Detecting Toxic Comments By Using LSTM-CNN Model

Prof. Bhalshankar S. T [1], Shubham Gupta[2], Akhilesh Yadav[3], Vaishnav Mahadik[4], Prajval Palwade[5], Prathamesh Kondhare [6]

[1] *Prof. Bhalshankar S.T, Computer Science Engineering Department, MIT College Of Railway Engineering And Research, Barshi, Maharashtra, India, Email ranjit24120@gmail.com*

[2] *Shubham Gupta, Computer Science Engineering Department, MIT College Of Railway Engineering And Research, Barshi, Maharashtra, India, Email sg133311@gmail.com*

[3] *Akhilesh Yadav, Computer Science Engineering Department, MIT College Of Railway Engineering And Research, Barshi, Maharashtra, India, Email akhilyadav569@gmail.com*

[4] *Vaishnav Mahadik, Computer Science Engineering Department, MIT College Of Railway Engineering And Research, Barshi, Maharashtra, India, Email vaishnav.mahadik@mitcorer.edu.inm*

[5] *Prajval Palwade, Computer Science Engineering Department, MIT College Of Railway Engineering And Research, Barshi, Maharashtra, India, Email prajvalpalwade97@gmail.com*

[6] *Prathamesh Kondhare, Computer Science Engineering Department, MIT College Of Railway Engineering And Research, Barshi, Maharashtra, India, Email prathameshkondhare41@gmail.com*

ABSTRACT

Nowadays users leave numerous comments on different social networks, news portals, and forums. Some of the comments are toxic or abusive. Due to the number of comments, it is unfeasible to manually moderate them, so most of the systems use some kind of automatic discovery of toxicity using machine learning models. In this work, we performed a systematic review o

The state-of-the-art in toxic comment classification using machine learning methods.

We have studied the impact of Support vector machines (SVM), Long Short- Term Memory Networks (LSTM), Convolutional Neural Networks (CNN), and Multilayer Perceptron (MLP) methods, in combination with word and character level embeddings, on identifying toxicity in text. We evaluated our approaches on Wikipedia comments from the Kaggle Toxic Comments Classification Challenge dataset. Regarding character-level classification, our best results occurred when using a CNN model.

Machine Learning Techniques for Detecting toxic Comments: An LSTM-CNN Model Evaluation

In recent years, the detection of toxic comments has become a critical task in maintaining healthy online communities. This introduction outlines the evaluation of a hybrid Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) model designed for this purpose. LSTMs are well-suited for capturing sequential dependencies in text, effectively handling the context and nuance in comment threads. Conversely, CNNs excel at identifying local patterns, such as key phrases and n-grams, which are often indicative of toxicity. By combining these architectures, the LSTM-CNN model

leverages the strengths of both to enhance the accuracy and robustness of toxic comment detection. This evaluation aims to compare the performance of the hybrid model against traditional approaches, highlighting its potential advantages in precision, recall, and overall effectiveness in real-world applications.

Subtopics: -

- Data Collection and Annotation
- Sources of toxic comments (social media, etc.)
- Methods for annotating comments (manual labelling, crowdsourcing, etc.)

- Preprocessing Techniques
- Text normalization (stemming, lemmatization)
- Handling imbalanced datasets
- Feature extraction (TF-IDF, word embeddings)

- Machine Learning Approaches
- Traditional models (Logistic Regression, SVM)
- Deep learning models (RNN, LSTM, CNN)
- Hybrid models (e.g., LSTM-CNN, BERT-based models)

- Evaluation Metrics
- Precision, recall, F1-score
- Confusion matrix analysis
- ROC-AUC score

- Challenges in Toxic Comment Detection
- Ambiguity in language (sarcasm, slang)
- Context-dependence of toxicity
- Dataset bias and fairness issues
-

- Real-world Applications and Deployment
- Implementation in social media platforms
- Real-time detection systems
- User feedback and system improvement

1. Overview of toxic comment using LSTM-CNN model.

Toxic Comment Classifier is a competition that has been organized by Jigsaw/Conversation AI and hosted on Kaggle. The data set for building the classification model was acquired from the competition site and it included the training set as well as the test set. The steps elaborated in the workflow below will describe the entire process from Data Pre-Processing to Model Testing.

