Features for detecting threats of violence in YouTube comments

Anonymous NAACL submission

Abstract

This article investigates the effect of various types of linguistic features (lexical, morphosyntactic and lexical semantic) for the task of detecting threats of violence in a data set of YouTube comments. It further examines the use of these features in combination with different representations of linguistic context, and feature backoff. 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 disproportionally 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 the members of those networks.

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 Previous 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 collected by a website. 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 do not go into detail about the methods used to construct these patterns. 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 from the internet, 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 of the classifications were obtained using only unigrams as 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.

Perhaps closest to the current work is the work of Hammer (2014), which reports on an experiment on the same data set that we will be using in our own experiments. 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. This makes it difficult to compare our results to those of Hammer (2014).

3 The YouTube threat data set

The YouTube threat data set is comprised of user-written comments from eight different YouTube videos (Hammer, 2014). A 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 data set furthermore records the username of the user that posted the comment. The eight videos that the comments were posted to cover religious and political topics like halal slaughter, immigration, Anders Behring Breivik, Jihad, etc. (Hammer, 2014).

The data set consists of 9,845 comments, comprised of 28,643 sentences, see Table 1. There are 1,285 comments containing threats, and 1,384 sentences containing threats, as seen in Table 1. 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

	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: Examples of comments from the data set.

containing threats of violence taken from the data set. The first line is the annonymized 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).

4 Experiments

Much of the previous work presented in Section 2 made use of precompiled 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. The previous work further highlights the effectiveness of lexical features (word-based features), however, several of the authors suggest that parsing may be beneficial for these tasks.

In this work we experiment with a wide range of linguistic features in our system and investigate the introduction of linguistic context through a set of 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 in order to construct our system.

4.1 Experimental setup

Pre-processing Since threat annotation is performed on the sentence level, the data set has been manually split into sentences as part of the annotation process. We performed tokenization, lemmati-

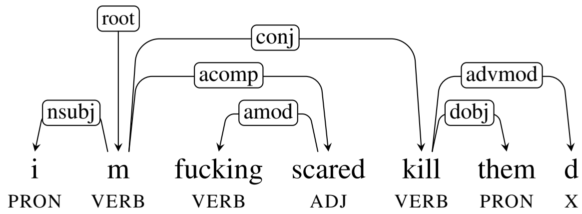

Figure 2: Dependency parse of example sentence from the data set, with assigned uPOS tags.

zation, PoS-tagging and dependency parsing using the SpaCy toolkit[1]. 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 Figure 2 for an example dependency graph from the data set. The data set 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 RCV1 corpus, a corpus containing Reuters English newswire text. 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 determine a word's synset, parent and grandparent synset.

Classifiers We test three different classifiers in our development testing; a Maximum Entropy (MaxEnt) classifier, a Support Vector Machine (SVM), and a RandomForest classifier (RF). We approach the task as a binary classification task, and we use scikit-learn (Pedregosa et al., 2011) for our classification.

Tuning When tuning each model, we aim to maximize the F-score of the model. In the MaxEnt classifier and SVM we tune the C-value, which for both classifiers is the inverse of regularization strength. 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 ever-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 RandomForest 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.

[1] https://spacy.io/

Features Based on the enriched data set, as described above, we experiment with the following feature types:

- Lexical:
  - Word form
  - Lemma
- Morphosyntactic:
  - Penn Treebank (PTB) POS
  - Universal (uPOS) PoS
  - Dependency Relation
- Semantic:
  - Brown cluster label (100, 320, 1000 and 3200 clusters)
  - WordNet synset, parent synset and grandparent synset

These features are structured according to a set of *feature templates*, which introduce varying degrees of linear order and syntactic context: bag-of-features (unordered), bigrams, trigrams and dependency triples. These feature templates are applied to the feature types presented above, in order to yield features like the following for our examples sentence in Figure 2:

- Bigrams of word forms: {i m, m fucking, fucking scared, ...}
- Depdendency triples: {<m, nsubj, i>, <root, root, m>, <scared, amod, fucking>, ... }

Our lexical 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 generalization of features by backoff to a more general category, e.g. from word form to lemma or PoS. 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*. We apply backoff to bigram, trigram and dependency triple features.

