TOXIC COMMENT CLASSIFICATION

ABHISHEK KANIKE abkanike@iu.edu
PREETHAM KOWSHIK pkowshik@iu.edu
ROSHITH RAGHAVAN roragh@iu.edu

OBJECTIVES AND SIGNIFICANCE

With rise of the Internet and social media platform, discussions are increasingly moving towards online forums. Often social media platforms are criticized for inefficiently regulating the comments and for facilitating cyber bullying. Regulating huge amount of comments that are generated everyday manually is a daunting task for any social media platform. Although these platforms enable their users to report comments, many of them are looking for automating these processes.

The aim of the project is to predict the toxicity of a sentence. On daily basis, we receive lot of short text information in online communication, e-commerce portals and through various digital devices. This situation contributes to the quality of human life, unfortunately it involves great dangers, since online texts with high toxicity can cause personal attacks, online harassment and bullying behaviours. This has provoked both industrial and research community in last few years with several attempts to identify an efficient model for toxic comment prediction. Automating the process of detecting toxic comments will not only help online forums but also help user safety as well as enhance discussions online. While these steps are promising new approaches and frameworks are still required. In additions the proliferation of data on a daily basis makes the development of new machine learning computational tools for managing this huge data, an imperative need.

This however is a double edged sword. On one hand, social media companies do not want to be seen as curtailing free speech and on the other hand, not acting on abusive online behaviour is allowing someone else to curtail their victim's right to free expression. There is a very fine thin line between the two which is not easy to find. But critical to finding this balance, is the ability to correctly classify the toxicity of a comment. Through this project we work on models that will help improve this ability. Our interest in this topic of free speech and our interest in language processing techniques is what motivated us to take this particular topic up.

BACKGROUND

The majority of popular social media use simple lexicon-based approach to filter offensive contents. Their lexicons are either predefined (such as Youtube) or composed by the users themselves (such as Facebook). Furthermore, most sites rely on users to report offensive contents to take actions. Because of their use of simple lexicon-based automatic filtering approach to block the offensive words and sentences, these systems have low accuracy and may generate many false positive alerts. In addition, when these systems depend on users and administrators to detect and report offensive contents, they often fail to take actions in a timely fashion.

Yin et al. (2009) applied machine learning techniques to detect harassment online. Using a combination of term frequency–inverse document frequency(TF/IDF), sentiment and contextual features, they were able to achieve the f-score of 0.481. Dinakar et al. (2011) performed ML experiments on Youtube comments trying to detect cyber-bullying. They suggest further research to incorporate more types of features, such as user information, as well as improving on the ones used in their research. They find that binary classifiers for each individual topic (sexuality, race, intelligence) perform better than multiclass classifiers. Dadvar et al. (2012) investigated whether gender information improves the accuracy detecting cyber-bullying. Their result shows that classification accuracy can improve by incorporating gender information. They showed that showed how user-based features such as history of user activity, frequency of profanity in previous comments by the user can improve the accuracy . The work presented by Nobata et al. (2016) is centered around finding the state-of-art method for detecting abusive user content online. They train a regression machine learning algorithm with years of labeled user comments from Yahoo news. Mehdad and Tetreault (2016) used the same dataset as that of Nobata et al. and managed to improve the f-score by simply using a character n-gram model (1...5) and a support vector machine with Naive Bayes (NB) features as classifier. They used a Support vector Machine (SVM) variant using NB log-count ratios as feature values which consistently performed well across different tasks and data sets.

Built by Google and Jigsaw, Perspective API uses machine learning models to score the perceived impact a comment will have on a conversation. The purpose of research here is to help increase participation, quality and empathy in online conversations. Perspective API's toxicity model allows text to be scored based on its similarity to comments people have said are toxic or likely to make then leave a conversation.

Perspective API's model however makes errors as it does not capture toxic comments not seen previously. There are also false positives when a non-toxic comment contains similar pattern to a previously flagged toxic comment. The aim is to improve the robustness of Perspective API by employing various techniques. The training dataset consists of comments from Wikipedia which have been crowd-evaluated for toxicity. One of the primary sources of bias in the existing model is imbalance in dataset caused by the fact that frequently attacked identities are over-represented in the toxic comments. For example, comments with the terms - gay, lesbian and transgender earn a high toxicity score even if they were used in a non-toxic context. This is being handled by strategically adding data. 4,620 article snippets are selected to balance toxicity distribution for specific terms.

