Deep Learning Techniques in NLP for Toxic Comment Classification

A PREPRINT

Suraj Ananya Bandarupalli*
Department of Computer Science
University Of Minnesota,Twin Cities
Minnesota,Mn-55414
banda057@umn.edu

Shrivardhan Bharadwaj
Department of Computer Science
University Of Minnesota,Twin Cities
Minnesota,Mn-55414
bhara054@umn.edu

May 17, 2019

ABSTRACT

The use of social media platforms to interact with our friends and family online has become a significant part of our daily lives. The vast reach, freedom to express our thoughts and opinions and the veil of anonymity on many sites often leads to the encounter different and exciting ideas. But it is equally true that the same factors lead to people misusing the site to say toxic and hurtful things. People's lives are disturbed in various proportions due to these comments. Therefore, this side of the internet cannot be ignored. To deal with this, in this paper, we look at numerous deep learning techniques to classify toxic comments. The goal of the paper is to analyze different techniques like LSTM, bidirectional LSTM, and GRU with different hyperparameters in the context of classifying toxic comments. The best model is a GRU with Xavier embedding having a ROC score of 97.92% at the end of 5 epochs.

***K*eywords** LSTM · GRU · Word Embeddings

1 Introduction

Every day, the userbase of online forums like social media, online gaming, and other site is increasing drastically. This ease of access and the freedom of expression on these platforms leaves the users of online communities like Facebook, Reddit, and Twitter, etc often vulnerable to vulgar and toxic comments. Due to the sheer size of these platforms, it has made content moderation a tough task. There is an added risk of psychological damage to the moderators during content moderation. Due to these factors, the moderation of comment at scale on the online platforms has been extremely difficult.

According to an online survey in 2017 by the Pew Research Center, roughly 40% of online users in America were subjected to online harassment[1]. Online sites that allow user content to be posted tend to face toxic and vulgar comments posted on their site. This affects adverse users psychologically. Sites that allow no content moderation on the platform have more toxic content than sites that have some kind of moderation. An analysis of online personal attacks in large numbers shows that the majority of these attacks on social media platforms like facebook and twitter not restricted to a few individuals and is also not due to the anonymity provided to the users. It is due to the fact that toxic comments lead to more toxic comments[2]. The users of the site can be influenced to act aggressively by having an environment of toxicity. Therefore online sites containing toxic comments will lead to encouraging more toxic comments from the users. Developing an automated tool to detect toxic comments on online sites would greatly help with content moderation.

To solve this problem, we try to automate the process of content moderation to assist content moderates to maintain the sites. This problem of classification of comments is done under one of the widely researched areas in computer

*Implementation can be found in https://github.umn.edu/banda057/Toxic-Classification

science called Natural Language Processing(NLP). Sentiment classification under Text classification is an area under NLP widely researched dating back to 2000s[3]. There is a need to classify these comments in multiple categories and not just into a binary classification of allowed or blocked. Due to this, it becomes much harder to solve this problem although this task closely resembles the classic classification tasks in NLP. Toxic comment classification is multi labeled toxic classification tasks and classification ranges over six toxicity labels namely toxic, insult, severe toxic, threat, obscene and identity hate. With state of the art deep learning algorithms, classification of a comment as toxic and non-toxic can be easily attained. But the classification into subcategories mentioned above is a more complex as a well-defined boundary does not exist between these labels.

In this paper, we describe a baseline approach to solve the multilabel classification problem using Naive Bayes SVM technique with word embeddings using word-vector embeddings. We compare the non-neural baseline to three different deep learning models to classify the sentences. Long Short Term Memory(LSTM), Bidirectional Long Short Term Memory(BiLSTM), Gated recurrent Unit(GRU) were explored and compared to check which model achieves the best results. Different embeddings like fasttext, word2vec, glove and Xavier were used to compare their performance with different models.

2 Review of related work

Toxicity classification is an essential field in NLP especially with the rise of cyberbullying and political hate speech. The comments have to be classified into one of six categories. Many text classification techniques use convolutional neural networks(CNN) and recurrent neural networks(RNN). The same set of operations are repeated for all words in an RNN but the drawback of RNNs is not efficient. Other variations like LSTM, BiLSTM, and GRU are used to solve problems in NLP.