	MaxEnt	SVM	RF
Bag of word forms	**0.6123**	**0.6068**	**0.5918**
Bag of lemmas	0.5902	0.5982	0.5856
Bigrams	0.4856	0.4887	0.4944
Trigrams	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 . MaxEnt is the MaximumEntropy classifier, SVM is the support vector machine, and RF is the RandomForest classifier.

4.2 Development results

During development, we first select a baseline system, consisting of a feature set of lexical features (word forms and lemmas) and a classifier, that we will compare the other results to. Next, we will test different combinations of the lexical feature sets used in the baseline testing. We will then introduce more linguistic features, both morphosyntactic and lexical semantic, as bags of features, and as backoff from word form n-grams. Finally, we will evaluate the inclusion of dependency triples, using both word form triples, and dependency triples with feature backoff.

Table 2 shows baseline results in terms of F-score for bag-of lexical features (word form and lemma), and bigram and trigram templates over word form, for the three different classifiers. The overall best result came from the MaxEnt classifier with the bag of word form feature set. Generally, we see that feature sets containing bag of lexical features outperform the n-gram feature sets. The feature set we will use as a baseline in the next stage of our experiments is bag of word forms. The feature set recieved an F-score of 0.6123, with a precision of 0.6777 and a recall of 0.5585, using the MaxEnt classifier after tuning. We will also be using only the MaxEnt classifier for the remainder of the feature set experiments.

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

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, and bag of words and lemma feature sets, combined with word form bigrams, trigrams or both.

	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: Results of bag of feature variations with different feature types. The results shown are the F-scores of each feature set after tuning, using the MaxEnt classifier.

Next, we add the different feature types. We use the bag-of feature template, and for each feature, test both the bag of features on its own, and with bag of words. As seen in Table 4, none of the feature types alone result in higher F-scores than bag of words. However, all feature types (except the lexicalized POS-tags), combined with bag of words, perform better than bag of words alone. Though none of them beat bag of words and lemmas combined, with an F-score of 0.6278.

Next, we combine the results of bag of features, and n-grams, with n-gram feature backoffs. In Ta-

	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 bag of features. Each backoff combination is the one that achieved the best result.

Dependency backoff	Simple	Combined
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. Simple is the feature set containing bag of words and dependency triples. Combined is the feature set containing bag of words, bag of lemmas, bigrams, trigrams and dependency triples. Each row backs off to a different feature type.

ble 5 we see that the lemma backoff consistently outperform the other feature types, 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 acheived 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 best model, as the best model with dependency triples got an F-score of 0.6586 compared to our best result of 0.6656.

The addition of backoff also did not improve upon our best model. Adding POS backoff to our simplest model resulted in a slightly increased F-score of 0.6298. Adding backoff to our combined model, however, only resulted in poorer F-scores, with the best feature backoff (also POS) resulting in an F-score of 0.6547.

Our final model (which we refer to as “Combined”) is the model consisting of bag of word forms, bag of lemmas, and word form bigrams and trigrams, which got the best result, with an F-score of 0.6656. In Subsection 4.4 we will also test that same model with the addition of trigram backoff to lemmas, which got an F-score of 0.6649.

	Precision	Recall	F-score
Baseline	0.7325	0.5943	0.6562
Combined	0.7532	**0.6299**	**0.6860**
Combined+backoff	**0.7703**	0.6085	0.6799

Table 7: Precision, recall and F-score for the baseline feature set, the best feature set, and the best backoff feature set, when classification is performed on the held-out test set.

4.3 Error analysis

The combined feature set acheived a precision of 0.7709, and a recall of 0.5857. Compared with our baseline system, which had a precision of 0.6777, and a recall of 0.5585, we see that the majority of the improvement over the baseline comes from the increase in precision. When reviewing a random subsample 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.

4.4 Held-out results

We performed our final testing on the held-out test set using the baseline feature set, the combined feature set, and the combined features set with trigram backoff using lemma features. As seen in Table 7, both the best feature set, and the second best, from our development testing, outperformed the baseline feature set. The best feature set from development also outperformed the second best. The difference in F-scores between the baseline and the best feature set was not as large as in our development testing. However, this may be due to the fact that the baseline performed a lot better in the held-out testing compared to the development testing, with an F-score of 0.6562. The F-scores of the combined feature sets were 0.6860 and 0.6799. In fact, both the combined feature sets improved upon the baseline in both precision, recall and F-scores.

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 313316.

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.