Deeper Attention to Abusive User Content Moderation

John Pavlopoulos
Straintek, Athens, Greece
ip@straintek.com

Prodromos Malakasiotis
Straintek, Athens, Greece
mm@straintek.com

Ion Androutsopoulos
Athens University of Economics and Business, Greece
ion@aueb.gr

Abstract

Experimenting with a new dataset of 1.6M user comments from a news portal and an existing dataset of 115K Wikipedia talk page comments, we show that an RNN operating on word embeddings outpeforms the previous state of the art in moderation, which used logistic regression or an MLP classifier with character or word n-grams. We also compare against a CNN operating on word embeddings, and a word-list baseline. A novel, deep, classification-specific attention mechanism improves the performance of the RNN further, and can also highlight suspicious words for free, without including highlighted words in the training data. We consider both fully automatic and semi-automatic moderation.

1 Introduction

User comments play a central role in social media and online discussion fora. News portals and blogs often also allow their readers to comment to get feedback, engage their readers, and build customer loyalty.[1] User comments, however, and more generally user content can also be abusive (e.g., bullying, profanity, hate speech) (Cheng et al., 2015). Social media are under pressure to combat abusive content, but so far rely mostly on user reports and tools that detect frequent words and phrases of reported posts.[2] Wulczyn et al. (2017) estimated that only 17.9% of personal attacks in Wikipedia discussions were followed by moderator actions. News portals also suffer from abusive user comments, which damage their reputations and make them liable to fines, e.g., when hosting comments encouraging illegal actions. They often employ moderators, who are frequently overwhelmed, however, by the volume and abusiveness of comments.[3] Readers are disappointed when non-abusive comments do not appear quickly online because of moderation delays. Smaller news portals may be unable to employ moderators, and some are forced to shut down their comments sections entirely.

We examine how deep learning (Goodfellow et al., 2016; Goldberg, 2016, 2017) can be employed to moderate user comments. We experiment with a new dataset of approx. 1.6M manually moderated (accepted or rejected) user comments from a Greek sports news portal (called Gazzetta), which we make publicly available.[4] This is one of the largest publicly available datasets of moderated user comments. We also provide word embeddings pre-trained on 5.2M comments from the same portal. Furthermore, we experiment on the 'attacks' dataset of Wulczyn et al. (2017), approx. 115K English Wikipedia talk page comments labeled as containing personal attacks or not.

In a fully automatic scenario, there is no moderator and a system accepts or rejects comments. Although this scenario may be the only available one, e.g., when news portals cannot afford moderators, it is unrealistic to expect that fully automatic moderation will be perfect, because abusive comments may involve irony, sarcasm, harassment without profane phrases etc., which are particularly difficult for a machine to detect. When moderators are available, it is more realistic to develop semi-

[1] See, for example, http://niemanreports.org/articles/the-future-of-comments/.

[2] Consult, for example, https://www.facebook.com/help/131671940241729 and https://www.theguardian.com/technology/2017/feb/07/twitter-abuse-harassment-crackdown.

[3] See, e.g., https://www.wired.com/2017/04/zerochaos-google-ads-quality-raters and https://goo.gl/89M2bI.

[4] The portal is http://www.gazzetta.gr/. Instructions to download the dataset will become available at http://nlp.cs.aueb.gr/software.html.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1125–1135
Copenhagen, Denmark, September 7–11, 2017.

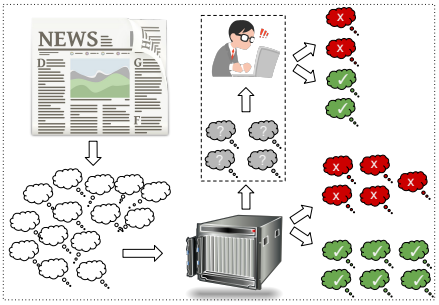

Figure 1: Semi-automatic moderation.

automatic systems aiming to assist, rather than replace the moderators, a scenario that has not been considered in previous work. In this case, comments for which the system is uncertain (Fig. 1) are shown to a moderator to decide; all other comments are accepted or rejected by the system. We discuss how moderation systems can be tuned, depending on the availability and workload of the moderators. We also introduce additional evaluation measures for the semi-automatic scenario.

On both datasets (Gazzetta and Wikipedia comments) and for both scenarios (automatic, semi-automatic), we show that a recurrent neural network (RNN) outperforms the system of Wulczyn et al. (2017), the previous state of the art for comment moderation, which employed logistic regression or a multi-layer Perceptron (MLP), and represented each comment as a bag of (character or word) n-grams. We also propose an attention mechanism that improves the overall performance of the RNN. Our attention mechanism differs from most previous ones (Bahdanau et al., 2015; Luong et al., 2015) in that it is used in a classification setting, where there is no previously generated output subsequence to drive the attention, unlike sequence-to-sequence models (Sutskever et al., 2014). In that sense, our attention is similar to that of of Yang et al. (2016), but our attention mechanism is a deeper MLP and it is only applied to words, whereas Yang et al. also have a second attention mechanism that assigns attention scores to entire sentences. In effect, our attention detects the words of a comment that affect most the classification decision (accept, reject), by examining them in the context of the particular comment.

Although our attention mechanism does not always improve the performance of the RNN, it has the additional advantage of allowing the RNN to highlight suspicious words that a moderator could consider to decide more quickly if a comment should be accepted or rejected. The highlighting comes for free, i.e., the training data do not contain highlighted words. We also show that words highlighted by the attention mechanism correlate well with words that moderators would highlight.

