AAAI Technical Report SS-12-06
Wisdom of the Crowd

Using Crowdsourcing to Improve Profanity Detection

Sara Owsley Sood
Computer Science Department
Pomona College
185 East Sixth Street
Claremont, CA 91711
sara@cs.pomona.edu

Judd Antin, Elizabeth F Churchill
Yahoo! Research
4301 Great America Parkway
Santa Clara, CA 95054
<jantin, echu>@yahoo-inc.com

Abstract

Profanity detection is often thought to be an easy task. However, past work has shown that current, list-based systems are performing poorly. They fail to adapt to evolving profane slang, identify profane terms that have been disguised or only partially censored (e.g., @ss, f$#%) or intentionally or unintentionally misspelled (e.g., biatch, shiiiit). For these reasons, they are easy to circumvent and have very poor recall. Secondly, they are a one-size fits all solution – making assumptions that the definition, use and perceptions of profane or inappropriate holds across all contexts. In this article, we present work that attempts to move beyond list-based profanity detection systems by identifying the context in which profanity occurs. The proposed system uses a set of comments from a social news site labeled by Amazon Mechanical Turk workers for the presence of profanity. This system far surpasses the performance of list-based profanity detection techniques. The use of crowdsourcing in this task suggests an opportunity to build profanity detection systems tailored to sites and communities.

Introduction

As user participation on the web grows, the presence of inappropriate content from user contributions becomes more problematic. Social news sites, forums, and any online community must manage user-generated content, censoring that which is not inline with the social norms and expectations of a community. Failure to remove such content not only deters potential users/visitors, but can also signal that such content is acceptable (Sukumaran et al. 2011).

Much of the focus in the detection of inappropriate content has focused on visual content—especially potentially sexually explicit or pornographic content. Research in computer vision has made great advances toward detecting images with 'too much skin' (Fleck, Forsyth, and Bregler 1996; Wang et al. 1998). With the rise in cyberbullying (Boyd and Marwick 2011; Kontostathis, Edwards, and Leatherman), some recent work in natural language processing has focused on detecting insults and harassment online with reasonable accuracy (Dinakar, Reichart, and Lieberman 2011; Sood, Churchill, and Antin 2011). However, a seemingly simpler problem has remained unaddressed, with little advances in recent years – profanity detection.

From our investigation, current profanity detection systems are performing poorly (Sood, Antin, and Churchill 2011). The current state of the art are list-based systems which simply identify terms that fall on a listing of known profane terms; in practice, lists are shared among forum hosts for this task. These systems fail to adapt to evolving profane slang, identify profane terms that have been disguised or only partially censored (e.g., @ss, f$#%) or intentionally or unintentionally misspelled (e.g., biatch, shiiiit). For these reasons, they are easy to circumvent and have very poor recall. Secondly, they are a one-size fits all solution – making assumptions that the definition, use and perceptions of profane or inappropriate holds across all contexts. These shortcomings result in profanity systems performing far worse than they are expected or thought to.

Beyond the list-based profanity detection systems that are in place, most online communities rely on community managers to scour the comments posted daily and remove or intervene on those that are inappropriate. While this is feasible for some small communities, the vastness of others makes it unmanageable, resulting in a situation in which many communities remain 'unmanaged.' Social moderation systems in which users can 'flag as inappropriate' have great potential, but are very prone to collusion (Lou, Chen, and Lei 2009).

While many are suspect of the power of social moderation, we rely on another distributed solution to this problem – crowdsourcing. Rather than utilizing static lists of profane terms, we propose a system that instead learns the context in which profanity occurs. This approach attempts to overcome the issue of disguised profanity, as well as slang words and misspellings that are not accounted for in current approaches. We utilize a corpus of comments from a social news site for training and testing. Amazon Mechanical Turk workers labeled these comments for the presence of profanity. We compare this system's performance to that of list-based profanity detection approaches to address the question: could a system trained on comment streams and labeled for profanity by crowdsourcing outperform existing list-based profanity detection systems?

Dataset

For training and testing of our profanity detection systems, we employ a large dataset of comments from a medium-sized social news site. As with most social news sites, users of this site contribute links to news items (i.e., stories, videos and images), and other community members then comment on these items. The dataset contains all comments posted between March and May 2010. The set of 1,655,131 comments span 168,973 threads. Each thread represents a news item; that is, a story, image or video contributed by a community member.