Recently launched, the objective of Perspective system is to detect insults, threats and other forms of abusive speech and to provide a safe environment for online conversations. The API allows for the identification and filtering of toxic speech and to this end Google Jigsaw has partnered with Wikipedia and New York Times. Google and Jigsaw developed the measurement tool by taking millions of comments from different publishers and then asking panels of ten people to rate the comments on a scale from "very toxic" to "very healthy" contribution. The resulting judgments provided a large set of training examples for the machine learning model. The API allows for real time reporting of toxicity of comments while it is being typed. Although not public available, Google alludes to the use of convolutional and/or recurrent neural networks in their model. A key challenge in machine learning algorithms such as these is that an adversary could modify text just enough to not get flagged as toxic by altering spellings or masking specific letters or adding punctuation between the letters. The system is also particularly susceptible to false alarms i.e. wrongly assigning high toxicity score to benign phrases.

We created several models that negate these susceptibilities in Perspective API. We first scraped comments from reddit to create a larger dataset. We then engineered a few features such as capitalization ratio, length of comment, count of exclamation marks and count of question marks. Using Wikipedia API or from scraped data we determined if the commenter was anonymous. We added this to our features because we believe toxic comments are more often than not made by anonymous social media users.

We determined how much these features correlated, either positively or negatively, with toxicity. We explored spell checking features of NLTK to capture masked swear words in the comments. For swear words in our training dataset we created regular expressions check to check for the existence of these words in the comments. Next we used NLTK to perform part of speech tagging on the comments and

tried to find correlation between number of nouns, verbs and adjectives with toxicity of comment. Where such a correlation was found, a new feature was generated.

We then ran models such as Logistic Regression, Naïve Bayes, Naïve Bayes-Logistic Regression, Convoluted Neural Network, CNN with LSTM and finally GRU to improve the accuracy of toxic comment classification.

METHODS

Data is obtained from the large number of comments from Wikipedia's talk page edits. These comments are manually labelled by approximately 10 annotators via Crowdflower on a spectrum of how toxic the comment is to how healthy the conversation is. The resulting judgments provided a large set of training examples for the machine learning model. This data was taken from Kaggle Toxic comment challenge in the form of csv file which consists of comment id, comment_text, and binary class labels viz, toxic, severe toxic, obscene, threat, insult and identity hate.

In addition to this we extracted data from figshare dataset. Two tables from this dataset were joined, i.e, Toxicity_annotated_comments.tsv is joined with toxicity_annotated_comments.tsv on rev_id. Columns extracted are rev_id (article revision id), toxicity_score (score on scale of -2 severe toxic to 2 toxic), comment (body). Using the toxicity_score we increased severe toxic and toxic class data by considering -2 toxic score and -1 toxic score respectively.

We also extracted user demographics information using Quarry query tool on Wikipedia. Comments table is queried on comment_ids and following user information was obtained -

gender: Gender of the user (Male or Female)

anonymous: 1 if the comment is by anonymous user else 0.

Also, data was scraped by making a urllib.request on hot topics such as 2016 elections, Roy Moore's selection as candidate in Alabama etcetra and parsed comments using regular expressions . We then proceeded to label them as per the following rules -

- For identity_hate: If the comments consists a mention of nationalities, religion, sexual orientation, transgender etc
- For threat: If the comments contains exclamation mark(!) and if the comment belongs to toxic class which uses multiple 'you'.

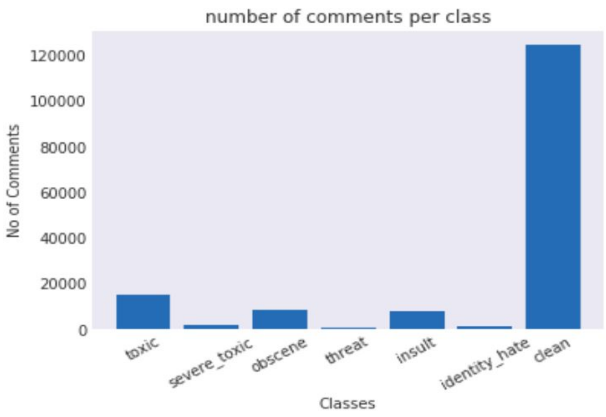

Figure 1

Shows class distribution of various types of categories

Data Cleaning

In order to achieve better insights or to build better algorithms, it is necessary to play with clean data. Social media data is know to be highly unstructured consisting of typos, bad grammar, slangs, and also consists of unwanted data such as URL, userids and so on. Initially we removed URLs and userids present in the corpus since urls userids and IP addresses are not part of standard english dictionary. We created a dictionary know as Apostrophe lookup which is used to replace words consisting of apostrophe with the actual word. For example words such as "we'll" are converted to "we will". We also used regular expressions to match vulgar words which are frequently misspelled and were replaced with actual word. Additionally we also used NLP techniques such as Lemmatization and Stemming to map derived words to their word. For example words such as "running", "ran" are mapped to run. These transformations made sure that words were consistent across the corpus and would augment the text transformation techniques. Additionally we converted the comments to lower case and removed unwanted characters and extra spaces and filled NAN values with empty string.