A non-neural approach to toxic comment detection involved an NLP regression classifier to the n-gram, linguistic, syntactic, and semantic features of comments posted on Yahoo! Finance, and News[4]. E. Wulczyn et.al proposed machine learning(ML) approaches with combinations of word or character n-grams, logistic or multilayer perceptron models with a large corpus of human ratings on Wikipedia comments[5]. A summary of common features of online abuse detection used for non-neural and early neural approaches was proposed by A. Schmidt et.al[6]

One of the earliest works in the field of machine learning based detection of online harassment was proposed by Yin et.al[7], where content, sentiments and contextual features was taken as input into a support vector machine(SVM). Collobert and Weston proposed a general CNN architecture to many NLP applications like Name Entity Recognition. In 2013, Kim proposed a simple CNN model with hyperparameter tuning which performs very well in text tasks. Wang and et.al proposed the use of LSTM for predicting polarity in tweets in 2015[8]. Zhou et.al proposed how a bidirectional LSTM with max pooling which improves the accuracy of text classification[9].

A model for sentiment label distribution that involved a combination of using a Deep CNN for character-level embeddings (to increase information of word-level embedding), with a BiLSTM to produce sentence wide feature representations for word-level embeddings was a proposed by Nyugen et.al[10]. The model achieved a prediction accuracy of 86.63% on Stanford Twitter Sentiment Corpus.

There are 2 main projects at Google and Wikipedia in the field of web toxicity.Google's Conversation AI team classifies toxic comments using Perspective API[11]. Over 100k high-quality human labeled comments and 63 million machine-labeled comments have been generated according to a joint paper by Wikipedia and Google's Jigsaw. Performance of various deep learning approaches to this problem using both word and character embedding was proposed by Chu and Jue[12]. They compared the performance of RNN with LSTM and word embeddings, CNN with character embeddings. An accuracy of 93% was achieved with CNN with character embeddings.

3 Approach

Our project explores different deep learning models to solve the toxic comment classification challenge on Kaggle.The goal of the project is to build a model that which can predict the probabilities that each comment contains each of 6 different forms of toxicity (described in more detail in the model section). As this problem comes with a large dataset of labelled comments, it helps us focus on experimenting with different models instead of spending time web scraping and labelling the data.

The objective of this project was to understand the best model to classify toxic comments. As a part of this project we explored the following models:

- A Naive Bayes SVM (Baseline)

- Recurrent Neural Networks (RNN) with Long Short Term Memory(LSTM) cells.
- Bidirectional RNN with LSTM cells.
- RNN with Gated Recurrent Units(GRU).
- Bidirectional RNN with GRU cells.

We begin with discussing the architecture of the network for the models.

3.1 Input/Output

The input and output layers are same across the models. The input is a fixed-size list of 200 words of a Wikipedia comment. Shorter comments are padded with NULL words. We chose 200 words after looking at the average number of word per comment in the training data. The output across all the models was a 6-unit layer with sigmoid activation to find the probabilities of each of the 6 types of toxicity. Batch size of 100 and learning rate of 1e-4 was selected after some trial over different options. We used the Adam Optimiser and binary cross entropy loss for training.

3.2 Word Embeddings

For extracting the best word embeddings for the model we looked at 4 different ways to do word embeddings. We set the word embedding size to 100 as this seemed to give the best results.

3.2.1 Random Word Embedding

The initial weights were initialized using Xavier Initialization. The embedding layer then trained the word embeddings from the vocabulary size of the data set to the selected size of 100.

3.2.2 GloVe Word Embedding

We experimented with twitter Glove word embeddings because they better account for misspellings and type of informal language that is used on the internet[13]. The words that matched with the words in the Glove data set were taken and others were set to random using the mean and standard deviation of the chosen Glove embeddings.

3.2.3 word2vec Word Embedding

Word2vec after its creation[14], has always been a standard word embedding in most text classification techniques. We used the google negative news word2vec vectors as the words closely resembled the words in the comments of the dataset.