Data Exploration, Data Pre-processing, and Feature Engineering

Step 1: Checking for missing values.

First and foremost, after importing the training and test data into the pandas dataframe, I decided to check for missing values in the downloaded data. Using the "isnull" function on both the training and test data, I discovered that there were no missing records and therefore, I moved on to the next step of my project.

Step 2: Text Normalization.

As I was now certain that there are no missing records in my data, I decided to start with data pre-processing. Firstly, I decided to normalize the text data since comments from online forums usually contain inconsistent language, use of special characters in place of letters (e.g. @rgument), as well as the use of numbers to represent letters (e.g. n0t). To tackle such inconsistencies in data, I decided to use ***Regex.*** The text normalization steps that I performed are listed below:-

- Removing Characters in between Text.
- Removing Repeated Characters.
- Converting data to lower-case.
- Removing Punctuation.

Step 3: Lemmatization.

Since the data is now clean and consistent, it is the right time to perform ***Lemmatization***. Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. For example, we do not want the Machine Learning algorithm to treat studying, studies, and study as three separate words because, in truth, they are not. Lemmatization helps reduce the words "studying" and "studies" to their root form, i.e. study. To implement Lemmatization, I imported "WordNetLemmatizer" from the "nltk" library, created a function "lemma" to perform Lemmatization, and applied it to the clean data that I procured from Step 2.

Step 4: Stopwords Removal.

Stopword removal, as we all know, is one of the most critical steps in text pre-processing for use-cases that involve text classification. Removing stopwords ensures that more focus is on those words that define the meaning of the text.

i. To remove stopwords from my data, I took the help of the "spacy" library. Spacy has a list of common stopwords, "STOP_WORDS" that can be used to remove stopwords from any textual data.
ii. Although the list provided by Spacy's library is quite extensive, I decided to search for additional stopwords that might be unique to my dataset.
iii. Firstly, I decided to add single-letter and two-letter words to the list of stopwords. While reading through random comments in my dataset, I came across instances where single-letter or two-letter words existed without any context, (e.g. *Wow such a lovely pillow w!!* or *He is such a happy guy bb.*) To make sure that such instances of single-letter or two-letter words do not affect the performance of my deep learning model, I added them to the list of stopwords. However, I made sure that words like *me*, *am*, *a*s, or letters like *I* and *a* are not added to the list of stopwords.

Step 5: Tokenization, Indexing, and Index Representation.

As we all know, machine learning and deep learning models work on numerical data irrespective of the use case. Therefore, to train a deep-learning model using clean text data, the data must be converted into its equivalent machine-readable form. To achieve such a feat. we need to perform the following steps [3]:
Tokenization — We need to break down the sentence into unique words. e.g. "I love cats and love dogs" will become ["I", "love", "cats", "and", "dogs"].
Indexing — We put the words in a dictionary-like structure and give them an index each e.g. {1: "I",2: "love",3: "cats",4: "and",5: "dogs"}.
Index Representation- We could represent the sequence of words in the comments in the form of an index, and feed this chain of index into our deep-learning model. For e.g. [1,2,3,4,2,5].
Using the "Tokenizer" class from the "Keras" library, the above-mentioned steps can be easily performed. This class allows vectorizing a text corpus, by turning each text into either a sequence of integers (each integer being the index of a token in a dictionary) or into a vector where the coefficient for each token could be binary, based on word count, based on tf-idf, etc [4]. The code snippet below demonstrates the conversion of text data into sequence vectors.

Step 6: Padding.