Dataset/Split	Accepted	Rejected	Total
G-TRAIN-L	960,378 (66%)	489,222 (34%)	1.45M
G-TRAIN-S	67,828 (68%)	32,172 (32%)	100,000
G-DEV	20,236 (68%)	9,464 (32%)	29,700
G-TEST-L	20,064 (68%)	9,636 (32%)	29,700
G-TEST-S	1,068 (71%)	432 (29%)	1,500
G-TEST-S-R	1,174 (78%)	326 (22%)	1,500
W-ATT-TRAIN	61,447 (88%)	8,079 (12%)	69,526
W-ATT-DEV	20,405 (88%)	2,755 (12%)	23,160
W-ATT-TEST	20,422 (88%)	2,756 (12%)	23,178

Table 1: Statistics of the datasets used.

Our main contributions are: (i) We release a dataset of 1.6M moderated user comments. (ii) We introduce a novel, deep, classification-specific attention mechanism and we show that an RNN with our attention mechanism outperforms the previous state of the art in user comment moderation. (iii) Unlike previous work, we also consider a semi-automatic scenario, along with threshold tuning and evaluation measures for it. (iv) We show that the attention mechanism can automatically highlight suspicious words for free, without manually highlighting words in the training data.

2 Datasets

We first discuss the datasets we used, to help acquaint the reader with the problem.

2.1 Gazzetta comments

There are approx. 1.45M training comments (covering Jan. 1, 2015 to Oct. 6, 2016) in the Gazzetta dataset; we call them G-TRAIN-L (Table 1). Some experiments use only the first 100K comments of G-TRAIN-L, called G-TRAIN-S. An additional set of 60,900 comments (Oct. 7 to Nov. 11, 2016) was split to development (G-DEV, 29,700 comments), large test (G-TEST-L, 29,700), and small test set (G-TEST-S, 1,500). Gazzetta's moderators (2 full-time, plus journalists occasionally helping) are occasionally instructed to be stricter (e.g., during violent events). To get a more accurate view of performance in normal situtations, we manually re-moderated (labeled as 'accept' or 'reject') the comments of G-TEST-S, producing G-TEST-S-R. The reject ratio is approx. 30% in all subsets, except for G-TEST-S-R where it drops to 22%, because there are no occasions where the moderators were instructed to be stricter in G-TEST-S-R.

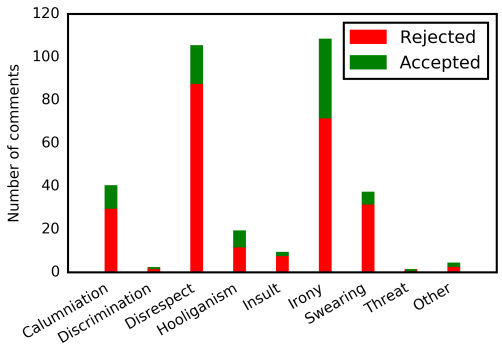

Figure 2: Re-moderated comments with at least one snippet of the corresponding category.

Each G-TEST-S-R comment was re-moderated by five annotators. Krippendorff's (2004) alpha was 0.4762, close to the value (0.45) reported by Wulczyn et al. (2017) for the Wikipedia 'attacks' dataset. Using Cohen's Kappa (Cohen, 1960), the mean pairwise agreement was 0.4749. The mean pairwise percentage of agreement (% of comments each pair of annotators agreed on) was 81.33%. Cohen's Kappa and Krippendorff's alpha lead to lower scores, because they account for agreement by chance, which is high when there is class imbalance (22% reject, 78% accept in G-TEST-S-R).

During the re-moderation of G-TEST-S-R, the annotators were also asked to highlight snippets they considered suspicious, i.e., words or phrases that could lead a moderator to consider rejecting each comment.[5] We also asked the annotators to classify each snippet into one of the following categories: calumniation (e.g., false accusations), discrimination (e.g., racism), disrespect (e.g., looking down at a profession), hooliganism (e.g., calling for violence), insult (e.g., making fun of appearance), irony, swearing, threat, other. Figure 2 shows how many comments of G-TEST-S-R contained at least one snippet of each category, according to the majority of annotators; e.g., a comment counts as containing irony if at least 3 annotators annotated it with an irony snippet (not necessarily the same). The gold class of each comment (accept or reject) is determined by the majority of the annotators. Irony and disrespect are particularly frequent in both classes, followed by calumniation, swearing, hooliganism, insults. Notice that comments that contain irony, disrespect etc. are not necessarily rejected. They are, however, more likely in the rejected class, considering that the accepted comments are 2.5 times more than the rejected ones (78% vs. 22%).

We also provide 300-dimensional word embeddings, pre-trained on approx. 5.2M comments (268M tokens) from Gazzetta using WORD2VEC (Mikolov et al., 2013a,b).[6] This larger dataset cannot be used to directly train classifiers, because most of its comments are from a period (before 2015) when Gazzetta did not employ moderators.

2.2 Wikipedia comments

The Wikipedia 'attacks' dataset (Wulczyn et al., 2017) contains approx. 115K English Wikipedia talk page comments, which were labeled as containing personal attacks or not. Each comment was labeled by at least 10 annotators. Inter-annotator agreement, measured on a random sample of 1K comments using Krippendorff's (2004) alpha, was 0.45. The gold label of each comment is determined by the majority of annotators, leading to *binary labels* (accept, reject). Alternatively, the gold label is the percentage of annotators that labeled the comment as 'accept' (or 'reject'), leading to *probabilistic labels*.[7] The dataset is split in three parts (Table 1): training (W-ATT-TRAIN, 69,526 comments), development (W-ATT-DEV, 23,160), and test (W-ATT-TEST, 23,178). In all three parts, the rejected comments are 12%, but this is an artificial ratio (Wulczyn et al. oversampled comments posted by banned users). By contrast, the ratio of rejected comments in all the Gazzetta subsets is the truly observed one. The Wikipedia comments are also longer (median length 38 tokens) compared to Gazzetta's (median length 25 tokens).

