Threat detection in online discussions

Aksel Wester and **Lilja Øvrelid** and **Erik Velldal**
Department of Informatics
University of Oslo, Norway
{aksellw, liljao, erikve}@ifi.uio.no

Abstract

This paper investigates the effect of various types of linguistic features (lexical, syntactic and semantic) for training classifiers to detect threats of violence in a corpus of YouTube comments. Our results show that combinations of lexical features outperform the use of more complex syntactic and semantic features for this task.

1 Introduction

Threats of violence constitute an increasingly common occurrence in online discussions. It disproportionately affects women and minorities, often to the point of effectively eliminating them from taking part in discussions online. Moderators of social networks operate on such a large scale that manually reading all posts is an insurmountable task. Methods for automatically detecting threats could therefore potentially be very helpful, both to moderators of social networks and to their members.

In this article, we evaluate different types of features for the task of detecting threats of violence in YouTube comments. We draw on both lexical, morphosyntactic and lexical semantic information sources and experiment with different machine learning algorithms. Our results indicate that successful detection of threats of violence is largely determined by lexical information.

2 Related work

There is little previous work specifically devoted to the detection of threats of violence in text. However, there is previous work which examines other types of closely related phenomena, such as 'cyberbullying' and hate-speech.

Dinakar et al. (2011) propose a method for the detection of cyberbullying by targeting combinations of profane or negative words, and words related to several predetermined sensitive topics. Their data set consists of over 50,000 YouTube comments taken from videos about controversial topics. The experiments reported accuracies from 0.63 to 0.80, but did not report precision or recall.

There has been quite a bit of work focused on the detection of threats in a data set of Dutch tweets (Oostdijk and van Halteren, 2013a; Oostdijk and van Halteren, 2013b), which consists of a collection of 5000 threatening tweets. In addition, a large number of random tweets were collected for development and testing. The system relies on manually constructed recognition patterns in the form of n-grams, but details about the strategy used to construct these patterns are not given. In Oostdijk and van Halteren (2013b), a manually crafted shallow parser is added to the system. This improves results to a precision of 0.39 and a recall of 0.59.

Warner and Hirschberg (2012) present a method for detecting hate speech in user-generated web text, which relies on machine learning in combination with template-based features. The task is approached as a word-sense disambiguation task, since the same words can be used in both hateful and non-hateful contexts. The features used in the classification were combinations of uni-, bi- and trigrams, part-of-speech-tags and Brown clusters. The best results were obtained using only unigram features, with a precision of 0.67 and a recall of 0.60. The other feature sets garnered much lower results, and the authors suggest that deeper parsing could reveal significant phrase patterns.

Most closely related to the current paper is the work of Hammer (2014), reporting on experiments on (a previous version of) the same corpus as used here. The method uses a logistic LASSO regression analysis on bigrams (skip-grams) of important words to classify sentences as either threats of violence or not. The system makes use of a list of words that are correlated with threats of violence. The article does not, however, describe exactly how these important words were selected, stating only that words were chosen that were significantly correlated with the response (violent/non-violent sentence). Results are reported only in terms of proportion of false positives for the two classes, and it is not clear how the data was split for training and evaluation, making it difficult to directly compare our results to those of Hammer (2014).

3 The YouTube threat corpus

The YouTube threat corpus[1] comprises user-written comments from eight different YouTube videos (Hammer, 2014). The videos cover religious and political topics like halal slaughter, immigration, Anders Behring Breivik, Jihad, etc. A given comment consists of a set of sentences, each of them manually annotated to be either a threat of violence (or support for a threat of violence), or not. The corpus also records usernames of the commenters.

As shown in Table 1, the corpus consists of 9,845 comments, comprising 28,643 sentences. There are 1,384 sentences containing threats, spread over 1,285 comments. Hammer (2014) reports inter-annotator agreement on this data set to be 98 %, as calculated on 120 of the comments, doubly annotated for evaluation.

Figure 1 provides some examples of comments from the corpus. The first line is the anonymized username, and the subsequent lines are the sentences of the comment. The sentences are annotated with a number indicating whether they contain a threat of violence (1) or not (0).

	Comments	Sentences	Users posting
Total	9,845	28,643	5,483
Threats	1,285	1,384	992

Table 1: Number of comments, sentences and users in the YouTube threat data set.

```
User #44
1    and i will kill every fucking
     muslim and arab!

User #88
0    Need a solution?
1    Drop one good ol' nuke on that
     black toilet in Mecca.
```

Figure 1: Example comments from the YouTube threat corpus.