Feature Engineering

Additional Features added:

1. *length_of_comments:* Assumption is that hateful or toxic comments tend to be shorter on average than non-toxic genuine comments.

2. *capitals:* Number of capitalized letters in comments. Toxic comments tend have more capitalization
3. *capitalization_ratio:* Ratio of capitals and lenght_of_comments. This is expected to be higher for toxic comments.
4. *count_exclamation:* Assumption is that toxic comments tend to have more exclamation than non-toxic comments
5. *count_questions:* Assumption is that toxic comments have fewer question marks than non-toxic comments.
6. *symbols:* Symbols such as * and $ are frequently used in conjunction with toxic comments.
7. *count_nouns:* Number of nouns in the comment text
8. *count_verbs:* Number of verbs in the comment text
9. *count_adjectives:* Number of adjectives in the comment text
10. *Male:* Identify gender of commenter if commenting after logging in
11. *Female:* Identify gender of commenter if commenting after logging in

Text Transformations

Term Frequency-Inverse Document Frequency(TF-IDF): TF-IDF is a statistical technique that tells about the importance of the a word in the document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. TF-IDF consists of two parts.

Term Frequency - number of times a word occurs in a collection or in a corpus.

Inverse Document Frequency - number of documents in a collection that contains that particular term.

One of the disadvantages with Term frequency is that words that appear in most of the documents have high weightage and these might not give proper results for document comparison. For example the word auto will be present in most of the documents in say automobile industry. The word auto doesn't say anything about the similarity of the document since it is present across entire corpus. By TF with the IDF value, these words are given lower importance. We can think of IDF as a regularization parameter. The mathematical formula for TF-IDF is given below.

$$x_{ij} = m_{i_j}/m_i \times log(n/n_j)$$

where m is the number of terms and n is the number of documents in the corpus . m_{ij} is the number of times that the j-th term appears in the i-th document, m_i is the number of terms in the i-th document and n_j is the number of documents containing the j-th term in the data set.

Word embeddings - is a collection of language modelling and feature learning technique in NLP where words of sentences are mapped to vectors of numerical values. These numerical values aims to quantify the semantic similarities of a word based on their distribution in a large samples of data. For our project we have used two pre trained word embedding models

1. FastText - Fastest is a library created by Facebook which consisting of word embeddings of words which are calculated using an hierarchical classifier. Every word is represented as vector of size 300.
2. Glovec - is an unsupervised learning algorithm for obtaining vector representation of words. The GloVe model is trained on the non-zero entries of a global word-word co-occurrence matrix, which tabulates how frequently words co-occur with one another in a given corpus. Pre-trained Glove embeddings model provides vector of different sizes but vector size of 100 is chosen for this project.

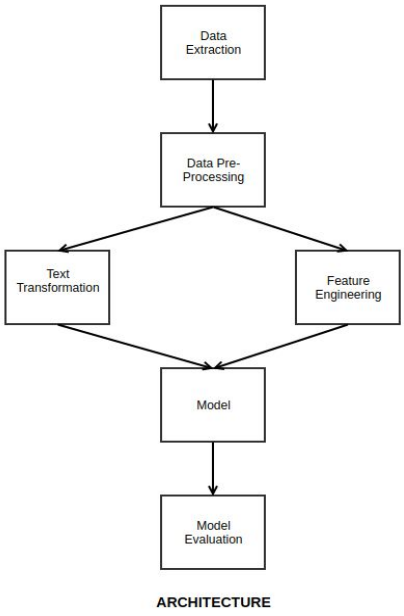

Figure 2

Project Architecture

Models

Logistic Regression

The categorization of a comment into toxic, severe_toxic, obscene, threat, insult, identity_hate is largely a probabilistic process and therefore can be modelled well using logistic regression. After merging the train and text comments and filling na for empty cells, the dataset is vectorized using TfIdf.

Initially the logit model was used to predict on train dataset and it's ROC AUC was computed. The features used initially were the binary variables toxic, severe_toxic, obscene, threat, insult, identity_hate. Additional engineered features were included in the model after checking Akaike Information Criterion and Bayesian Information criterion. These are measures used for logistic model comparisons. The lower the values of AIC and BIC the more preferred the model is. The probability that each record should be classified to a particular class is computed. One major advantage of logistic regression is that we can convert coefficients of independent variables to odds ratios by exponentiating them. We can then interpret the effect of the variable in terms of its effect on the odds. Additionally, unlike ROC-AUC, log loss is highly sensitive to class imbalance. Refer to the results section for discussion on model accuracy.