Comments found on online forums or social media platforms are variable in length, some are one-word replies while others are vastly elaborated thoughts. Variable-length sentences are converted into variable-length sequence vectors and we cannot

pass vectors of inconsistent lengths to our deep-learning model. To circumvent this issue, we use Padding. With the help of padding, we can make the shorter sentences as long as the others by filling the shortfall by zeros, and on the other hand, we can trim the longer ones to the same length as the short ones [3]. I used the "pad_sequences" function from the "Keras" library and, I fixed the sentence length at *200* words and applied ***post*** padding ***(i.e. for shorter sentences, 0's will be added at the end of the sequence vector)***. As soon as we are done with the padding of our sequence vectors, we can start creating our deep-learning models.

Model Creation & Model Assessment

Step 1: Split Training Data into Train-Set and Validation-Set.

Since we have completed the data pre-processing and feature engineering part of our project, we move on to the model creation and model assessment part of the project. Before trying to fit a deep learning model on the training data, I randomly split the data into train-set and validation-set. The validation set accounts for 20% of the training data.

Step 2: Import fastText's pre-trained word embeddings.

As mentioned earlier, in the ***Problem Statement***, I wanted to use pre-trained word embeddings from ***fastTex*t** to harness the power of **Transfer Learning**. To do so, I load the fastText word embeddings into my own environment, and then, I create an embedding matrix by assigning the vocabulary with the pre-trained word embeddings.

Step 3: Model Creation (LSTM).

It is now time to choose a deep-learning model and train the model using the train-set and the validation-set. Since we are working on a Natural Language Processing use-case, it is ideal that we use the Long Short Term Memory model (LSTM). LSTM networks are similar to RNNs with one major difference that hidden layer updates are replaced by memory cells. This makes them better at finding and exposing long-range dependencies in data which is imperative for sentence structures [5].

i. Firstly, I imported the "Talos*"* library since it will help us perform hyperparameter tuning as well as model evaluation. Using the "Scan" function, I performed a GridSearchCV and found the best parameters that would give me the highest accuracy.
ii. Next, using the best hyper-parameters, I defined the required number of layers for the LSTM model, compiled the model, and lastly trained the model using the train-set and validation-set.

Step 4: Model Creation (LSTM-CNN).

During the research phase of my project, I came across papers that achieved Toxic Comment Classification using a hybrid model (***i.e. an LSTM and CNN model that worked together***). Such architecture, for a deep-learning model, intrigued me. LSTM can effectively preserve the characteristics of historical information in long text sequences whereas CNN can extract the local features of the text [6]. Combining the two traditional neural network architectures will help us harness their combined capabilities. Therefore, I decided to implement an LSTM-CNN hybrid model as a part of my project. The goal was to compare the performance of both the deep-learning architectures and ascertain the best deep-learning model for my project.

Similar to the process followed in Step 3, I discovered the best hyper-parameters for my **hybrid** model using "Talos". Once the operation was completed, I evaluated the results and picked the hyperparameters which gave me the highest accuracy. Finally, I trained my hybrid model using the train-set and the validation-set.

Step 5: Evaluate the Model Accuracy and Model Loss during the training phase.

As we have completed the training of both our deep-learning models, we should now visualize their accuracy and loss values during the entire training process. Ideally, the loss value for any deep-learning model should decrease as the number of epochs increases, on the other hand, the accuracy should increase as the number of epochs increases. This gives us a fairly decent idea about the quality of our deep-learning model, and whether it has been appropriately trained. Trends in the accuracy and the loss values during every epoch can be seen in the images below.