4 Experiments

Much of the previous related work presented in Section 2 made use of pre-compiled lists of correlated words and manually crafted patterns. Whereas these resources can be effective, they are highly task-specific and do not easily lend themselves to replication. Furthermore, while much of the previous work seem to highlight the effectiveness of lexical features, several of the authors also suggest that parsing may be beneficial for these tasks.

The approach followed in the current paper is to train a machine-learned model to automatically detect threats. We experiment with a range of different sources of linguistic information for defining our feature templates. We also generalize these features through a 'backoff' technique. Throughout, we make use of resources and tools that are freely available and reusable.

4.1 Experimental setup

Pre-processing Since threat annotation is performed on the sentence level, the corpus has been manually split into sentences as part of the annotation process. We performed tokenization, lemmatization, POS-tagging and dependency parsing using the spaCy NLP toolkit. SpaCy assigns both the standard Penn Treebank POS-tags (Marcus et al., 1993), as well as the more coarse-grained Universal POS tag set of Petrov et al. (2012). The dependency parser assigns an analysis in compliance with the ClearNLP converter (Choi and Palmer, 2012), see

[1] Please contact the authors if you want to obtain access to the corpus for your own research.

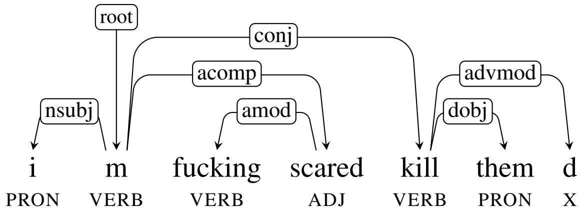

Figure 2: Dependency parse of example sentence from the corpus, with assigned uPOS tags.

Figure 2 for an example dependency graph from the corpus. The corpus was further enriched with the cluster labels described in Turian et al. (2010), created using the Brown clustering algorithm (Brown et al., 1992) and induced from the Reuters corpus of English newswire text (RCV1). We vary the number of clusters to be either 100, 320, 1000 or 3200 clusters and use the full cluster label. We also make use of the WordNet resource (Fellbaum, 1998) to include information about the synset of a word, as well as its parent and grandparent synsets.

Classifiers We test three different classification frameworks in our development testing: Maximum Entropy (MaxEnt), Support Vector Machines (SVM), and Random Forests (RF). We approach the task as a binary classification task, using the implementations found in the scikit-learn toolkit (Pedregosa et al., 2011).

Tuning When tuning each model, we aim to maximize the F-score. For the MaxEnt and SVM classifiers we tune the regularization parameter (C), where a smaller value corresponds to stronger regularization. When tuning these classifiers, we start with C-values from 1 to 150 in 10-value increments, select the best performing C-value and repeat the process with decreasing increments, with the range of C-values centered on the best performing C-value thus far. After 6 iterations, we terminate the tuning, and select the best performing C-value. When tuning the Random Forest classifier, we perform a grid search over the maximum number of features used when splitting a node in the tree (sqrt(n), log(n) or n, where n is the number of features in the model), and the number of trees (10, 100 or 1000, depending on the size of the feature set). In the following experiments, we perform tuning on all feature sets.

Features Based on the enriched corpus, as described above, we experiment with the following sources of information for defining our features:

- Lexical:
  - Word form
  - Lemma
- Morphosyntactic:
  - Penn Treebank (PTB) POS
  - Universal POS (uPOS)
  - Dependency Relation
- Semantic:
  - Brown cluster label
  - WordNet synset, + parent and grandparent

The features are structured according to a set of *feature templates*, which record varying degrees of linear order and syntactic context: *bag-of* features (unordered), bigrams, trigrams and dependency triples. Examples of the latter, given the sentence in Figure 2, would be: {<m, nsubj, i>, <root, root, m>, <scared, amod, fucking>, ...}

Our lexicalized features are very specific and require the exact combination of two lexical items in order to apply to a new instance. Following Joshi and Penstein-Rose (2009), we therefore experiment with generalizing features by 'backing off' to a more general category, e.g., from word form to lemma or POS. For example, a dependency triple over word forms like <kill, dobj, them> would thus be generalized to <VERB, dobj, them> using *head-backoff*, and <kill, dobj, PRON> using *modifier-backoff*. These additional backoff features are included for bigrams and trigrams as well as dependency triples.

We impose a simple count-based reduction of the feature set; only features appearing at least twice in the training data are included in the model.