Wulczyn et al. (2017) also provide two additional datasets of English Wikipedia talk page comments, which are not used in this paper. The first one, called 'aggression' dataset, contains the same comments as the 'attacks' dataset, now labeled as 'aggressive' or not. The (probabilistic) labels of the 'attacks' and 'aggression' datasets are very highly correlated (0.8992 Spearman, 0.9718 Pearson) and we did not consider the aggression dataset any further. The second additional dataset, called 'toxicity' dataset, contains approx. 160K comments labeled as being toxic or not. Experiments we reported elsewhere (Pavlopoulos et al., 2017) show that results on the 'attacks' and 'toxicity' datasets are very similar; we do not include

[5] Treating snippet overlaps as agreements, the mean pairwise Dice coefficient for snippet highlighting was 50.03%.

[6] We used CBOW, window size 5, min. term freq. 5, negative sampling, obtaining a vocabulary size of approx. 478K.

[7] We also construct probabilistic labels for G-TEST-S-R, where there are five annotators.

results on the latter in this paper to save space.

3 Methods

We experimented with an RNN operating on word embeddings, the same RNN enhanced with our attention mechanism (*a*-RNN), a vanilla convolutional neural network (CNN) also operating on word embeddings, the DETOX system of Wulczyn et al. (2017), and a baseline that uses word lists.

3.1 DETOX

DETOX (Wulczyn et al., 2017) was the previous state of the art in comment moderation, in the sense that it had the best reported results on the Wikipedia datasets (Section 2.2), which were in turn the largest previous publicly available dataset of moderated user comments.[8] DETOX represents each comment as a bag of word n-grams ($n \leq 2$, each comment becomes a bag containing its 1-grams and 2-grams) or a bag of character n-grams ($n \leq 5$, each comment becomes a bag containing character 1-grams, ..., 5-grams). DETOX can rely on a logistic regression (LR) or MLP classifier, and it can use binary or probabilistic gold labels (Section 2.2) during training.

We used the DETOX implementation provided by Wulczyn et al. and the same grid search (and code) to tune the hyper-parameters of DETOX that select word or character n-grams, classifier (LR or MLP), and gold labels (binary or probabilistic). For Gazzetta, only binary gold labels were possible, since G-TRAIN-L and G-TRAIN-S have a single gold label per comment. Unlike Wulczyn et al., we tuned the hyper-parameters by evaluating (computing AUC and Spearman, Section 4) on a random 2% of held-out comments of W-ATT-TRAIN or G-TRAIN-S, instead of the development subsets, to be able to obtain more realistic results from the development sets while developing the methods. For both Wikipedia and Gazzetta, the tuning selected character n-grams, as in the work of Wulczyn et al. Also, for both Wikipedia and Gazzetta, it preferred LR to MLP, whereas Wulczyn et al. reported slightly higher performance for the MLP on W-ATT-DEV.[9] The tuning also selected probabilistic labels for Wikipedia, as in the work of Wulczyn et al.

3.2 RNN-based methods

RNN: The RNN method is a chain of GRU cells (Cho et al., 2014) that transforms the tokens $w_1 \ldots, w_k$ of each comment to the hidden states $h_1 \ldots, h_k$, followed by an LR layer that uses h_k to classify the comment (accept, reject). Formally, given the vocabulary V, a matrix $E \in \mathbb{R}^{d \times |V|}$ containing d-dimensional word embeddings, an initial h_0, and a comment $c = \langle w_1, \ldots, w_k \rangle$, the RNN computes $h_1, \ldots, h_k$ as follows ($h_t \in \mathbb{R}^m$):

$$\begin{aligned} \tilde{h}_t &= \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \\ z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\ r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \end{aligned}$$

where $\tilde{h}_t \in \mathbb{R}^m$ is the proposed hidden state at position t, obtained by considering the word embedding x_t of token w_t and the previous hidden state h_{t-1}; $\odot$ denotes element-wise multiplication; $r_t \in \mathbb{R}^m$ is the reset gate (for r_t all zeros, it allows the RNN to forget the previous state h_{t-1}); $z_t \in \mathbb{R}^m$ is the update gate (for z_t all zeros, it allows the RNN to ignore the new proposed $\tilde{h}_t$, hence also x_t, and copy h_{t-1} as h_t); σ is the sigmoid function; $W_h, W_z, W_r \in \mathbb{R}^{m \times d}$; $U_h, U_z, U_r \in \mathbb{R}^{m \times m}$; $b_h, b_z, b_r \in \mathbb{R}^m$. Once h_k has been computed, the LR layer estimates the probability that comment c should be rejected, with $W_p \in \mathbb{R}^{1 \times m}$, $b_p \in \mathbb{R}$:

$$P_{\text{RNN}}(\textit{reject}|c) = \sigma(W_p h_k + b_p)$$

***a*-RNN:** When the attention mechanism is added, the LR layer considers the weighted sum h_{sum} of all the hidden states, instead of just h_k (Fig. 3):[10]

$$\begin{aligned} h_{sum} &= \sum_{t=1}^{k} a_t h_t \qquad (1) \\ P_{a-\text{RNN}}(\textit{reject}|c) &= \sigma(W_p h_{sum} + b_p) \end{aligned}$$

The weights a_t are produced by an attention mech-

[8] Two of the co-authors of Wulczyn et al. (2017) are with Jigsaw, who recently announced Perspective, a system to detect 'toxic' comments. Perspective is not the same as DETOX (personal communication), but we were unable to obtain scientific articles describing it. An API for Perspective is available at https://www.perspectiveapi.com/, but we did not have access to the API at the time the experiments of this paper were carried out.