Naive Bayes + Logistic Regression

Sida Wang and Christopher D. Manning in their paper Baselines and Bigrams: Simple, Good Sentiment and Topic Classification demonstrate that bag of features models are still strong performers on snippet sentiment classification tasks. We create bag of words matrices and combine this approach with the logistic regression from the previous part. The main model variants are formulated as linear classifiers. The prediction for test case k becomes

$$y^k = sign(w^T x^k + b)$$

Here we combine generative and discriminative classifiers to create a simple model variant where a logit is built over NB log-count ratios as feature values. The logistic regression is implemented for l2 penalty with liblinear solver. C parameter is set to 4, to indicate weaker regularization in the regression. Refer to the results section for discussion on model accuracy. The results show how a reasonable prior work well with regularization.

CNN (Convolutional Neural Network)

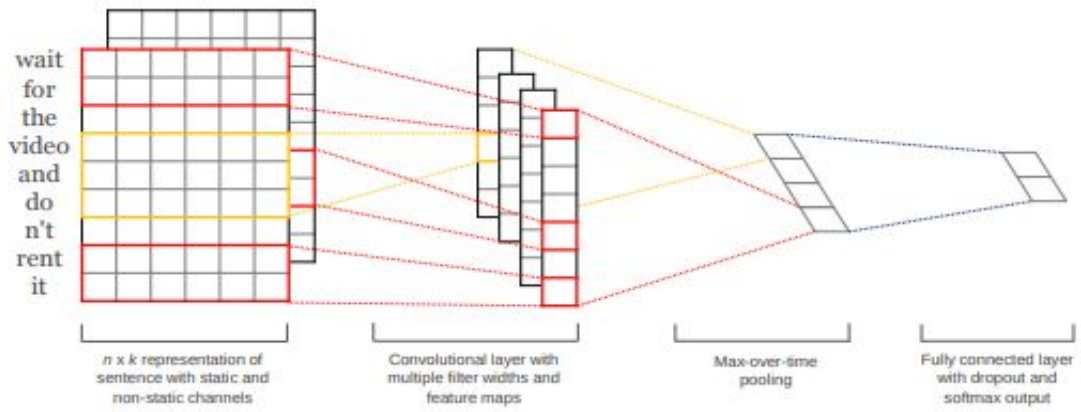

Figure 3

CNN workflow

Convolutional Neural Networks has the ability to learn complex feature representations. Hence, we are applying a CNN-based approach to categorization of text fragments, at a sentence level, based on the prominent semantics. Similar to the approach outlined by *Yoon-Kim* in *'Convolutional Neural Networks for Sentence Classification',* we converted each sentence into a word level matrix where each row is a sentence vector extracted from fastText embeddings. CNNs require input to be of fixed size and the sentence length varies greatly. Therefore, a statistical metric equal to mean + 1 standard deviation (over the sentence length distribution which covers 86% (150) of sentences without truncation) is set as the static sentence length for the network. 20% of 1D features are dropped out in the first step reducing the trainable parameters for the network using SpatialDropout. SpatialDropout regulates independence between the features by dropping the correlated features (words).

300 convolution filters of 4-gram model filters with rectified linear activation function i.e, 300 random convolution filters of 4 window size (four words) is used as first layer feature learning which emits 147 feature maps. A global max pooling is performed that selects a maximum of 147 filter value for each of 300 dimensions. Also a global max pooling on pre-convolution step data is also done which selects a maximum out of 150 for each of 300 dimension. Alongwith above two outputs, features extracted from feature engineering step are concatenated to giving a 608 dimension vector. Regularization is done after this step to prevent overfitting by dropping 30% of random values. This layer is given to a dense layer of 6 neurons to emit each class probability with sigmoid activation function. Output of this model are tabulated in Results section