3.2.4 Fast2Text character Embedding

FastText is quite different from the above 2 embeddings. While Word2Vec and GLOVE treats each word as the smallest unit to train on, FastText uses n-gram characters as the smallest unit. The biggest benefit of using FastText is that it generate better word embeddings for rare words, or even words not seen during training because the n-gram character vectors are shared with other words. This is something that Word2Vec and GLOVE cannot achieve.

3.3 Models

3.3.1 Naive Bayes SVM (Baseline)

NBSVM was introduced by Sida Wang and Chris Manning[15]. In this model, we use sklearn's logistic regression, rather than SVM, although in practice the two are nearly identical (sklearn uses the liblinear library behind the scenes).We'll started by creating a bag of words representation, as a term document matrix. We used ngrams, as suggested in the NBSVM paper. We fit the model and use it to predict on the test data.

3.3.2 LSTM

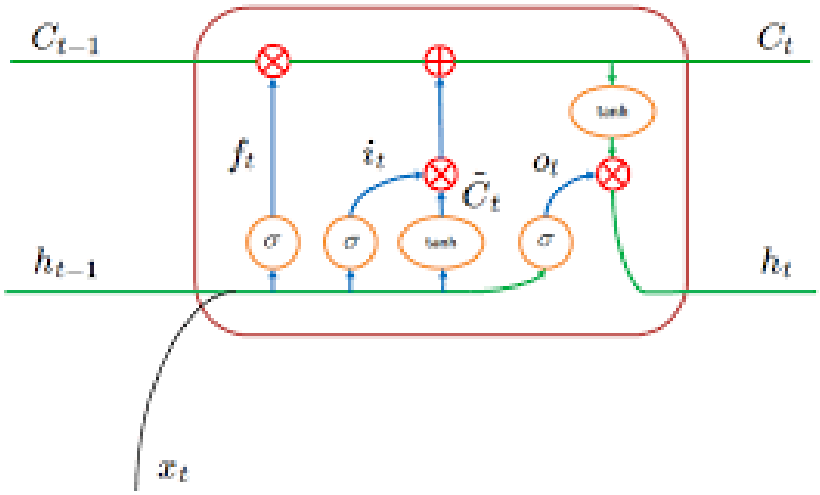
Generally sequential data problems are solved using Recurrent Neural Networks. But RNN suffer from vanishing and exploding gradient problems. This hinders in remembering the relation between terms that are far apart. Hence we used the LSTM architecture proposed by x. LSTM cells have gates that monitor the degree to keep the previous states and to memorise the extracted features from the current input[16]. The LSTM does so via input, forget, and output gates;

the input gate regulates how much of the new cell state to keep, the forget gate regulates how much of the existing memory to forget, and the output gate regulates how much of the cell state should be exposed to the next layers of the network[17]. This helps it solve the vanishing gradient problem that plagues vanilla RNNs. The word embeddings that are trained are passed into the LSTM of 60 hidden states. The 3d output tensor is passed through a max polling layer to reshape the dimension and pass it on a fully connected layer of 50 nodes. The output of this layer is fed into the final sigmoid layer for classification.

Figure 1: LSTM Formulae

$$
\begin{aligned}
i &= \sigma(x_t U^i + s_{t-1} W^i) \\
f &= \sigma(x_t U^f + s_{t-1} W^f) \\
o &= \sigma(x_t U^o + s_{t-1} W^o) \\
g &= \ tanh(x_t U^g + s_{t-1} W^g) \\
c_t &= c_{t-1} \circ f + g \circ i \\
s_t &= \tanh(c_t) \circ o
\end{aligned}
$$

Figure 2: LSTM Cell



3.3.3 Bidirectional LSTM

Bidirectional LSTM is putting together two lstms with different order of words. The input sequence is fed in normal time order for one network, and in reverse time order for another. This structure allows the networks to have both backward and forward information about the sequence at every time step. Most of the architecture is same as the LSTM model. There are 120 hidden states which monitor both forward and backward information. The output is passed into max polling layer and then into a fully connected layer of 50 nodes. Finally its passed into the sigmoid layer for classification.