[9] We repeated the tuning by evaluating on W-ATT-DEV, and again character n-grams with LR were selected.

[10] We tried replacing the LR layer by a deeper classification MLP, and the RNN chain by a bidirectional RNN (Schuster and Paliwal, 1997), but there were no improvements.

anism, which is an MLP with l layers:

$$a_t^{(1)} = \text{RELU}(W^{(1)} h_t + b^{(1)}) \quad (2)$$

$$\dots$$

$$a_t^{(l-1)} = \text{RELU}(W^{(l-1)} a_t^{(l-2)} + b^{(l-1)})$$

$$a_t^{(l)} = W^{(l)} a_t^{(l-1)} + b^{(l)}$$

$$a_t = \texttt{softmax}(a_t^{(l)}; a_1^{(l)}, \dots, a_k^{(l)}) \quad (3)$$

where $a_t^{(1)}, \dots, a_t^{(l-1)} \in \mathbb{R}^r$, $a_t^{(l)}, a_t \in \mathbb{R}$, $W^{(1)} \in \mathbb{R}^{r \times m}$, $W^{(2)}, \dots, W^{(l-1)} \in \mathbb{R}^{r \times r}$, $W^{(l)} \in \mathbb{R}^{1 \times r}$, $b^{(1)}, \dots, b^{(l-1)} \in \mathbb{R}^r$, $b^{(l)} \in \mathbb{R}$. The `softmax` operates across the $a_t^{(l)}$ ($t = 1, \dots, k$), making the weights a_t sum to 1. Our attention mechanism differs from most previous ones (Mnih et al., 2014; Bahdanau et al., 2015; Xu et al., 2015; Luong et al., 2015) in that it is used in a classification setting, where there is no previously generated output subsequence (e.g., partly generated translation) to drive the attention (e.g., assign more weight to source words to translate next), unlike seq2seq models (Sutskever et al., 2014). It assigns larger weights a_t to hidden states h_t corresponding to positions where there is more evidence that the comment should be accepted or rejected.

Yang et al. (2016) use a similar attention mechanism, but ours is deeper. In effect they always set $l = 2$, whereas we allow l to be larger (tuning selects $l = 4$).[11] On the other hand, the attention mechanism of Yang et al. is part of a classification method for longer texts (e.g., product reviews). Their method uses two GRU RNNs, both bidirectional (Schuster and Paliwal, 1997), one turning the word embeddings of each sentence to a sentence embedding, and one turning the sentence embeddings to a document embedding, which is then fed to an LR layer. Yang et al. use their attention mechanism in both RNNs, to assign attention scores to words and sentences. We consider shorter texts (comments), we have a single RNN, and we assign attention scores to words only.[12]

***da*-CENT:** We also experiment with a variant of *a*-RNN, called *da*-CENT, which does not use the hidden states of the RNN. The input to the first layer of the attention mechanism is now directly the embedding x_t instead of h_t (cf. Eq. 2), and

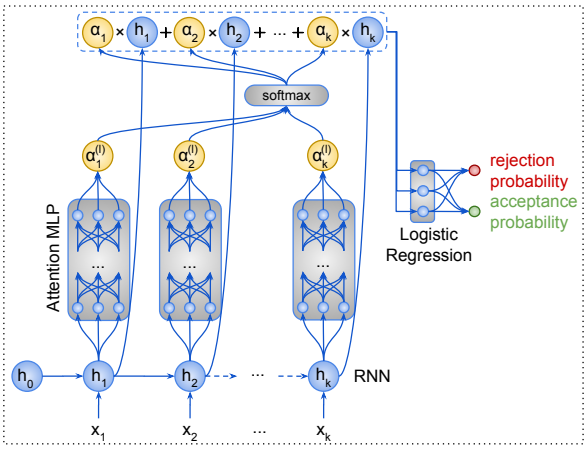

Figure 3: Illustration of *a*-RNN.

h_{sum} is now the weighted sum (centroid) of word embeddings $h_{sum} = \sum_{t=1}^{k} a_t x_t$ (cf. Eq. 1).[13]

We set $l = 4, d = 300, r = m = 128$, having tuned all hyper-parameters on the same 2% held-out comments of W-ATT-TRAIN or G-TRAIN-S that were used to tune DETOX. We use Glorot initialization (Glorot and Bengio, 2010), categorical cross-entropy loss, and Adam (Kingma and Ba, 2015).[14] Early stopping evaluates on the same held-out subsets. For Gazzetta, word embeddings are initialized to the WORD2VEC embeddings we provide (Section 2.1). For Wikipedia, they are initialized to GLOVE embeddings (Pennington et al., 2014).[15] In both cases, the embeddings are updated during backpropagation. Out of vocabulary (OOV) words, meaning words for which we have no initial embeddings, are mapped to a single randomly initialized embedding, also updated.

3.3 CNN

We also compare against a vanilla CNN operating on word embeddings. We describe the CNN only briefly, because it is very similar to that of of Kim (2014); see also Goldberg (2016) for an introduction to CNNs, and Zhang and Wallace (2015).

For Wikipedia comments, we use a 'narrow' convolution layer, with kernels sliding (stride 1) over (entire) embeddings of word n-grams of sizes $n = 1, \dots, 4$. We use 300 kernels for each n value, a total of 1,200 kernels. The outputs of each kernel, obtained by applying the kernel to the different n-grams of a comment c, are then

[11] Yang et al. use tanh instead of RELU in Eq. 2, which works worse in our case, and no bias $b^{(l)}$ in the l-th layer.