While this dataset contains some metadata (e.g., category/domain of the news items, time posted, whether users 'rated up' or 'rated down' the news items and comments), more metadata is needed. Most importantly, we need to know whether each comment contains profanity. Rather than label the comments ourselves, or hire expert coders who can be quite costly, we chose to use crowdsourcing as an efficient and economical method by which to label our corpus (Tetreault, Filatova, and Chodorow 2010; Mason and Watts 2009). As noted by others, crowdsourcing has the added benefit of gathering a diverse set of non-expert opinions, that, even while often disagreeing, can produce high quality results (Callison-Burch 2009; Sheng, Provost, and Ipeirotis 2008).

We chose to use Amazon Mechanical Turk for this task. Prior to using MTurk, the authors conducted a small pilot study to hone in on the language to be used in questions/instructions for this task, to ensure usable results. Understanding that the concept of 'profane' has community specific definitions and interpretations, the purpose of this pilot task was to clarify instructions, but more importantly, to see if the authors find any agreement on their own definitions of profanity, thus making this a worthwhile use of MTurk. After iterating through multiple revisions of the questions/instructions, the authors landed on the following question (among others) to be asked of each MTurk worker when analyzing a single comment:

> *"Does this message contain any content you would describe as 'profanity?' (including profanity that is disguised or modified such as @ss, s***, and biatch) (Yes/No)"*

Since training and testing of these systems does not require such a large dataset of labeled comments (1,655,131 comments), we randomly selected 6500 comments from the dataset to be labeled by MTurk workers. After a mere 5 days, 221 Turkers had provided labels for different subsections of our dataset (25,965 judgments in total). We required a minimum of 3 judgments per comment and for our systems, we chose to only use comments for which a consensus of 66% was reached. This agreement threshold eliminated 2.2% of the corpus, leaving us with 6354 comments labeled for the presence of profanity. Of those 6354 comments, 9.4% were labeled as positive (containing profanity).

In addition to requiring a 66% consensus on labels, MTurk workers were also challenged with 'gold' questions (questions with clear predetermined correct answers) that tested the validity of each worker. MTurk workers were barred from the labeling task, and all of their previous labels were removed, if they answered multiple 'gold' questions incorrectly. CrowdFlower[1] managed this validation of workers. More information about this dataset and past studies using it can be found in our past work (Sood, Antin, and Churchill 2011; Sood, Churchill, and Antin 2011).

Methods

In the sections that follow, we outline techniques used within our profanity detection systems. These techniques are combined in various ways in the evaluation section in an attempt to compare current list-based approaches to our suggested system that utilizes crowdsourcing and machine learning techniques.

List-based Approaches

The current state of the art in profanity detection takes a 'list-based' approach. That is, in order to determine if a document (e.g., email, comment, tweet, blog) contains profanity, these systems simply examine each word in the document. If any of the words are present on a list of profane terms, then the document is labeled as profane. Such lists of profane terms are typically shared among forum hosts; we downloaded a shared list from *phorum.com*.

Since list-based systems face issues of recall, a logical approach would be to find a more extensive listing of

[1] http://crowdflower.com/

terms. With this in mind, the authors downloaded a second list of terms from *noswearing.com*, a site that hosts a list of community contributed profane terms and their meanings. Since the community contributes the words, this list has the added benefit of holding slang profanity and words that have emerged over time.

To make a further effort to address poor recall, some list-based approaches might also include a stemmer (Porter 1980). Beyond just searching for the presence of terms on a list of profane terms, a system that uses a stemmer can also check if any words in the target document share stems with any known profane terms.

Levenshtein Edit Distance

While a more extensive list may help with Internet slang and evolving language, it will not identify variations in spellings and disguised profane terms. For example, one may write "shiiiit" to either emphasize the word, or to evade profanity detection systems. Similarly, users may substitute punctuation for letters to either partially censor a word, or avoid being detected. For example, a user may write "@ss" or "sh1t." List-based approaches will fail for these cases, as there are countless variations that could be used – too many to enumerate in a list.

After first checking to see if a word exists on a profanity list, we then make a second pass to look for variations of the words. To identify these instances of profanity, we utilize a tool to calculate the Levenshtein edit distance between two terms (Levenshtein 1966). This edit distance measures the number of letter insertions, deletions and changes to transform one word into another. To check whether or not a term might be profanity, we calculate the edit distance from that term to each term on a list of known profane terms (such as those described in the previous section). If the edit distance is equal to the number of punctuation marks in the term, or if it is below some threshold (which varies by the term length), then it is flagged as profane.