4.2 Development results

We start by defining an informed baseline system, empirically selecting an initial set of features and a classification framework to use as a basis and reference point for further development. The features for this initial round of tuning comprise basic lexical features; word forms and lemmas, in addition to n-grams defined over these.

In the second round of experiments we test combinations of these basic lexical feature types, be-

	MaxEnt	SVM	RF
Bag-of-words	**0.6123**	**0.6068**	**0.5918**
Bag-of-lemmas	0.5902	0.5982	0.5856
Bigrams of word forms	0.4856	0.4887	0.4944
Trigrams of word forms	0.2776	0.2856	0.2859

Table 2: Results for baseline system; F-score for bag-of lexical features (word form and lemma), and bigram and trigram templates over word forms. From left to right the columns correspond to Maximum Entropy classifiers, Support Vector Machines, and Random Forests.

n-gram combination	BoW	BoW+BoL
no n-grams	0.6123	0.6278
+bigrams	0.6376	0.6577
+trigrams	0.6180	0.6453
+bi- and trigrams	0.6337	**0.6656**

Table 3: F-scores of the bag-of-words feature set, with different combinations of the other feature sets tested in Table 2, namely bag-of-lemmas and n-grams of word forms.

fore moving on to introduce more linguistic features, both syntactic and semantic, as *bag-of* features, and as backoff from word form n-grams. Finally, we will evaluate the inclusion of dependency triples.

Table 2 shows initial development results in terms of F-score for the the three different classifiers across four different feature sets; bags of word-forms and lemmas, as well as bigrams and trigrams over word forms. Generally, we see that feature sets containing lexical *bag-of* features outperform the n-gram features. The overall best result came from the MaxEnt classifier with the bag-of-word form feature set. This model yielded an F-score of 0.6123, with a precision of 0.6777 and a recall of 0.5585, using the MaxEnt classifier after tuning. This is the feature set we will use as our basic reference model in the next stage of our experiments is the bag-of-word forms. We will also only be using the the MaxEnt classifier for the remainder for reported development experiments. Not only did the MaxEnt model have the best performance for the basic BoW feature sets, but it is also the framework that is most convenient to use for exploring many combinations of features given that a MaxEnt model can be trained in much less time than an SVM or Random Forest classifier.

We go on to test various combinations of these word form n-grams and the lexical *bag-of* features.

	POS_{Lex}	Dep_{Lex}	Synset	Brown
BoF	0.6018	0.5655	0.4922	0.4688
BoW+BoF	0.6071	0.6185	0.6176	0.6145

Table 4: F-scores of *bag-of* features (BoF) with and without bag-of-words (BoW). POS_{Lex} and Dep_{Lex} are lexicalized POS and dep-tags, respectively, where each feature is a tuple consisting of the tag and the word form of the token.

	bigram	trigram	bi+trigram
Lemma backoff	0.6611	0.6480	**0.6649**
POS backoff	0.6410	0.6294	0.6320
Dep backoff	0.6208	0.6194	0.6220
Synset backoff	0.6454	0.6448	0.6537
Brown backoff	0.6285	0.6173	0.6335

Table 5: Backoff from different combinations of n-grams. The models also contain bag-of-words and the other *bag-of* features. Each backoff combination is the one that achieved the best result.

As seen in Table 3, we test bag-of-words alone, and bag-of-words (BoW) with bag-of-lemmas (BoL), combined with the word form variants of bigrams, trigrams and both. The best result without BoL comes from BoW with only bigrams, with an F-score of 0.6376, closely followed by BoW with both types of n-grams, which got an F-score of 0.6337. However, the best result overall came from the combination of BoW, BoL, and both types of n-grams, which got an F-score of 0.6656.

Next, we include feature types based on the other information sources listed above, i.e., POS, dependencies, WordNet synsets, and Brown clusters. All these features are instantiated both with the *bag-of* feature template on their own, and in combination with the bag-of-words features. As seen in Table 4, none of the feature types alone yield higher F-scores than bag-of-words. On the other hand, when combined with BoW, all feature types (except the lexicalized POS-tags) perform better than BoW alone. However, none of them outperform BoW and BoL combined (F=0.6278, cf. Table 3), or the combination of Bow+BoL with n-grams (F=0.6656).