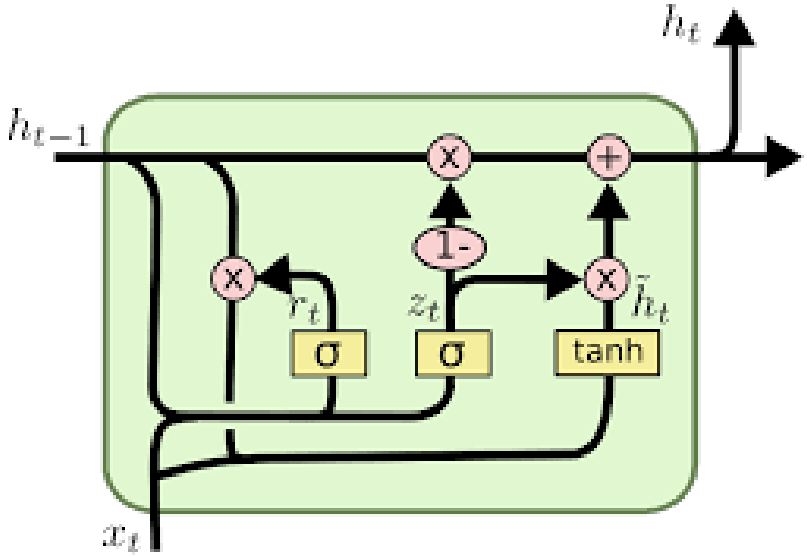
3.3.4 GRU and bidirectional GRU

Gated Recurrent Units are similar to LSTMs in most ways. The only key differences is that they contain 2 gates instead of the 3 gates in LSTM and that the GRU unit controls the flow of information like the LSTM unit, but without having to use a memory unit. It just exposes the full hidden content without any control. According to empirical evaluations in Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling[18] and An Empirical Exploration of Recurrent Network Architectures[19], there isn't a clear winner. Hence we decided to run both the models with the same configuration. Hence the GRU has the same architecture of LSTM and the bidirectional GRU has that of bidirectional LSTM.

Figure 3: GRU Formulae

$$z = \sigma(x_t U^z + s_{t-1} W^z)$$
$$r = \sigma(x_t U^r + s_{t-1} W^r)$$
$$h = tanh(x_t U^h + (s_{t-1} \circ r) W^h)$$
$$s_t = (1 - z) \circ h + z \circ s_{t-1}$$

Figure 4: GRU Cell



4 Experiment

4.1 DataSet

The dataset contains 119K Wikipedia's talk page edit comments which are annotated on the basis of their toxicity. Each comment has 6 labels which are not mutually exclusive. It means that a comment can be labelled with multiple types of toxicity. The labels are "toxic", "severe-toxic", "obscene", "threat", "insult" and "identity-hate". We used a test-set of 40K comments to evaluate the models.

4.2 Pre-Processing

The comments in the dataset were tokenised and a vocabulary was created. The size of the vocabulary was around 50k, unique word count. We applied a minimum 20 frequency filter to reduce the number of words that the model has to learn from. One drawback of this dataset, is the amount of misspelled words. We decided to not take up spell correction because it is a difficult process and there was no guarantee of a better result.

4.3 Evaluation Metric and Loss Function

We use mean Area Under the Receiver Operating Curve (mean-AUROC) across the classes to evaluate the model. The reasons we chose this metric are

- The labels in the dataset are heavily imbalanced. The "threat" and "identity-hate" labels are weak spots because their number of comments in the raw data are the fewest. In this case, accuracy is a poor measure while using high mean-AUROC scores demand both high precision and recall.
- This is also the metric used by Kaggle which helps us identify how good our models performed with respect to the competition.

We use the mean binary cross entropy loss across the training set.The formula for it is as shown in Fig5.

Figure 5: Loss Function

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

4.4 Word-Embedding

Initial testing with different word-embeddings, over all the models showed that the Glove and Xavier initialized word embeddings consistently outperformed the others.

4.5 Model Evaluation

In order to get the best models we had to tune the hyperparameters. We fixed the learning rate to 1e-4. The hyper-parameters we tuned are as below.

4.5.1 Dropout

We tested all the models with 0, 0.1, 0.3 and 0.5 dropouts through out the network. The values showed that without the dropout the model was often over fitting.