[12] We tried a bidirectional instead of unidirectional GRU chain in our methods, also replacing the LR layer by a deeper classification MLP, but there were no improvements.

[13] For experiments with additional variants of *a*-RNN, consult Pavlopoulos et al. (2017).

[14] We implemented the methods of this sub-section using Keras (`keras.io`) and TensorFlow (`tensorflow.org`).

[15] See `https://nlp.stanford.edu/projects/glove/`. We use 'Common Crawl' (840B tokens).

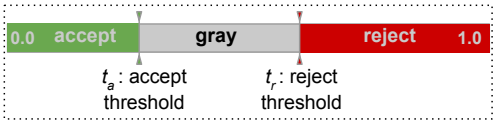

Figure 4: Illustration of threshold tuning.

max-pooled, leading to a single output per kernel. The resulting feature vector (1,200 max-pooled outputs) goes through a dropout layer (Hinton et al., 2012) ($p = 0.5$), and then to an LR layer, which provides $P_{\text{CNN}}(\textit{reject}|c)$. For Gazzetta, the CNN is the same, except that $n = 1, \ldots, 5$, leading to 1,500 features per comment. All hyperparameters were tuned on the 2% held-out comments of W-ATT-TRAIN or G-TRAIN-S that were used to tune the other methods. Again, we use 300-dimensional embeddings, which are now randomly initialized, since tuning indicated this was better than initializing to pre-trained embeddings. OOV words are treated as in the RNN-based methods. All embeddings are updated during backpropagation. Early stopping evaluates on the held-out subsets. Again, we use Glorot initialization, categorical cross-entropy loss, and Adam.[16]

3.4 LIST baseline

A baseline, called LIST, collects every word w that occurs in more than 10 (for W-ATT-TRAIN, G-TRAIN-S) or 100 comments (for G-TRAIN-L) in the training set, along with the precision of w, i.e., the ratio of rejected training comments containing w divided by the total number of training comments containing w. The resulting lists contain 10,423, 16,864, and 21,940 word types, when using W-ATT-TRAIN, G-TRAIN-S, G-TRAIN-L, respectively. For a comment c, LIST returns as $P_{\text{LIST}}(\textit{reject}|c)$ the maximum precision of all the words in c.

3.5 Tuning thresholds

All methods produce a $p = P(\textit{reject}|c)$ per comment c. In semi-automatic moderation (Fig. 1), a comment is directly rejected if its p is above a rejection theshold t_r, it is directly accepted if p is below an acceptance threshold t_a, and it is shown to a moderator if $t_a \leq p \leq t_r$ (gray zone of Fig. 4).

In our experience, moderators (or their employers) can easily specify the approximate percentage of comments they can afford to check manually (e.g., 20% daily) or, equivalently, the approximate percentage of comments the system should handle automatically. We call *coverage* the latter percentage; hence, $1 - \textit{coverage}$ is the approximate percentage of comments to be checked manually. By contrast, moderators are baffled when asked to tune t_r and t_a directly. Consequently, we ask them to specify the approximate desired coverage. We then sort the comments of the development set (G-DEV or W-ATT-DEV) by p, and slide t_a from 0.0 to 1.0 (Fig. 4). For each t_a value, we set t_r to the value that leaves a $1 - \textit{coverage}$ percentage of development comments in the gray zone ($t_a \leq p \leq t_r$). We then select the t_a (and t_r) that maximizes the weighted harmonic mean $F_\beta(P_{\textit{reject}}, P_{\textit{accept}})$ on the development set:

$$F_\beta(P_{\textit{reject}}, P_{\textit{accept}}) = \frac{(1+\beta^2) \cdot P_{\textit{reject}} \cdot P_{\textit{accept}}}{\beta^2 \cdot P_{\textit{reject}} + P_{\textit{accept}}}$$

where $P_{\textit{reject}}$ is the *rejection precision* (correctly rejected comments divided by rejected comments) and $P_{\textit{accept}}$ is the *acceptance precision* (correctly accepted divided by accepted). Intuitively, coverage sets the width of the gray zone, whereas $P_{\textit{reject}}$ and $P_{\textit{accept}}$ show how certain we can be that the red (reject) and green (accept) zones are free of misclassified comments. We set $\beta = 2$, emphasizing $P_{\textit{accept}}$, because moderators are more worried about wrongly accepting abusive comments than wrongly rejecting non-abusive ones.[17] The selected t_a, t_r (tuned on development data) are then used in experiments on test data. In fully automatic moderation, $\textit{coverage} = 100$ and $t_a = t_r$; otherwise, threshold tuning is identical.

4 Experimental results

4.1 Comment classification evaluation

Following Wulczyn et al. (2017), we report in Table 2 AUC scores (area under ROC curve), along with Spearman correlations between system-generated probabilities $P(\textit{accept}|c)$ and human probabilistic gold labels (Section 2.2) when probabilistic gold labels are available.[18] Wulczyn et al. reported DETOX results only on W-ATT-DEV, shown in brackets. Table 2 shows that RNN is

[16] We implemented the CNN directly in TensorFlow.

[17] More precisely, when computing F_β, we reorder the development comments by time posted, and split them into batches of 100. For each t_a (and t_r) value, we compute F_β per batch and macro-average across batches. The resulting thresholds lead to F_β scores that are more stable over time.

[18] When computing AUC, the gold label is the majority label of the annotators. When computing Spearman, the gold label is probabilistic (% of annotators that accepted the comment). The decisions of the systems are always probabilistic.