Next, the n-grams and *bag-of* features are combined with generalized versions of n-gram features using the 'backoff' strategy described in the previous section. In Table 5 we see that the lemma backoff consistently outperform the other feature types,

Dependency backoff	BoW+dep	All
w/o backoff	0.6240	**0.6586**
Lemma	0.6224	0.6507
POS	**0.6298**	0.6547
Synset	0.6234	0.6516
Brown	0.6299	0.6504

Table 6: Dependency triples with and without feature backoff, in combination with other features: 'BoW+dep' is the feature set containing bag-of-words and dependency triples. 'All' is the feature set containing BoW, BoL, bi- and trigrams and dependency triples. Each row backs off to a different feature type.

but that even lemma backoff does not improve upon the results without backoff features. We test all possible backoff combinations.

Lastly, we will test the addition of dependency triples to our models. We add dependency triples consisting of word forms, both alone, and in conjunction with feature backoffs. As with n-gram backoff, we test multiple variations of backoff, both head-backoff and modifier-backoff. The results reported are the backoff variant that achieved the best results. As seen in Table 6, the inclusion of word form dependency triples improved upon the simplest model, with an F-score of 0.6240, compared to 0.6123 for bag-of-words alone. Dependency triples did not, however, improve upon the results achieved by our previous best performer with an F-score of 0.6656. Neither did the addition of backoff improve upon this. Adding POS backoff to the 'BoW+dep' configuration resulted in a slightly increased F-score of 0.6298. However, adding backoff to the 'All' feature set only resulted in poorer F-scores, with the best feature backoff (also POS) resulting in an F-score of 0.6547.

For the development data then, our best performer remains the feature set comprising bag-of-word forms, bag-of-lemmas, and word form bigrams and trigrams; with an F-score of 0.6656. We will refer to this model as 'lexical n-grams'.

The lexical n-gram feature set achieved a precision (P) of 0.7709 and a recall (R) of 0.5857. Compared with our initial BoW system, which had P=0.6777 and R=0.5585, we see that the majority of the improvement comes from the increase in precision. When reviewing a random sub-sample of the false positives and false negatives, we see that the noisy data has caused some problems for the preprocessor. Another source of errors, specifically for precision, are comments which use multiple typically threatening words in a non-threatening context.

	Precision	Recall	F-score
BoW	0.7325	0.5943	0.6562
Lexical n-grams	**0.7532**	**0.6299**	**0.6860**

Table 7: Precision, recall and F-score on the held-out test set for the basic BoW model and the best development feature set; the lexical n-gram model.

4.3 Held-out results

We performed our final testing on the held-out test set using the basic BoW model and the lexical n-gram model. As seen in Table 7, the n-gram model outperforms the BoW model by a good margin, with F-scores of 0.6860 and 0.6562 respectively. However, the difference in F-scores between the two feature sets is not as large as on the development data. At the same time, we see that both models actually achieves higher scores on the held-out data than the development data, and this effect is particularly strong for the BoW model (with the F-score going up from 0.6123 to 0.6562).

5 Conclusion

This paper has developed and compared several machine-learned models for automatically detecting threats of violence in online discussions. The data set comprises a manually annotated corpus of comments from YouTube videos. We have reported experimental results for different classification frameworks and a wide range of different linguistic features. The best performance was observed for combinations of simple lexical features (bag-of-words- and lemmas, in combination with bi- and trigrams). Introducing more complex features – drawing on information from WordNet synsets, Brown clusters, POS tags and dependency parses – did not improve on the simpler surface-based feature set.

References

P.F. Brown, P.V. deSouza, R.L. Mercer, V.J.D. Pietra, and J.C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18.

Jinho D. Choi and Martha Palmer. 2012. Guidelines for the clear style constituent to dependency conversion. Technical Report 01-12, Institute of Cognitive Science, University of Colorado Boulder.

Karthik Dinakar, Roi Reichart, and Henry Lieberman. 2011. Modeling the detection of textual cyberbullying. In *Proceedings of The Social Mobile Web*.

Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press, Cambridge, MA.

Hugo Lewi Hammer. 2014. Detecting threats of violence in online discussion using bigrams of important words. In *Proceedings of Intelligence and Security Informatics Conference (JISIC)*, pages 319–319.

Mahesh Joshi and Carolyn Penstein-Rose. 2009. Generalizing dependency features for opinion mining. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, page 313–316.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpora of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

Nelleke Oostdijk and Hans van Halteren. 2013a. N-gram-based recognition of threatening tweets. In *Proceedings of Computational Linguistics and Intelligent Text Processing*, pages 183–196. Springer.

Nelleke Oostdijk and Hans van Halteren. 2013b. Shallow parsing for recognizing threats in Dutch tweets. In *Proceedings of IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, Niagara, Canada. ACM.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 2089–2096.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Meeting of the Association for Computational Linguistics*.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26. Association for Computational Linguistics.