4.5.2 Recurrent Dropout

Recurrent dropout is the dropout used in the RNN layers. It performs recursive dropouts on all the nodes in an unrolled RNN while maintaining the same mask throughout all timestamps. This helps in generalizing the model without loss of temporal information. We tested the models with 0,0.1,0.3 and 0.5 recurrent dropouts.

4.5.3 Batch Size

We tested the models for 3 batch size of 32, 64 and 100. There was no decrease in mean-AUROC scores with increase in batch size. Hence we went with 100 to improve model runtime.

After tuning the hyper-paramters, we observed that the bidirectional LSTM outperformed the other models. We further experimented with increasing the bidirectional LSTM layers but got diminishing returns.

5 Results and Analysis

As shown in the values in all the below figures, across all models a dropout of 0.1 gets the highest mean-AUROC value. As the dropout is further increased, the mean-AUROC values dip. This shows that for a no dropout the model is overfitting and higher dropout values, the model is underfitting due to excess loss of information. We also checked for different dropouts for different layers. We observed that dropout for the embedding layer plays the most prominent role in change in mean-AUROC values. We think that this is due to two major reason, one it is the first layer and hence overfitting in the embedding layer propagates through-out the network and second small dorpout in embedding layer removes certain words from the sentences which helps the network better generalize.

Based on our experiments with recurrent dropouts, we see that recurrent dropout prevents overfitting in the RNN networks. It also shows that maintaining the same mask throughout the LSTM timestates solves the problem of loss of temporal information.

Based on the experiments, random word embeddings intilizaed by Xavier Initialization outperforms GloVe word embedding. We checked the number of words that are matching with different word embeddings. As shown in the figure6, all the embeddings have significantly lower matches when compared to the original vocabulary. This explains the bad performance of pre-trained word embeddings compared to training our own word embeddings.

Among the models, BiLSTM outperforms all the other models but the difference is quite low. This shows that the problem of vanishing and exploding gradient is not as significant as expected. Overall high mean-AUROC values prove that Recurrent Neural Networks are well suited to text-classification.

Figure 6: Number of words used in embeddings

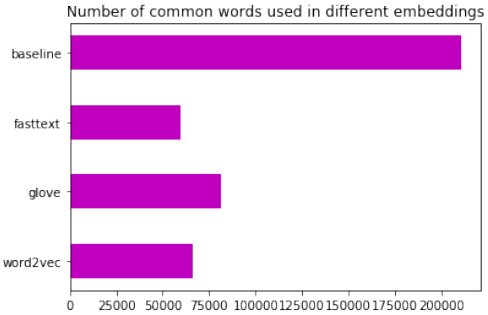


Figure 7: Mean-AUROC values for different combinations of dropouts in LSTM-glove

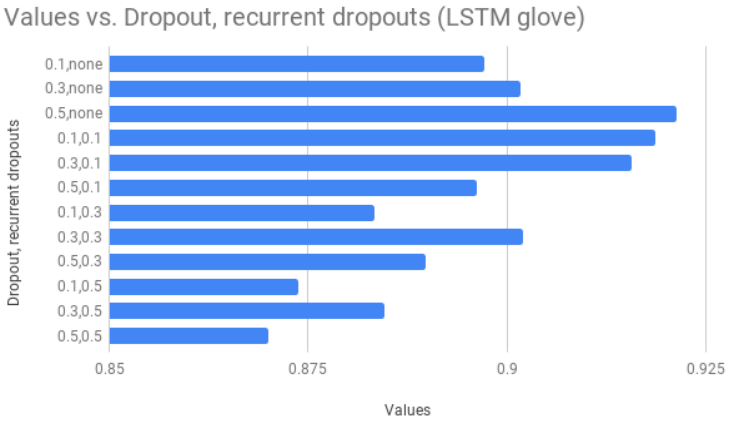


Figure 8: Mean-AUROC values for different combinations of dropouts in LSTM-xavier