Training	Evaluation	Score	RNN	*a*-RNN	*da*-CENT	CNN	DETOX	LIST
G-TRAIN-S	G-DEV	AUC	75.75	**76.19**	74.91	70.97	72.50	61.47
	G-TEST-L	AUC	75.10	**76.15**	74.72	71.34	72.06	61.59
	G-TEST-S	AUC	74.40	**75.83**	73.79	70.88	71.59	61.26
	G-TEST-S-R	AUC	80.27	**80.41**	78.82	76.03	75.67	64.19
		Spearman	51.89	**52.51**	49.22	42.88	43.80	24.33
G-TRAIN-L	G-DEV	AUC	79.50	**79.64**	78.73	77.57	–	67.04
	G-TEST-L	AUC	79.41	**79.58**	78.64	77.35	–	67.06
	G-TEST-S	AUC	79.23	**79.67**	78.62	78.16	–	66.17
	G-TEST-S-R	AUC	84.17	**84.69**	83.53	83.98	–	69.51
		Spearman	59.31	**60.87**	57.82	55.90	–	33.61
W-ATT-TRAIN	W-ATT-DEV	AUC	97.39	**97.46**	96.58	96.91	96.26 (96.59)	93.05
		Spearman	**71.92**	71.59	68.59	70.06	67.75 (68.17)	55.39
	W-ATT-TEST	AUC	**97.71**	97.68	96.83	97.07	96.71	92.91
		Spearman	**72.79**	72.32	68.86	70.21	68.09	54.55

Table 2: Comment classification results. Scores reported by Wulczyn et al. (2017) are shown in brackets.

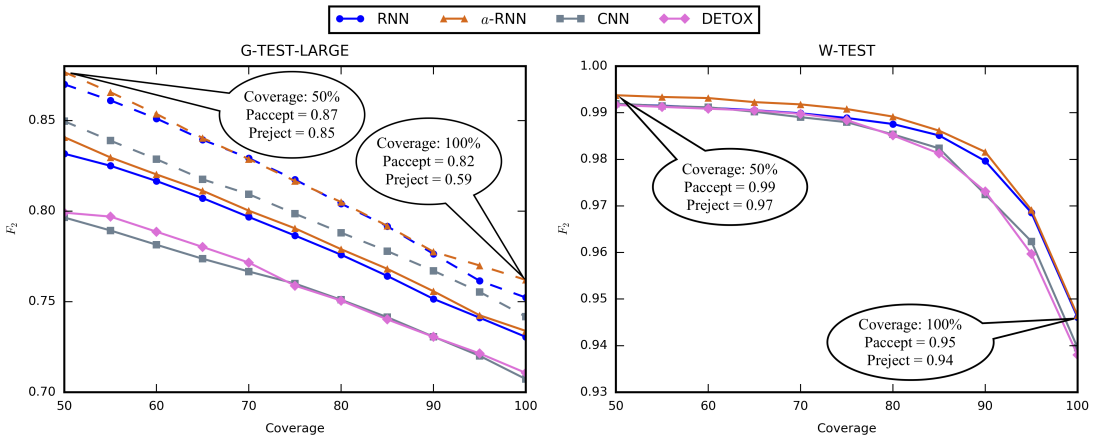

Figure 5: F_2 scores for varying coverage. Dotted lines were obtained using a larger training set.

always better than CNN and DETOX; there is no clear winner between CNN and DETOX. Furthermore, *a*-RNN is always better than RNN on Gazzetta comments, but not on Wikipedia comments, where RNN is overall slightly better according to Table 2. Also, *da*-CENT is always worse than *a*-RNN and RNN, confirming that the hidden states (intuitively, context-aware word embeddings) of the RNN chain are important, even with the attention mechanism. Increasing the size of the Gazzetta training set (G-TRAIN-S to G-TRAIN-L) significantly improves the performance of all methods. The implementation of DETOX could not handle the size of G-TRAIN-L, which is why we do not report DETOX results for G-TRAIN-L. Notice, also, that the Wikipedia dataset is easier than the Gazzetta one (all methods perform better on Wikipedia comments, compared to Gazzetta).

Figure 5 shows $F_2(P_{reject}, P_{accept})$ on G-TEST-L and W-ATT-TEST, when t_a, t_r are tuned on G-DEV, W-ATT-DEV for varying coverage. For G-TEST-L, we show results training on G-TRAIN-S (solid lines) and G-TRAIN-L (dotted). The differences between RNN and *a*-RNN are again small, but it is now easier to see that *a*-RNN is overall better. Again, *a*-RNN and RNN are better than CNN and DETOX. All three deep learning methods benefit from the larger training set (dotted). In Wikipedia, *a*-RNN obtains $P_{accept}, P_{reject} \geq 0.94$ for all coverages (Fig. 5, call-outs). On the more difficult Gazzetta dataset, *a*-RNN still obtains $P_{accept}, P_{reject} \geq 0.85$ when tuned for 50% coverage. When tuned for 100% coverage, comments for which the system is uncertain (gray zone) cannot be avoided and there are inevitably more misclassifications; the use of F_2 during threshold tuning places more emphasis on avoiding wrongly accepted comments, leading to high P_{accept} (0.82), at the expense of wrongly rejected comments, i.e., sacrificing P_{reject} (0.59). On the re-moderated G-TEST-S-R (similar diagrams, not shown), P_{accept}, P_{reject} become 0.96, 0.88 for coverage 50%, and 0.92, 0.48 for coverage 100%.

