Improved Abusive Comment Moderation with User Embeddings

Abstract

Experimenting with a dataset of approximately 1.6M user comments from a Greek news sports portal, we explore how a state of the art RNN-based moderation method can be improved by adding user embeddings, user type embeddings, user biases, or user type biases. We observe improvements in all cases, with user embeddings leading to the biggest performance gains.

Introduction

News portals often allow their readers to comment on articles, in order to get feedback, engage their readers, and build customer loyalty. User comments, however, can also be abusive (e.g., bullying, profanity, hate speech), damaging the reputation of news portals, making them liable to fines (e.g., when hosting comments encouraging illegal actions), and putting off readers. Large news portals often employ moderators, who are frequently overwhelmed by the volume and abusiveness of comments. 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.

In previous work BIBREF0 , we introduced a new dataset of approx. 1.6M manually moderated user comments from a Greek sports news portal, called Gazzetta, which we made publicly available. Experimenting on that dataset and the datasets of Wulczyn et al. Wulczyn2017, which contain moderated English Wikipedia comments, we showed that a method based on a Recurrent Neural Network (rnn) outperforms detox BIBREF1 , the previous state of the art in automatic user content moderation. Our previous work, however, considered only the texts of the comments, ignoring user-specific information

(e.g., number of previously accepted or rejected comments of each user). Here we add user embeddings or user type embeddings to our rnn-based method, i.e., dense vectors that represent individual users or user types, similarly to word embeddings that represent words BIBREF2 , BIBREF3 . Experiments on Gazzetta comments show that both user embeddings and user type embeddings improve the performance of our rnn-based method, with user embeddings helping more. User-specific or user-type-specific scalar biases also help to a lesser extent.

Dataset

We first discuss the dataset we used, to help acquaint the reader with the problem. The dataset contains Greek comments from Gazzetta BIBREF0 . There are approximately 1.45M training comments (covering Jan. 1, 2015 to Oct. 6, 2016); we call them g-train (Table TABREF5). An additional set of 60,900 comments (Oct. 7 to Nov. 11, 2016) was split to development set (g-dev, 29,700 comments) and test set (g-test, 29,700). Each comment has a gold label (`accept', `reject'). The user id of the author of each comment is also available, but user id s were not used in our previous work.

When experimenting with user type embeddings or biases, we group the users into the following types. INLINEFORM0 is the number of training comments posted by user (id) INLINEFORM1 . INLINEFORM2 is the ratio of training comments posted by INLINEFORM3 that were rejected.

Red: Users with INLINEFORM0 and INLINEFORM1 .

Yellow: INLINEFORM0 and INLINEFORM1 .

Green: INLINEFORM0 and INLINEFORM1 .

Unknown: Users with INLINEFORM0 .

Table TABREF6 shows the number of users per type.

Methods

rnn: This is the rnn-based method of our previous work BIBREF0 . It is a chain of gru cells BIBREF4 that transforms the tokens INLINEFORM0 of each comment to the hidden states INLINEFORM1 (INLINEFORM2). Once INLINEFORM3 has been computed, a logistic regression (lr) layer estimates the probability that comment INLINEFORM4 should be rejected: DISPLAYFORM0

INLINEFORM0 is the sigmoid function, INLINEFORM1 , INLINEFORM2 .

uernn: This is the rnn-based method with user embeddings added. Each user INLINEFORM0 of the training set with INLINEFORM1 is mapped to a user-specific embedding INLINEFORM2 . Users with INLINEFORM3 are mapped to a single `unknown' user embedding. The lr layer is modified as follows; INLINEFORM4 is the embedding of the author of INLINEFORM5 ; and INLINEFORM6 . DISPLAYFORM0

ternn: This is the rnn-based method with user type embeddings added. Each user type INLINEFORM0 is mapped to a user type embedding INLINEFORM1 . The lr layer is modified as follows, where INLINEFORM2 is the embedding of the type of the author of INLINEFORM3 . DISPLAYFORM0

ubrnn: This is the rnn-based method with user biases added. Each user INLINEFORM0 of the training set with INLINEFORM1 is mapped to a user-specific bias INLINEFORM2 . Users with INLINEFORM3 are mapped to a single `unknown' user bias. The lr layer is modified as follows, where INLINEFORM4 is the

bias of the author of INLINEFORM5 . DISPLAYFORM0

We expected ubrnn to learn higher (or lower) INLINEFORM0 biases for users whose posts were frequently rejected (accepted) in the training data, biasing the system towards rejecting (accepting) their posts.