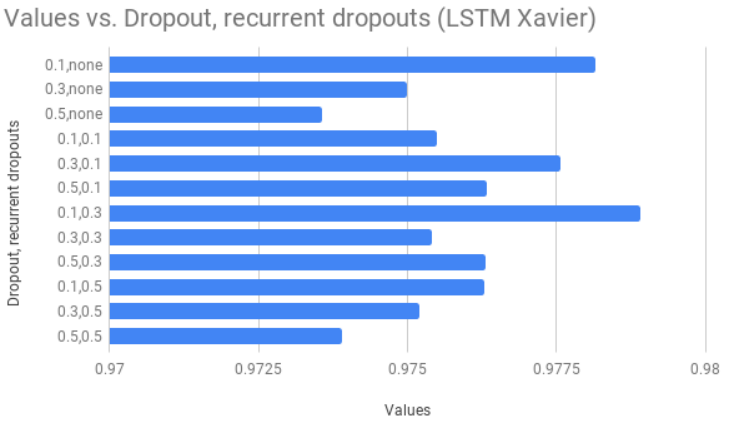


Figure 9: Mean-AUROC values for different combinations of dropouts in BiLSTM-glove

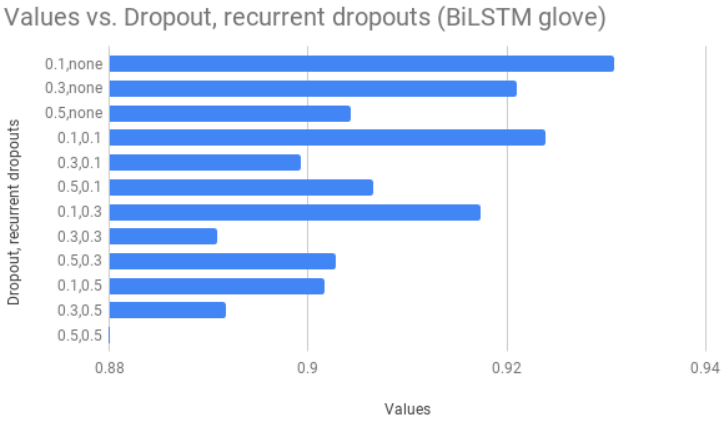


Figure 10: Mean-AUROC values for different combinations of dropouts in BiLSTM-xavier

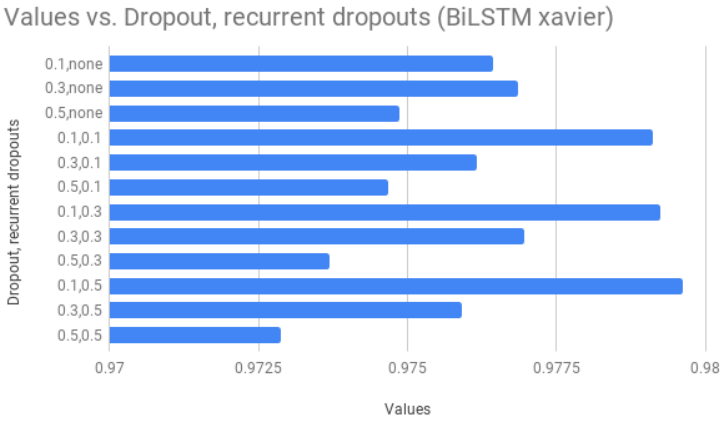


Figure 11: Mean-AUROC values for different combinations of dropouts in GRU-xavier