A pilot use of this tool found many false positives – English words with a small edit distance from profane terms (e.g. "shirt", "fitch") and first names (e.g. "mike", "ash"). To avoid these, we first verify that the target word does not appear in an English dictionary or a known list of male and female first names. If it does not appear in these three corpora, we then calculate the edit distances and judge accordingly. Finally, the pilot run also found some false negatives – when misspellings came in different tenses or forms (e.g. "@sses", "sh1ting"). To handle these we employ a simple stemmer (described above), calculating the edit distance from the stem of the target word to known profane words and their stems as well.

Support Vector Machines

Finally, the above two systems utilize lists of known profane terms and tools that attempt to find variations in these terms. However, as profane language evolves over time, these lists must be updated to catch new cases. A more robust approach may be to look at the context in which the profane language occurs. As such, we employ support vector machines, known for their performance in text classification (Joachims 1998), to learn a model of profanity. We utilize the 6500 profanity labeled comments from the dataset described above. Using a bag of words approach, we found the optimal features to be bigrams and stems using a binary presence representation and a linear kernel. The performance of the system trained on other feature sets is omitted for brevity.

Evaluation

We evaluate the performance of all systems via precision, recall, maximal f1 and maximal accuracy averaged over 5 trials of 10-fold cross validation of the profanity labeled comments from our dataset. These measures are typical for this type of work, and give a more complete picture than each alone (Joachims 1998; Yin et al. 2009). While our past work has shown that list-based systems do not suffice, here we aim to compare list-based approaches to the present suggested approach of utilizing crowdsourcing labeled data in this task. For comparison to the current state of the art in profanity detection, the last 9 rows in Table 1 contain evaluations of list-based profanity detection systems from our past study (Sood, Antin, and Churchill 2011).

The first column in Table 1, 'system,' describes the technique being evaluated. For example, the 'no swearing.com or phorum.com w/ stemming' system analyzes a

System	Precision	Recall	Maximal F1	Maximal Accuracy
SVM *or* no swearing.com *or* Levenshtein	0.62	0.64	0.63	0.93
SVM *or* no swearing.com *or* Levenshtein	0.65	0.60	0.62	0.93
SVM *or* phorum.com *or* no swearing.com	0.60	0.62	0.61	0.93
SVM *or* phorum.com	0.73	0.49	0.59	0.94
SVM	0.84	0.42	0.56	0.94
no swearing.com *or* Levenshtein	0.55	0.42	0.48	0.91
no swearing.com w/ stemming	0.53	0.40	0.46	0.91
no swearing.com *or* phorum.com w/ stemming	0.49	0.41	0.45	0.90
no swearing.com	0.56	0.37	0.44	0.91
no swearing.com *or* phorum.com	0.52	0.39	0.44	0.91
phorum.com w/ stemming	0.63	0.23	0.34	0.91
phorum.com	0.64	0.20	0.30	0.91
random	0.10	0.50	0.16	0.50
weighted random	0.11	0.11	0.11	0.83

Table 1: Precision, Recall, Maximal F1 and Maximal Accuracy evaluations for a set of systems faced with the task of detecting the presence of profanity in a comment. The bottom 9 rows are findings from previous work, included for comparison to the current approach (Sood, Antin, and Churchill 2011).

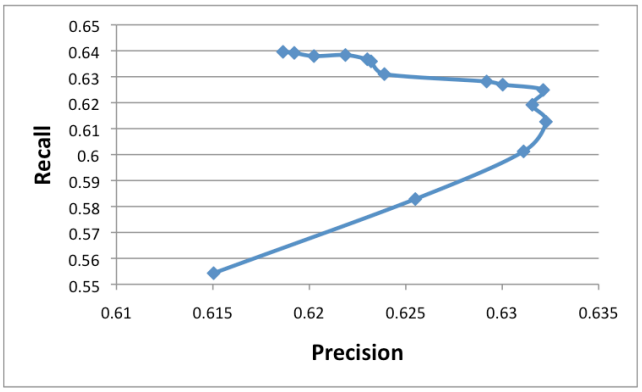

Figure 1: Precision - Recall curve for the "SVM *or* no swearing.com *or* Leven." system (the top performing of Table 1).

document by checking if any words are present on either list (no swearing.com or phorum.com) or share any commons stems with words on those lists. The random and weighted random systems are included as baseline comparisons. The random system randomly labels each comment as profane or not, while the weighted random system also takes into account the prior distribution of profane comments in the training corpus.