We also repeated the *annotator ensemble* experiment of Wulczyn et al. (2017) on 8K randomly chosen comments of W-ATT-TEST (4K comments

Figure 6: Word highlighting by a-RNN.

from random users, 4K comments from banned users).[19] The decisions of 10 randomly chosen annotators (possibly different per comment) were used to construct the gold label of each comment. The gold labels were then compared to the decisions of the systems and the decisions of an ensemble of k other annotators, k ranging from 1 to 10. Table 3 shows the mean AUC and Spearman scores, averaged over 25 runs of the experiment, along with standard errrors (in brackets). We conclude that RNN and a-RNN are as good as an ensemble of 7 human annotators; CNN is as good as 4 annotators; DETOX is as good as 4 in AUC and 3 annotators in Spearman correlation, which is consistent with the results of Wulczyn et al. (2017).

k	AUC	Spearman
1	84.34 (0.64)	53.82 (0.77)
2	92.14 (0.42)	61.61 (0.51)
3	95.05 (0.41)	65.20 (0.55)
4	96.43 (0.31)	67.25 (0.52)
5	97.17 (0.25)	68.46 (0.49)
6	97.68 (0.22)	69.45 (0.40)
7	97.99 (0.21)	70.16 (0.33)
8	98.21 (0.18)	70.67 (0.31)
9	98.39 (0.15)	71.12 (0.32)
10	98.51 (0.14)	71.50 (0.34)
RNN	98.03 (0.13)	70.58 (0.27)
a-RNN	98.00 (0.13)	70.19 (0.30)
CNN	97.29 (0.14)	67.92 (0.32)
DETOX	97.00 (0.14)	66.21 (0.32)

Table 3: Comparing to an ensemble of k humans.

4.2 Snippet highlighting evaluation

To investigate if the attention scores of a-RNN can highlight suspicious words, we focused on G-TEST-S-R, the only dataset with suspicious snippets annotated by humans. We removed comments with no human-annotated snippets, leaving 841 comments (515 accepted, 326 rejected), a total of 40,572 tokens, of which 13,146 were inside a suspicious snippet of at least one annotator. In each remaining comment, each token was assigned a *gold suspiciousness* score, defined as the percentage of annotators that included it in their snippets.

We evaluated three methods that score each token w_t of a comment c for suspiciousness. The first one assigns to each w_t the attention score a_t (Eq. 3) of a-RNN (trained on G-TRAIN-L). The second method assigns to each w_t its precision, as computed by LIST (Section 3.4). The third method (RAND) assigns to each w_t a random (uniform distribution) score between 0 and 1. In the latter two methods, a `softmax` is applied to the scores of all the tokens per comment, as in a-RNN. Figure 6 shows three comments (from W-ATT-TEST) highlighted by a-RNN; heat corresponds to attention.[20]

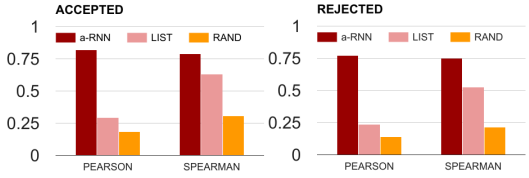

Figure 7: Suspicious snippet highlighting results.

We computed Pearson and Spearman correlations between the gold suspiciousness scores and the scores of the three methods on the 40,572 tokens. Figure 7 shows the correlations on comments that were accepted (left) and rejected (right) by the majority of moderators. In both cases, a-RNN performs better than LIST and RAND by both Pearson and Spearman correlations. The high Pearson correlations of a-RNN also show that its attention scores are to a large extent linearly related to the gold ones. By contrast, LIST performs reasonably well in terms of Spearman correlation, but much worse in terms of Pearson, indicating that its precision scores rank reasonably well the tokens from most to least suspicious ones, but are not linearly related to the gold scores.

5 Related work

Djuric et al. (2015) experimented with 952K manually moderated comments from Yahoo Finance, but their dataset is not publicly available. They convert each comment to a comment embedding using DOC2VEC (Le and Mikolov, 2014), which is then fed to an LR classifier. Nobata et al. (2016) experimented with approx. 3.3M manually moderated comments from Yahoo Finance and News; their data are also not available.[21] They used Vowpal Wabbit[22] with character n-grams ($n = 3, \ldots, 5$) and word n-grams ($n = 1, 2$), handcrafted features (e.g., number of capitalized or black-listed words), features based on dependency

[19] We used the protocol, code, and data of Wulczyn et al.

[20] In innocent comments, a-RNN spreads its attention to all tokens, leading to quasi-uniform low color intensity.

[21] According to Nobata et al., their clean test dataset (2K comments) would be made available, but it is currently not.

[22] See `http://hunch.net/~vw/`.

trees, averages of WORD2VEC embeddings, and DOC2VEC-like embeddings. Character n-grams were the best, on their own outperforming Djuric et al. (2015). The best results, however, were obtained using all features. We use no hand-crafted features and parsers, making our methods more easily portable to other domains and languages.

Mehdad et al. (2016) train a (token or character-based) RNN language model per class (accept, reject), and use the probability ratio of the two models to accept or reject user comments. Experiments on the dataset of Djuric et al. (2015), however, showed that their method (RNNLMs) performed worse than a combination of SVM and Naive Bayes classifiers (NBSVM) that used character and token n-grams. An LR classifier operating on DOC2VEC-like comment embeddings (Le and Mikolov, 2014) also performed worse than NBSVM. To surpass NBSVM, Mehdad et al. used an SVM to combine features from their three other methods (RNNLMs, LR with DOC2VEC, NBSVM).

Wulczyn et al. (2017) experimented with character and word n-grams. We included their dataset and moderation system (DETOX) in our experiments. Waseem et al. (2016) used approx. 17K tweets annotated for hate speech. Their best results were obtained using an LR classifier with character n-grams ($n = 1, \ldots, 4$), plus gender. Warner and Hirschberg (2012) aimed to detect anti-semitic speech, experimenting with 9K paragraphs and a linear SVM. Their features consider windows of at most 5 tokens, examining the tokens of each window, their order, POS tags, Brown clusters etc., following Yarowsky (1994).