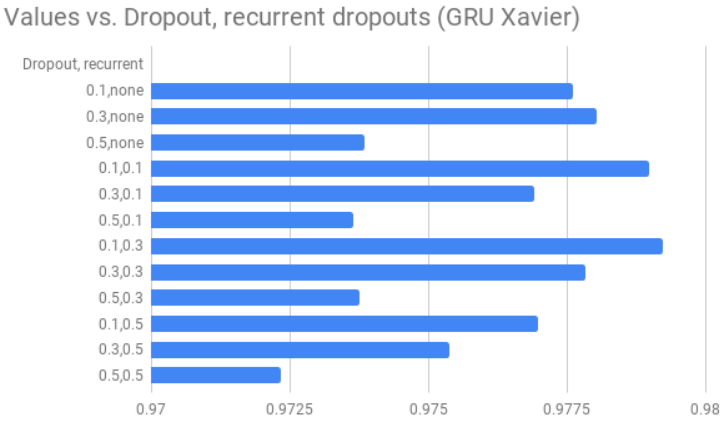


Figure 12: Mean-AUROC values for different combinations of dropouts in GRU-glove

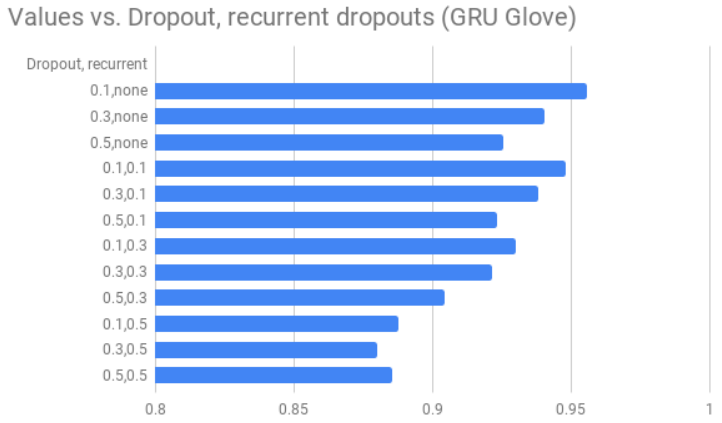


Figure 13: Comparison of top mean-AUROC values across models

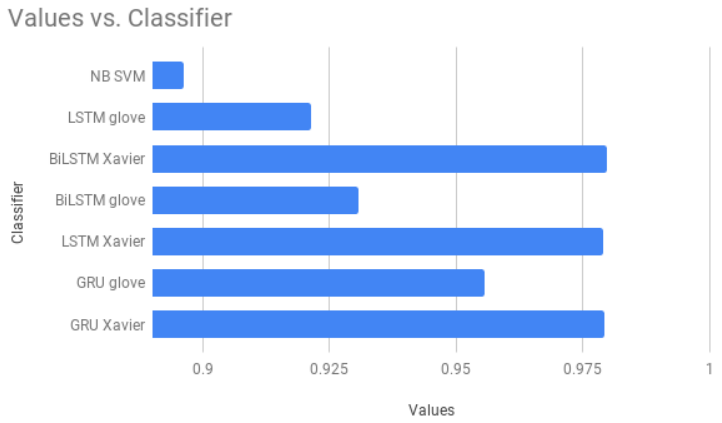


Figure 14: Top mean-AUROC values across models

Classifier	NB SVM	LSTM glove	BiLSTM Xavier	BiLSTM glove	LSTM Xavier	GRU glove	GRU Xavier
Values	0.89634	0.921305	0.979623	0.930718	0.978899	0.95557	0.979232

6 Conclusion and Future Work

Thus, we experimented with different deep learning models in NLP to classify toxic comments. As expected Bidirectional LSTM outperformed the other models but the difference was not significant. Unexpectedly Random word embedding with Xavier Initialization significantly outperformed other word embedding techniques.The huge difference between non-neural and Neural Network models show that word embeddings play a vital role in extracting latent information from text data.

Through the experimentation, it can be concluded that in order to get higher evaluation metrics, the key factor is to understand the data and improve the word-embeddings using custom rules. As future work, we will explore word embedding improvements like spell-checking and case-checking. Further experimentation can be done on the number of hidden nodes in the RNNs.

References

[1] M.Duggan, L.Rainie, A.Smith, C.Fuck, A.Lenhart and M.Madden(2014) Online harassment. Pew research center

[2] Cheng, J.; Bernstein, M. S.; Danescu-Niculescu-Mizil, C. Leskovec, J. (2017), Anyone Can Become a Troll: Causes of Trolling Behavior in Online Discussions., in Charlotte P. Lee; Steven E. Poltrock; Louise Barkhuus; Marcos Borges Wendy A. Kellogg, ed., 'CSCW' , ACM, , pp. 1217-1230 . https://files.clr3.com/papers/2017_anyone.pdf

[3] Pang, B.; Lee, L. Vaithyanathan, S. (2002), Thumbs up? Sentiment Classification Using Machine Learning Techniques, in 'emnlp2002' , pp. 79-86 . https://www.aclweb.org/anthology/W02-1011