Convolutional Neural Network + Long Short Term Memory

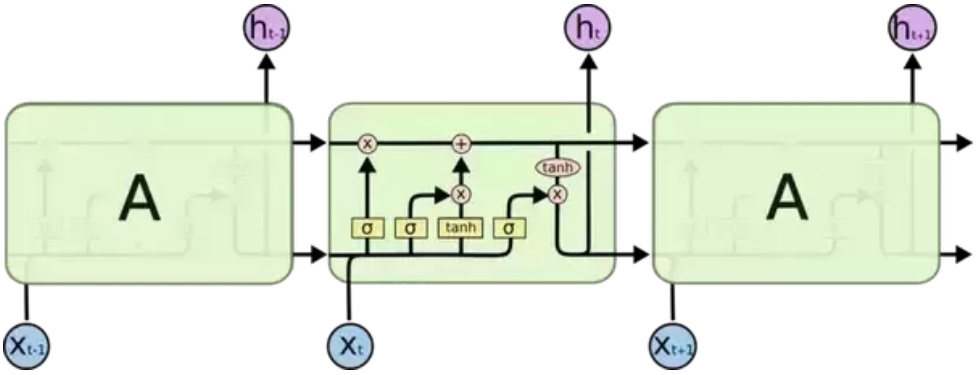

Figure 4

LSTM Units

For sequence modeling such as this task, the traditional recurrent neural network process a sequence of arbitrary length by recursively applying a transition function to its internal hidden state vector of the input sequence. Using RNN, a simple strategy for this is to embed word vectors(fastText) and then to feed the vector to a softmax layer for classification. But the disadvantages of Recurrent Neural Network is that its in ability to learn semantic relationships between words which are far apart in a given sentence. Consider this example " I grew up in France …. I speak French". Let say we are trying to predict the word French. Recent information says that the next word should be a language . Since we a word France we can predict the word as French. However RNN fails to capture this context since these are far apart in the sentence. Hochreiter and Schmidhuber in there paper propose Long Short Term Memory which overcomes the exploding or vanishing gradient problem of RNNs. LSTM maintains a separate memory cell inside it that updates and exposes its content only when deemed necessary. We combined CNN and LSTM, where CNN extracts feature maps from the embeddings. This is fed to LSTM as a sequence model to predict the probabilities of the label class.

Similar to above CNN model, input layer is an embedding layer of length 150 with 300 dimension. Over this, 300 random convolution filters of 4 window size (4 words) with rectified linear activation function are applied. This results in an output of 147 features of 300 dimension each. An LSTM layer of 50 neurons is applied to the output of previous layer. The feature list is reversed and applied the same LSTM layer functionality with 50 neurons (mimicking the biredictionlity function). This reduces the feature dimensionality to 100. After this an average pooling over this 147 vectors of 100 dimension is done.

With above output, 8 features from exploratory analysis step are combined and fed to a 30% dropout layer to maintain regularization which prevents overfitting of data. The regularized data is connected to dense 6

neuron layer to emit the probabilities of each as a sigmoid activation function. Performance of the model is mentioned in Results section.

Gated Recurrent Unit (GRU)

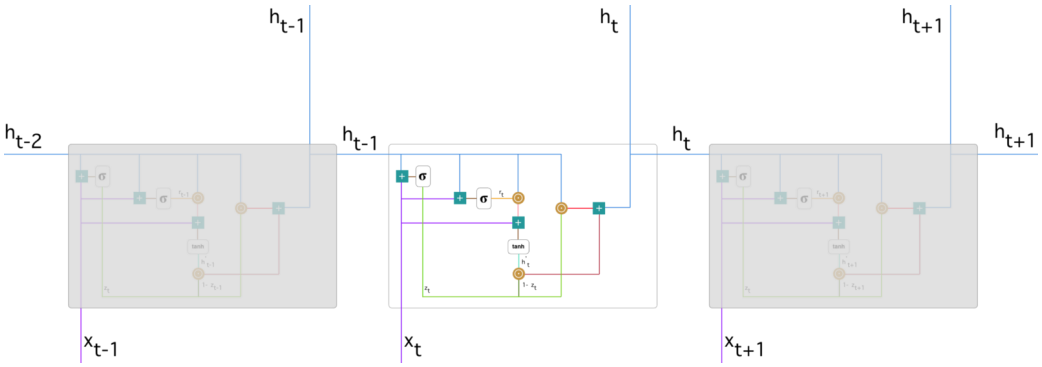

Figure 4

GRU units

Gated Recurrent Unit (GRU) uses a similar technique as that of LSTM but uses only two gates, the update gate and the reset gate. The update gates control the amount of previous information flows to the future and reset gate is used to determine how much past information is to be forgotten. We have assumed the maximum length of each comment to 100 words and the input layer is initialized with 100 neurons. The Keras embedding layer is feed with embedding weight matrix which consists of word embeddings from GloVe model. We have initially added a spatial dropout layer in order to promote any independence among feature maps generated by embedding layer. A bidirectional layer with GRU for every cell is created which does similar functionality as in LSTM but with GRU gates. Two pooling layers to reduce the dimensions were used and a finally a dense layer of 6 neurons at the end.

Blending

A blend of above two deep learning methods (CNN+LSTM and LSTM+GRU) was done by taking weighted average of probabilities generated. Weight of each technique was assigned as the ROC AUC score. The results are tabulated in the below Results section.