tbrnn: This is the rnn-based method with user type biases. Each user type INLINEFORM0 is mapped to a user type bias INLINEFORM1 . The lr layer is modified as follows; INLINEFORM2 is the bias of the type of the author. DISPLAYFORM0

We expected tbrnn to learn a higher INLINEFORM0 for the red user type (frequently rejected), and a lower INLINEFORM1 for the green user type (frequently accepted), with the biases of the other two types in between.

In all methods above, we use 300-dimensional word embeddings, user and user type embeddings with INLINEFORM0 dimensions, and INLINEFORM1 hidden units in the gru cells, as in our previous experiments BIBREF0 , where we tuned all hyper-parameters on 2% held-out training comments. Early stopping evaluates on the same held-out subset. User and user type embeddings are randomly initialized and updated by backpropagation. Word embeddings are initialized to the word2vec embeddings of our previous work BIBREF0 , which were pretrained on 5.2M Gazzetta comments. Out of vocabulary words, meaning words not encountered or encountered only once in the training set and/or words with no initial embeddings, are mapped (during both training and testing) to a single randomly initialized word embedding, updated by backpropagation. We use Glorot initialization BIBREF5 for other parameters, cross-entropy loss, and Adam BIBREF6 .

ubase: For a comment INLINEFORM0 authored by user INLINEFORM1 , this baseline returns the

rejection rate INLINEFORM2 of the author's training comments, if there are INLINEFORM3 training comments of INLINEFORM4 , and 0.5 otherwise. INLINEFORM5

tbase: This baseline returns the following probabilities, considering the user type INLINEFORM0 of the author. INLINEFORM1

Results and Discussion

Table TABREF15 shows the auc scores (area under roc curve) of the methods considered. Using auc allows us to compare directly to the results of our previous work BIBREF0 and the work of Wulczyn et al. Wulczyn2017. Also, auc considers performance at multiple classification thresholds INLINEFORM0 (rejecting comment INLINEFORM1 when INLINEFORM2 , for different INLINEFORM3 values), which gives a more complete picture compared to reporting precision, recall, or F-scores for a particular INLINEFORM4 only. Accuracy is not an appropriate measure here, because of class imbalance (Table TABREF5). For methods that involve random initializations (all but the baselines), the results are averaged over three repetitions; we also report the standard error across the repetitions.

User-specific information always improves our original rnn-based method (Table TABREF15), but the best results are obtained by adding user embeddings (uernn). Figure FIGREF16 visualizes the user embeddings learned by uernn. The two dimensions of Fig. FIGREF16 correspond to the two principal components of the user embeddings, obtained via pca.The colors and numeric labels reflect the rejection rates INLINEFORM0 of the corresponding users. Moving from left to right in Fig. FIGREF16 , the rejection rate increases, indicating that the user embeddings of uernn capture mostly the rejection rate INLINEFORM1 . This rate (a single scalar value per user) can also be captured by the simpler user-specific biases of ubrnn, which explains why ubrnn also performs well (second best results in Table TABREF15). Nevertheless, uernn performs better than ubrnn, suggesting that user embeddings capture

more information than just a user-specific rejection rate bias.

Three of the user types (Red, Yellow, Green) in effect also measure INLINEFORM0 , but in discretized form (three bins), which also explains why user type embeddings (ternn) also perform well (third best method). The performance of tbrnn is close to that of ternn, suggesting again that most of the information captured by user type embeddings can also be captured by simpler scalar user-type-specific biases. The user type biases INLINEFORM1 learned by tbrnn are shown in Table TABREF18 . The bias of the Red type is the largest, the bias of the Green type is the smallest, and the biases of the Unknown and Yellow types are in between, as expected (Section SECREF3). The same observations hold for the average user-specific biases INLINEFORM2 learned by ubrnn (Table TABREF18).