[4] Nobata, C.; Tetreault, J. R.; Thomas, A.; Mehdad, Y. Chang, Y. (2016), Abusive Language Detection in Online User Content., in Jacqueline Bourdeau; Jim Hendler; Roger Nkambou; Ian Horrocks Ben Y. Zhao, ed., 'WWW' , ACM, , pp. 145-153 . http://www.yichang-cs.com/yahoo/WWW16_Abusivedetection.pdf

[5] Wulczyn, E.; Thain, N. Dixon, L. (2017), Ex Machina: Personal Attacks Seen at Scale., in Rick Barrett; Rick Cummings; Eugene Agichtein Evgeniy Gabrilovich, ed., 'WWW' , ACM, , pp. 1391-1399 . https://arxiv.org/abs/1610.08914

[6] Schmidt, A. Wiegand, M. (2017), A Survey on Hate Speech Detection using Natural Language Processing., in Lun-Wei Ku Cheng-Te Li, ed., 'SocialNLP@EACL' , Association for Computational Linguistics, , pp. 1-10 . https://www.aclweb.org/anthology/W17-1101

[7] Michael Baumerand Anthony Ho, Detection of Harassment on Web 2.0, Proceedings of the Content Analysis in WEB 2.0(CAW2.0) Workshop at WWW2009. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6834909.pdf

[8] Wang, X.; Liu, Y.; Sun, C.; Wang, B. Wang, X. (2015), Predicting Polarities of Tweets by Composing Word Embeddings with Long Short-Term Memory., in 'ACL (1)' , The Association for Computer Linguistics, , pp. 1343-1353. https://www.aclweb.org/anthology/P15-1130

[9] Zhou, P.; Qi, Z.; Zheng, S.; Xu, J.; Bao, H. Xu, B. (2016), 'Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling.', CoRR abs/1611.06639 . https://aclweb.org/anthology/C16-1329

[10] Nguyen, H. Nguyen, M.-L. (2017), 'A Deep Neural Architecture for Sentence-level Sentiment Classification in Twitter Social Networking' , cite arxiv:1706.08032Comment: PACLING Conference 2017, 6 pages . https://arxiv.org/abs/1706.08032

[11] Perspective API. https://www.perspectiveapi.com/

[12] Kylie J., Max L. WangPublished(2017). Comment Abuse Classification with Deep Learning. https://www.semanticscholar.org/paper/Comment-Abuse-Classification-with-Deep-Learning-Chu-Jue/30b82ff201bcacc8b3e1dbecfd79308f3137706f

[13] Pennington, J.; Socher, R. Manning, C. D. (2014), Glove: Global Vectors for Word Representation., in 'EMNLP' , pp. 1532–1543

[14] Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S. Dean, J. (2013), Distributed Representations of Words and Phrases and their Compositionality 'NIPS' , Curran Associates, Inc., , pp. 3111–3119.

[15] Wang, S. I. Manning, C. D. (2012), Baselines and Bigrams: Simple, Good Sentiment and Topic Classification., in 'ACL (2)' , The Association for Computer Linguistics, , pp. 90-94.

[16] Wang, S. I. Manning, C. D. (2012), Baselines and Bigrams: Simple, Good Sentiment and Topic Classification., in 'ACL (2)' , The Association for Computer Linguistics, , pp. 90-94.

[17] "Understanding LSTM Networks", Colah's Blog, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[18] Chung, J.; Gülçehre, Ç.; Cho, K. Bengio, Y. (2014), 'Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling', CoRR abs/1412.3555 .

[19] Józefowicz, R.; Zaremba, W. Sutskever, I. (2015), An Empirical Exploration of Recurrent Network Architectures., in Francis R. Bach David M. Blei, ed., 'ICML' , JMLR.org, , pp. 2342-2350 .

7 Contribution

All the deep learning models and embeddings were learned by both the members. The experimentation on embeddings was split. Xavier and word2Vec were done by Suraj. Fasttext and Glove were done by Shrivardhan. GRU and Naive Bayes SVM were experimented by Suraj, and LSTM and BiLSTM were done by Shrivardan. The writing of the report was shared equally by both the members.