RESULTS

Feature Selection:

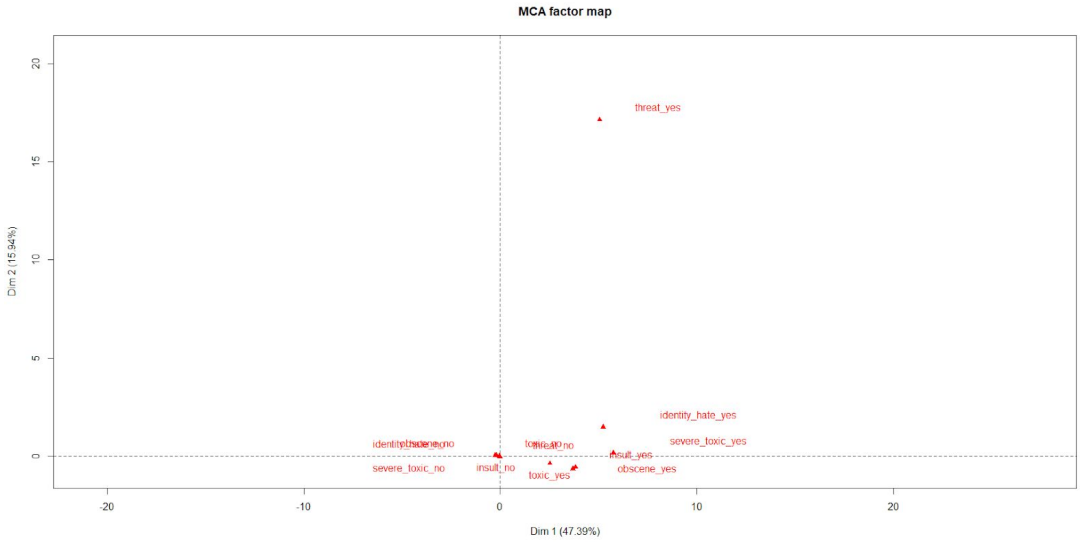

Figure 5

Correspondence Analysis showing the correlation between different categories of comments.