Table 1, in descending order by maximal f1, shows the systems of interest in the first six rows; the use of Levenshtein edit distance and linear kernel support vector machine with bigrams and stems as binary presence features. For brevity, we exclude an evaluation of systems trained on other features, feature representations and kernels; past work on this dataset has found presence of bigrams and stems as the most effective features set and representations (Sood, Churchill, and Antin 2011).

Noting that an SVM on its own (line 5 of Table 1) has high precision (0.84), but relatively low recall, we tested its performance when '*or-ed*' with profanity list-based systems and the Levenshtein edit distance tool. As seen in Table 1, we found optimal performance from a system that tests whether or not the support vector machine classifies the comment as profane *or* the noswearing.com list *or* the Levenshtein edit distance tool does so (line 1 of Table 1). If any of these three systems mark a comment as profane, then this combination system will mark the comment as profane. The combination of these systems with *or* increases recall, as expected, with a minimal loss of precision (see Figure 1 for a precision/recall curve of this system).

Finally, one might not value f-measure most in judging a profanity detection system, but rather precision or recall. We find high precision using a similar system to the one just described, but rather than combine the three base systems using an *or*, we do so using an *and*. That is, if the support vector machine flags a comment as profane *and* either the comment contains a word in the profanity list *or* the Levenshtein edit distance tool labels a comment as profane, then the system would flag the comment as profane. This system is highly precise as a profanity flag has to be verified by two separate systems (SVM and List/Levenshtein). These results are reported in Table 2, with a precision/recall curve in Figure 2. Unfortunately, given the nature of the problem, optimizing for recall is

harder, and we found that highest recall matches with the highest maximal f-measure – reported in Table 1.

System	P	R	F1	Acc.
SVM *and* (no swear. *or* Lev.)	0.90	0.20	0.32	0.92
SVM *and* no swear.	0.86	0.19	0.31	0.92

Table 2: Precision, Recall, Maximal F1 and Maximal Accuracy evaluations for two profanity detection systems optimized for high precision.

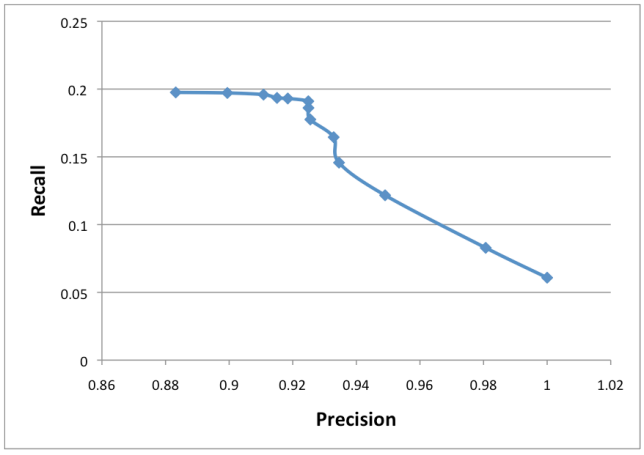

Figure 2: Precision - Recall curve for the "SVM and (no swearing.com or Leven.)" system (the top performing of Table 2).

Discussion

Using our current profanity labeled data set, along with list-based approaches, a Levenshtein edit distance tool and support vector machines, we have drastically improved performance in profanity detection. A list-based approach performed with a maximal F1 of 0.46 (noswearing.com w/ stemming) while our new system reaches a maximal F1 of 0.63. Our evaluation confirms our hypothesis that moving beyond a list-based approach to take into account the context in which profanity occurs, can greatly improve performance. While our system is an improvement in both precision and recall, recall sees the greatest performance increase, as expected, since we are no longer limited to the profane terms that appear in a profanity list.

In practice, a community manager might choose to use a system like this completely automatically, or as a way to filter down the set of comments they must 'eye-ball.' For the former, a high precision system is desired – avoiding false positives - so that one does not mistakenly flag and remove a comment that does not actually contain profanity. Toward this need, we have a system that reaches 90% precision at 20% recall (see Table 2). For the latter, a high recall system is desired, filtering the set of *all* comments down into those that *might* contain profanity, with the intention that a community manager look through that smaller set by hand. Unfortunately, given the nature of the problem, we are unable to optimize our systems for high recall. This remains a problem of interest to the research community.

