring mechanism of the classifier combines the length of the matching pattern with its correlation to the threatening class. For each pattern we computed the MCC (Eq. 1, $K = 0$) on threatening and non-threatening training examples and used this value as a weight in the scoring computation. The threat score for a certain tweet T is then computed as follows:

$$S_{\text{pattern}}(T) = \sum_{j=1}^{n_T} \text{MCC}(p_j(T)) \cdot \text{len}(p_j(T))/M \qquad (4)$$

where n_T is the total number of triggers in tweet T, $p_j(T)$ is the longest matching pattern of the j-th trigger with the highest score among all matching patterns with equal length, and M is the maximum length of all patterns.

III. EXPERIMENTS AND RESULTS

A. Description of the data sets

For training and development, we used two Dutch datasets. The first set contains 3096 death threats and the same number of randomly selected, non-threatening tweets. The threats were taken from the publicly available website *doodsbedreiging.nl*, which aims to collect death threats expressed on Twitter. The second data set we used for development was created on March 5th 2014 by performing a Twitter scrape based on the list of trigger keywords of Sec. II-B, resulting in 75,435 tweets. Because this set was harvested using the threat triggers, it contains quite a lot of actual threats.

For testing and evaluation, we used two other Dutch datasets. Firstly, a set which we harvested on March 22nd 2014 by performing another scrape with the same trigger list, resulting in 74,330 tweets. Finally, we used an event-based collection, which was created around and during the coronation of the Dutch king on April 30th 2013, based on keywords related to the event (such as *kroningsdag, Beatrix, Willem-Alexander* etc.). This collection contains 157,048 tweets.

B. Experimental setup

Since it is very time consuming to annotate all tweets, for each evaluated method we ranked the test tweets by their scores. Only the top of these rankings were completely annotated in order to compute the method's precision. To obtain an estimate of the recall, a fraction (1 or 2%) was sampled from the remainder of the dataset. Volunteers (three native Dutch speakers) were instructed to indicate three threat categories: no threat, weak threat and strong threat.

In our experiments, we compared five approaches. The two methods presented in this paper (Context, Pattern) are compared to a baseline that only discards tweets that do not contain a trigger keyword (Random), a baseline that sorts

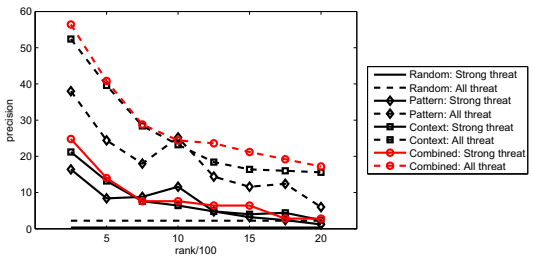

Fig. 2. Precision on the 74k test set of March 22nd.

tweets based on the trigger score of Eq. 3 (Sort), and a combined version of Context and Pattern (Combined, which is based on a weighted sum).

C. Evaluation on enriched set of 74k tweets

The results in Figure 2 and Table V show that the Context performs slightly better than the Pattern, and the combination results in a further improvement. In the top 1000 tweets of the combined method on the data of March 22nd, the precision of threats is 37.6%, the precision of strong threats is 13.5% and the related recall is approximately 54%. Based on 2% of the samples in the complete set we observe that 2.2% of the tweets is threatening. The related (binomial) 99% confidence interval of threats is between 1.3% and 3.4%. This indicates that all approaches give a significant improvement ($p < 0.01$) in the recognition of threatening content. For the detection of threats with the combined method, the 99% confidence interval on the top 1000 tweets is between 33.7% and 41.6%.

D. Evaluation on coronation set with 157k tweets

In the coronation set with 157k tweets, we were able to obtain a precision of 14% in the top 100 tweets with the Context approach. The 1% samples in the complete set that did not contain threats, allows us to estimate with 99% certainty that the expected number of threats in this set is below 0.3%. Based on this worst-case estimate (the boundary of the confidence interval instead of the expected value), we can conclude that in this range the precision is *at least* 47 times higher than without a system.