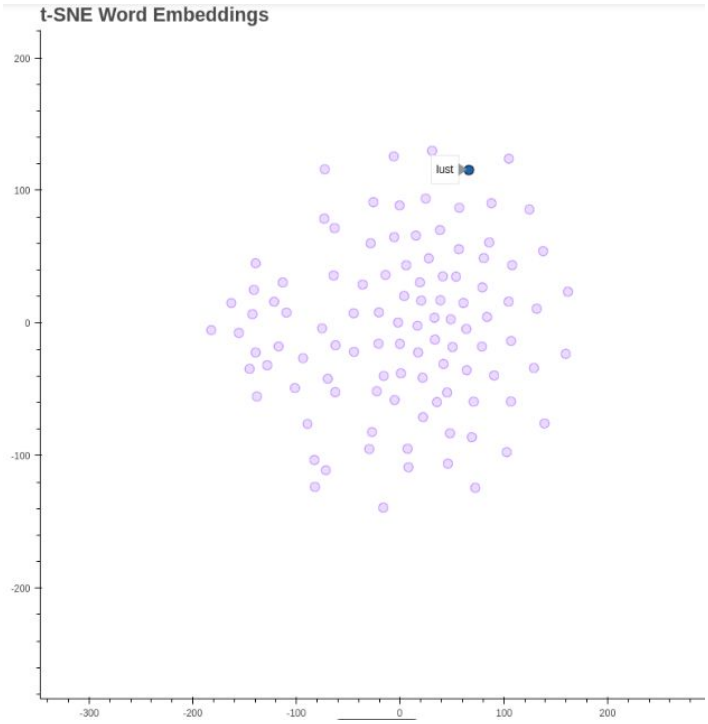

Figure 6

Word embeddings of 104 unique swear words

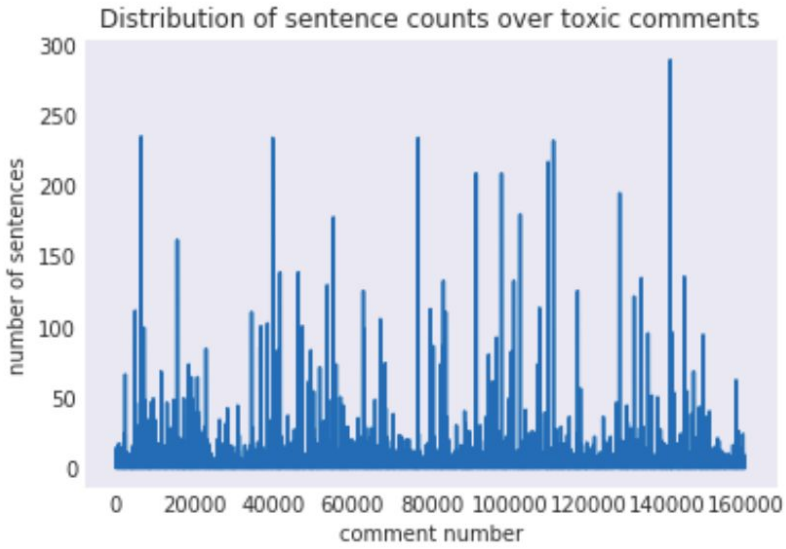

Figure 7

Distribution of sentence counts over toxic comments

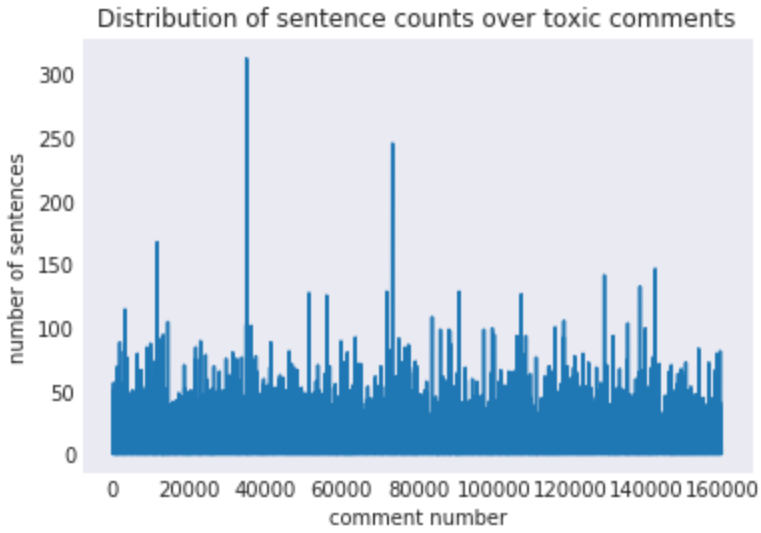

Figure 8

Distribution of sentence counts over toxic comments

The plots above shows the distribution of the number of sentences across various comments. Our initial assumption that the toxic comments have higher number of sentences was false. We see that the plots for

both clean as well as toxic comments with respect to sentence length was almost similar. Thus we choose to calculate the length of comments as number of characters in the entire comment.

Correlation matrix:

	identity_hate	insult	obscene	severe_toxic	threat	toxic
length_of_comments	-0.013670	-0.045128	-0.043010	0.010021	-0.007993	-0.054582
capitals	0.053464	0.075581	0.081326	0.143032	0.033631	0.090645
capitalization_ratio	0.093529	0.169632	0.181673	0.168975	0.055653	0.219709
count_exclamation	0.006005	0.027010	0.024362	0.060578	0.034202	0.037337
count_questions	-0.000516	0.004859	0.005404	-0.004541	-0.003949	0.027820
symbols	0.020551	0.022631	0.028426	0.031005	0.005205	0.034040
count_nouns	0.007451	-0.004308	0.000369	0.066341	0.002891	-0.002960
count_verb	-0.021038	-0.051585	-0.051647	-0.014499	-0.004137	-0.061246
count_adjective	-0.005014	-0.036856	-0.039865	0.001506	-0.016169	-0.051500
male	-0.031025	-0.077413	-0.077056	-0.036360	-0.020180	-0.095154
female	-0.000600	-0.002019	-0.002633	-0.001426	-0.000739	-0.002380

Table 1

Correlation matrix showing correlation between features and classes

The strong features are:

1. capitalization_ratio is strongly positively correlated with toxicity.
2. count_exclamation is positively correlated with toxicity.
3. count_questions and length_of_comment are negatively correlated with toxicity.
4. count_nouns is correlated with severe_toxic comments.

Model Performance:

The model performance is presented on the basis of four experiments- ROC area under the curve for predictions on training dataset with original set of features, ROC AUC on training dataset with additional features added, ROC AUC for prediction on test with original features and ROC AUC for prediction on test dataset with engineered features included. These experiments allow us to track which models perform better, overfit more, and the impact of features included.

Logistic Regression

Basic logistic regression model was built to predict the class that each record belonged to.