Cheng et al. (2015) aimed to predict which users would be banned from on-line communities. Their best system used a random forest or LR classifier, with features examining readability, activity (e.g., number of posts daily), community and moderator reactions (e.g., up-votes, number of deleted posts).

Sood et al. (2012a; 2012b) experimented with 6.5K comments from Yahoo Buzz, moderated via crowdsourcing. They showed that a linear SVM, representing each comment as a bag of word bigrams and stems, performs better than word lists. Their best results were obtained by combining the SVM with a word list and edit distance.

Yin et al. (2009) used posts from chat rooms and discussion fora (<15K posts in total) to train an SVM to detect online harassment. They used TF-IDF, sentiment, and context features (e.g., similarity to other posts in a thread). Our methods might also benefit by considering threads, rather than individual comments. Yin at al. point out that unlike other abusive content, spam in comments or dicsussion fora (Mishne et al., 2005; Niu et al., 2007) is off-topic and serves a commercial purpose. Spam is unlikely in Wikipedia discussions and not an issue in the Gazzetta dataset (Fig. 2).

For a more extensive discussion of related work, consult Pavlopoulos et al. (2017).

6 Conclusions

We experimented with a new publicly available dataset of 1.6M moderated user comments from a Greek sports news portal and an existing dataset of 115K English Wikipedia talk page comments. We showed that a GRU RNN operating on word embeddings outpeforms the previous state of the art, which used an LR or MLP classifier with character or word n-gram features, also outperforming a vanilla CNN operating on word embeddings, and a baseline that uses an automatically constructed word list with precision scores. A novel, deep, classification-specific attention mechanism improves further the overall results of the RNN, and can also highlight suspicious words for free, without including highlighted words in the training data. We considered both fully automatic and semi-automatic moderation, along with threshold tuning and evaluation measures for both.

We plan to consider user-specific information (e.g., ratio of comments rejected in the past) (Cheng et al., 2015; Waseem and Hovy, 2016) and explore character-level RNNs or CNNs (Zhang et al., 2015), e.g., as a first layer to produce embeddings of unknown words from characters (dos Santos and Zadrozny, 2014; Ling et al., 2015), which would then be passed on to our current methods that operate on word embeddings.

Acknowledgments

This work was funded by Google's Digital News Initiative (project ML2P, contract 362826).[23] We are grateful to Gazzetta for the data they provided. We also thank Gazzetta's moderators for their feedback, insights, and advice.

[23] See https://digitalnewsinitiative.com/.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*. San Diego, CA, USA.

Justin Cheng, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2015. Antisocial behavior in online discussion communities. In *Proceedings of the 9th International AAAI Conference on Web and Social Media*. Oxford University, England, pages 61–70.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, pages 1724–1734.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20(1):37–46.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web*. Florence, Italy, pages 29–30.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning*. Beijing, China, pages 1818–1826.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*. Sardinia, Italy, pages 249–256.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research* 57:345–420.

Yoav Goldberg. 2017. *Neural Network Methods in Natural Language Processing*. Morgan and Claypool Publishers.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* abs/1207.0580.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*. San Diego, CA, USA.

Klaus Krippendorff. 2004. *Content Analysis: An Introduction to Its Methodology (2nd edition)*. Sage Publications.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*. Beijing, China, pages 1188–1196.

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 1520–1530.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 1412–1421.

Yashar Mehdad and Joel Tetreault. 2016. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Los Angeles, CA, pages 299–303.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at International Conference on Learning Representations*. Scottsdale, AZ, USA.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, GA, pages 746–751.

Gilad Mishne, David Carmel, and Ronny Lempel. 2005. Blocking blog spam with language model disagreement. In *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web*. Chiba, Japan.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*. Montreal, Canada, pages 2204–2212.

Yuan Niu, Yi-Min Wang, Hao Chen, Ming Ma, and Francis Hsu. 2007. A quantitative study of forum spamming using context-based analysis. In *Proceedings of the 14th Annual Network and Distributed System Security Symposium*. San Diego, CA, pages 79–92.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*. Montreal, Canada, pages 145–153.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deep learning for user comment moderation. In *Proceedings of the 1st ACL Workshop on Abusive Language Online*. Vancouver, Canada.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, pages 1532–1543.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transacions of Signal Processing* 45(11):2673–2681.

Sara Sood, Judd Antin, and Elizabeth F. Churchill. 2012a. Profanity use in online communities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Austin, TX, USA, pages 1481–1490.

Sara Sood, Judd Antin, and Elizabeth F. Churchill. 2012b. Using crowdsourcing to improve profanity detection. In *AAAI Spring Symposium: Wisdom of the Crowd*. Stanford, CA, USA, pages 69–74.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*. Montreal, Canada, pages 3104–3112.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the World Wide Web. In *Proceedings of the 2nd Workshop on Language in Social Media*. Montreal, Canada, pages 19–26.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*. San Diego, CA, USA, pages 88–93.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*. Perth, Australia, pages 1391–1399.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*. Lille, France, pages 2048–2057.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, CA, USA, pages 1480–1489.

David Yarowsky. 1994. Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*. Las Cruces, NM, USA, pages 88–95.

Dawei Yin, Zhenzhen Xue, Liangjie Hong, Brian D Davison, April Kontostathis, and Lynne Edwards. 2009. Detection of harassment on Web 2.0. In *Proceedings of the WWW workshop on Content Analysis in the Web 2.0*. Madrid, Spain.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*. Montreal, Canada, pages 649–657.

Ye Zhang and Byron C. Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR* abs/1510.03820.