Finally, this system would not be possible without the labels provided by crowdsourcing. The use of Amazon Mechanical Turk provided us not only with a quickly labeled dataset, but multiple judgments on each comment as a measure of confidence. Not only did MTurk allow us to train and test new systems for the social news site we studied, it shows great potential as a way to quickly tailor profanity detection systems to new sites/communities. By first extracting a set of content from a community, labeling that data via crowdsourcing, and employing support vector machines, one can quickly learn a model of profanity context and use specific to a site/community, enabling community-specific profanity detection systems. While a community manager is still requisite to understand social norms and tolerance for profanity within a community, they can be greatly aided by a system that detects the context in which profanity is typically used in their community.

References

Boyd, Danah, and Alice Marwick. 2011. "Why Cyberbullying Rhetoric Misses the Mark." *The New York Times*, September 22, sec. Opinion. http://www.nytimes.com/2011/09/23/opinion/why-cyberbullying-rhetoric-misses-the-mark.html.

Callison-Burch, Chris. 2009. Fast, cheap, and creative: evaluating translation quality using Amazon's Mechanical Turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*.

Dinakar, Karthik, Roi Reichart, and Henry Lieberman. 2011. Modeling the Detection of Textual Cyberbullying. In *Proceedings of International AAAI Conference on Weblogs and Social Media, Workshop "Social Mobile Web."*

Fleck, Margaret M, David A Forsyth, and Chris Bregler. 1996. "Finding Naked People." *European Conference on Computer Vision* II: 592 - 602.

Joachims, Thorsten. 1998. Text categorization with Support Vector Machines: Learning with many relevant features. In *Machine Learning: ECML-98*, ed. Claire Nédellec and Céline Rouveirol, 1398:137-142. Berlin/Heidelberg: Springer-Verlag. http://www.springerlink.com/content/drhq581108850171/.

Kontostathis, April, Lynne Edwards, and Amanda Leatherman. "Text Mining and Cybercrime": 149-164. doi:10.1002/9780470689646.ch8.

Levenshtein, V I. 1966. “Binary codes capable of correcting deletions, insertions, and reversals.” *Soviet Physics Doklady* 10 (8): 707-710.

Lou, J. K, K. T Chen, and C. L Lei. 2009. A collusion-resistant automation scheme for social moderation systems. In *IEEE Consumer Communications and Networking Conference, 2009.*, 571 -- 575.

Mason, Winter, and Duncan J Watts. 2009. Financial incentives and the “performance of crowds.” In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, 77–85. HCOMP ’09. Paris, France: ACM. doi:10.1145/1600150.1600175.

Porter, M.F. 1980. “An algorithm for suffix stripping.” *Program: electronic library and information systems* 14: 130-137. doi:10.1108/eb046814.

Sheng, Victor S., Foster Provost, and Panagiotis G. Ipeirotis. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’08*, 614. Las Vegas, Nevada, USA. doi:10.1145/1401890.1401965. http://dl.acm.org/citation.cfm?id=1401965.

Sood, Sara Owsley, Judd Antin, and Elizabeth Churchill. 2012. Profanity Use in Online Communities. In *proceedings of ACM SIGCHI.*

Sood, Sara Owsley, Elizabeth Churchill, and Judd Antin. 2011. “Automatic identification of personal insults on social news sites.” *In press - the Journal of the American Society for Information Science and Technology.*

Sukumaran, Abhay, Stephanie Vezich, Melanie McHugh, and Clifford Nass. 2011. Normative influences on thoughtful online participation. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, 3401–3410. CHI ’11. New York, NY, USA: ACM. doi:10.1145/1978942.1979450.

Tetreault, Joel R, Elena Filatova, and Martin Chodorow. 2010. Rethinking Grammatical Error Annotation and Evaluation with the Amazon Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, 45 - 48. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.170.7443.

Wang, James Z, Jia Li, Gio Wiederhold, and Oscar Firschein. 1998. “System for Screening Objectionable Images.” *COMPUTER COMMUNICATIONS JOURNAL* 21: 1355--1360.

Yin, Dawei, Zhenzhen Xue, Liangjie Hong, Brian Davison, April Kontostathis, and Lynne Edwards. 2009. Detection of Harassment on Web 2.0. In *Proceedings of the Content Analysis in the WEB 2.0 (CAW2.0) Workshop at WWW2009*. Madrid, Spain, April.