Dataset Specification	ROC AUC
Original Training dataset	0.9903
Training dataset with engineered features	0.9910
Test dataset with original feature set	0.9721
Test dataset with engineered features included	0.9739

Table 2

Performance results for Logistic regression model

Naive Bayes + Logistic Regression

Dataset Specification	ROC AUC
Original Training dataset	0.9924
Training dataset with engineered features	0.9930
Test dataset with original feature set	0.9747
Test dataset with engineered features included	0.9761

Table 3

Performance results for Naive Bayes combined with Logistic regression model

Convolutional Neural Network

Dataset Specification	ROC AUC
Original Training dataset	0.9873
Training dataset with engineered features	0.9885
Test dataset with original feature set	0.9747
Test dataset with engineered features included	0.9779

Table 4

Performance results for CNN

Convolutional Neural Network + Long Short Term (LSTM)

Dataset Specification	ROC AUC
Original Training dataset	0.9815
Training dataset with engineered features	0.9867
Test dataset with original feature set	0.9798
Test dataset with engineered features included	0.9811

Table 5

Performance results for combination of CNN and LSTM

Gated Recurrent Network (GRU)

Dataset Specification	ROC AUC
Original Training dataset	0.984
Training dataset with engineered features	0.986
Test dataset with original feature set	0.9807
Test dataset with engineered features included	0.9819

Table 6

Performance results for GRU

As expected, additional features that we engineered improved the performance of all the models. The baseline models seem to overfit the training set more than the deep learning models which outperform the latter in predicting the classes of test dataset.

CONCLUSION

Our baseline models are based on logistic regression and naive bayes and a combination of the two. We then proceed to employ deep learning techniques. Further, we create an ensemble of CNN and GRU and we find that this gives the best result of all our models. CNN gave us a ROC AUC of 0.9783 whereas GRU gave us a ROC AUC of 0.981. The combination of these two gave an ROC AUC of 0.983. We expected our best model to be a blend of these techniques as some feature probabilities are better studied by some techniques than others.

Several deep learning techniques such as RNN, LSTM and GRU are built for NLP and therefore it is not surprising that these techniques outperform our baseline models just as we had expected.

The following steps could be implemented to further improve our study:

1. Create a survey to rate toxicity of our newly acquired datasets
2. Experiment with word2vec and topic modelling techniques.
3. Collect more data to improve performance of investigated models.
4. Reduce the error from non-convexity in DL results. Use bagging and cross-validation more rigorously using GPUs to handle the computational complexity.
5. Instead of using pre trained model for word embeddings we could try to initialize the embedding layer with random weights and allow the network to learn these embedding values. However, an overhead with this approach is that it could increase the training time since number of parameters to learn will increase.
6. We could also try increasing the vector size of the each word embeddings . Presently vector dimension of 100 is used Glove Embedding we can extended it to 300 dimension.

INDIVIDUAL TASKS

I worked on collecting data from figshare and MediaWiki table, feature engineered on nlp tags using nltk and built deep learning models viz, CNN, CNN+LSTM with fastText embeddings. I also attempted a blend of CNN+LSTM with LSTM+GRU. My teammate Roshith worked scraping data from Reddit, feature engineering and built a logistic regression model along with naive-bayes + logistic regression model. Preetham Kowshik has done word embeddings on GloVe, cleaned data and performed exploratory data analysis. He also built a gated Recurrent Neural network. In addition to that, both Roshith and Preetham performed Tf-idf vectorization by tuning the parameters.

Additionally, we were all equally involved in rating toxicity after agreeing on a schema to prepare a training dataset using the additional scraped data from online discussion forums (reddit). Report preparation too was a collective effort with equal division of labour. We together attempted creating multiple blendings.

This project was not an extension of any other project and we worked from scratch solely to for the course Data Mining.

REFERENCES

[1] Hossein Hosseini, Sreeram Kannan, Baosen Zhang and Radha Poovendran, “Deceiving Google’s Perspective API Built for Detecting Toxic Comments”, arXiv preprint arXiv:1702.08138v1,2017

[2] Theodora Chu, Kylie Jue and Max Wang, “Comment Abuse Classification with Deep Learning”, 2017

[3] Sida Wang and Christopher D. Manning, “Baselines and Bigrams: Simple, Good Sentiment and Topic Classification”, 2012

[4] David J Hand, Robert J Till, “A Simple Generalization of the Area Under the ROC Curve for Multiple Class Classification Problems”, 2001

[5] Yoon Kim, “Convolutional Neural Networks for Sentence Classification”, 2014

[6] Pengfei Liu, Xipeng Qiu, Xuanjing Huang, “Recurrent Neural Network for Text Classification with Multi-Task Learning”, 2016

[7] Anton Lundborg, “Text classification of short messages”, 2017

[8] Ying Chen , Sencun Zhu, Yilu Zhou and Heng Xu, “Detecting Offensive Language in Social Media to Protect Adolescent Online Safety ”, 2012