Overall, Table TABREF15 indicates that user-specific information (uernn, ubrnn) is better than user-type information (ternn, tbrnn), and that embeddings (uernn, ternn) are better than the scalar biases (ubrnn, tbrnn), though the differences are small. All the rnn-based methods outperform the two baselines (ubase, tbase), which do not consider the texts of the comments.

Let us provide a couple of examples, to illustrate the role of user-specific information. We encountered a comment saying just “Ooooh, down to Pireaus...” (translated from Greek), which the moderator had rejected, because it is the beginning of an abusive slogan. The rejection probability of rnn was only 0.34, presumably because there are no clearly abusive expressions in the comment, but the rejection probability of uernn was 0.72, because the author had a very high rejection rate. On the other hand, another comment said “Indeed, I know nothing about the filth of Greek soccer.” (translated, apparently not a sarcastic comment). The original rnn method marginally rejected the comment (rejection probability 0.57), presumably because of the `filth' (comments talking about the filth of some sport or championship are often rejected), but uernn gave it a very low rejection probability (0.15), because the author of the comment had a very low rejection rate.

Related work

In previous work BIBREF0 , we showed that our rnn-based method outperforms detox BIBREF1 , the previous state of the art in user content moderation. detox uses character or word INLINEFORM0 -gram features, no user-specific information, and an lr or mlp classifier. Other related work on abusive content moderation was reviewed extensively in our previous work BIBREF0 . Here we focus on previous work that considered user-specific features and user embeddings.

Dadvar et al. Dadvar2013 detect cyberbullying in YouTube comments, using an svm and features examining the content of each comment (e.g., second person pronouns followed by profane words, common bullying words), but also the profile and history of the author of the comment (e.g., age, frequency of profane words in past posts). Waseem et al. Waseem2016 detect hate speech tweets. Their best method is an lr classifier, with character INLINEFORM0 -grams and a feature indicating the gender of the author; adding the location of the author did not help.

Cheng et al. Cheng2015 predict which users will be banned from on-line communities. Their best system uses a Random Forest or lr classifier, with features examining the average readability and sentiment of each user's past posts, the past activity of each user (e.g., number of posts daily, proportion of posts that are replies), and the reactions of the community to the past actions of each user (e.g., up-votes, number of posts rejected). Lee et al. Lee2014 and Napoles et al. Napoles2017b include similar user-specific features in classifiers intended to detect high quality on-line discussions.

Amir et al. Amir2016 detect sarcasm in tweets. Their best system uses a word-based Convolutional Neural Network (cnn). The feature vector produced by the cnn (representing the content of the tweet) is concatenated with the user embedding of the author, and passed on to an mlp that classifies the tweet as sarcastic or not. This method outperforms a previous state of the art sarcasm detection method BIBREF8

that relies on an lr classifier with hand-crafted content and user-specific features. We use an rnn instead of a cnn, and we feed the comment and user embeddings to a simpler lr layer (Eq. EQREF10), instead of an mlp. Amir et al. discard unknown users, unlike our experiments, and consider only sarcasm, whereas moderation also involves profanity, hate speech, bullying, threats etc.

User embeddings have also been used in: conversational agents BIBREF9 ; sentiment analysis BIBREF10 ; retweet prediction BIBREF11 ; predicting which topics a user is likely to tweet about, the accounts a user may want to follow, and the age, gender, political affiliation of Twitter users BIBREF12 .

Our previous work BIBREF0 also discussed how machine learning can be used in semi-automatic moderation, by letting moderators focus on `difficult' comments and automatically handling comments that are easier to accept or reject. In more recent work BIBREF13 we also explored how an attention mechanism can be used to highlight possibly abusive words or phrases when showing `difficult' comments to moderators.

Conclusions

Experimenting with a dataset of approx. 1.6M user comments from a Greek sports news portal, we explored how a state of the art rnn-based moderation method can be improved by adding user embeddings, user type embeddings, user biases, or user type biases. We observed improvements in all cases, but user embeddings were the best.

We plan to compare uernn to cnn-based methods that employ user embeddings BIBREF14 , after replacing the lr layer of uernn by an mlp to allow non-linear combinations of comment and user embeddings.

Acknowledgments

This work was funded by Google's Digital News Initiative (project ml2p, contract 362826). We are grateful to Gazzetta for the data they provided. We also thank Gazzetta's moderators for their feedback, insights, and advice.