IV. Conclusion

In this paper, we presented a two-stage classification method for detecting death threats in tweets. The first stage uses trigger keywords as a pre-selection of potential threats based on positive evidence, which makes it practical to implement and extend. In the second stage, tweets which were not filtered out in the first stage are classified based on the statistical relation between triggers and their surrounding words. For this stage we compared two approaches. The Context approach mines positive and negative cues related to a specific trigger keyword, and the Pattern approach uses alignment to mine more structured threatening word patterns. We showed that using these reinforcing and weakening cues and patterns results in a huge improvement of threat-classification performance over using just a keyword list. Future work may include the development of a real-time early warning system for threats and a more effective combination of both approaches.

TABLE V. Precision in different rank ranges on the rich test set of 74k tweets.

Type	Method	Precision in a rank range		
		0k-1k	1k-2k	2k-74k (2%)
Strong threat	Trigger Random	0.3%		
	Trigger Sort	1.5%	1.6%	0.3%
	Trigger Pattern	11.3%	2.9%	0.1%
	Trigger Context	12.1%	3.9%	0.1%
	Trigger Combined	**13.5%**	**4.6%**	0.1%
All threat	Trigger Random	2.2%		
	Trigger Sort	5.2%	3.8%	2.3%
	Trigger Pattern	26.4%	11.1%	1.7%
	Trigger Context	35.9%	16.6%	1.9%
	Trigger Combined	**37.6%**	**20.3%**	1.8%

TABLE VI. Precision in different rank ranges on the coronation test set of 157k tweets.

Method	Precision in a rank range			
	0-100	100-200	200-300	300:157k (1%)
Random	0.0%			
Trigger Pattern	6%	4%	5%	0.0%
Trigger Context	14%	3%	0%	0.0%

Acknowledgment

The work for this paper was supported by the Dutch Ministry of Security and Justice in the program for safety and security research: "Veilige maatschappij" (Topic 3).

References

[1] G. Backfried, C. Schmidt, M. Pfeiffer *et al.*, "Open source intelligence in disaster management," in *EISIC*, 2012, pp. 254–258.

[2] F. Abel, C. Hauff, G. Houben, R. Stronkman, and K. Tao, "Semantics + filtering + search = twitcident. exploring information in social web streams," in *ACM Hypertext and social media*, 2012, pp. 285–294.

[3] D. Davidov, O. Tsur, and A. Rappoport, "Semi-supervised recognition of sarcastic sentences in twitter and amazon," in *Comp. Nat. Lang. Learning*, 2010, pp. 107–116.

[4] W. Pan, E. Zhong, and Q. Yang, "Transfer learning for text mining," in *Mining Text Data*, 2012, pp. 223–249.

[5] H. Bouma, O. Rajadell, D. Worm, C. Versloot, and H. Wedemeijer, "On the early detection of threats in the real world based on open-source information on the internet," in *ITSEC*, 2012.

[6] N. Oostdijk and H. van Halteren, "N-gram based recognition of threatening tweets," in *Comp. Ling. and Intel. Text Proc.*, 2013, pp. 183–196.

[7] ——, "Shallow parsing for recognizing threats in dutch tweets," in *Proc. IEEE/ACM Int. Conf. Adv. Social Networks and Mining*, 2013, pp. 1034–1041.

[8] A. van den Bosch, B. Busser, S. Canisius, and W. Daelemans, "An efficient memory-based morphosyntactic tagger and parser for Dutch," in *Comp. Ling. in the Netherlands*, 2007, pp. 99–114.

[9] H. Ng, W. Goh, and K. Low, "Feature selection, perceptron learning, and a usability case study for text categorization," in *ACM SIGIR Forum*, vol. 31, no. SI, 1997, pp. 67–73.

[10] M. Spitters, "Comparing feature sets for learing text categorization," in *Proc. Content-Based Multimedia Access*, 2000.

[11] F. Amardeilh, W. Kraaij, M. Spitters, C. Versloot, and S. Yrtsever, "Semi-automatic ontology maintenance in the virtuoso news monitoring system," in *IEEE EISIC*, 2013, pp. 135–138.

[12] S. Needleman and C. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J. Molecular Biology*, vol. 48, no. 3, pp. 443–453, 1970.

[13] J. Cordeiro, G. Dias, and G. Cleuziou, "Biology based alignments of paraphrases for sentence compression," in *ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, 2007, pp. 177–184.