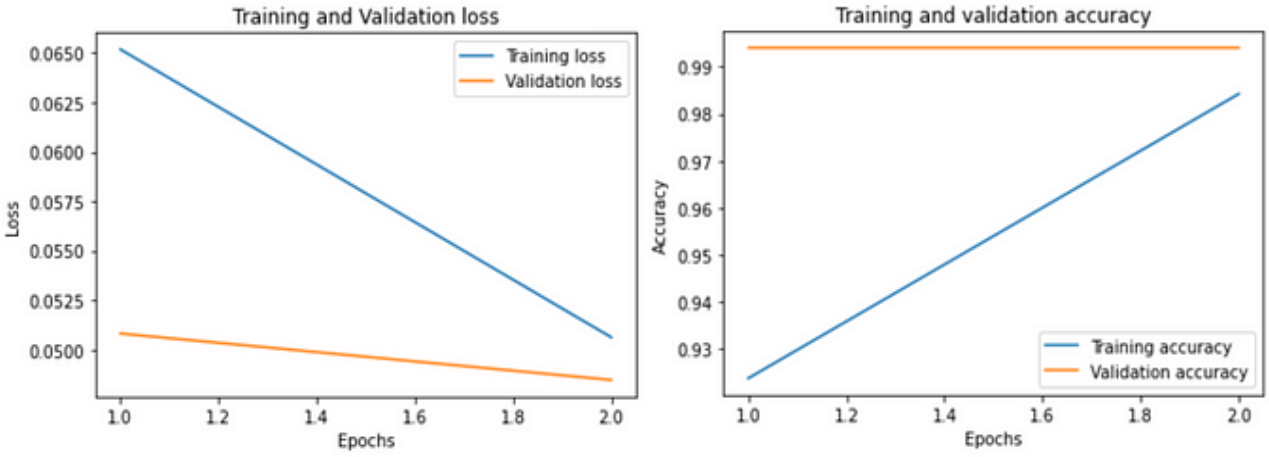


Figure1. Loss and Accuracy values for the LSTM model throughout 2 epochs.

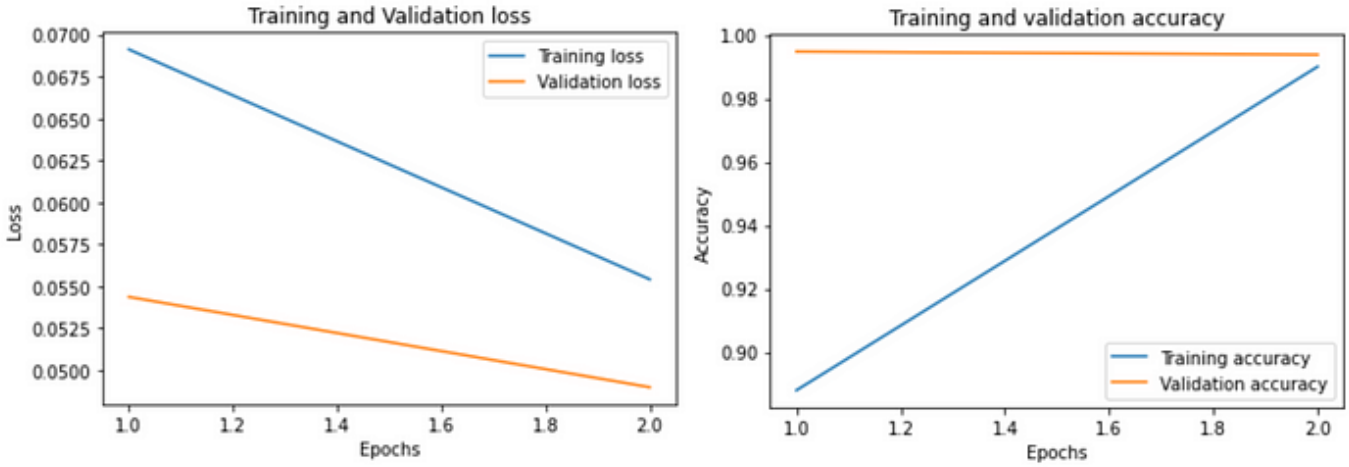


Figure 2..Loss and Accuracy values for the LSTM-CNN model over a span of 2 epochs.

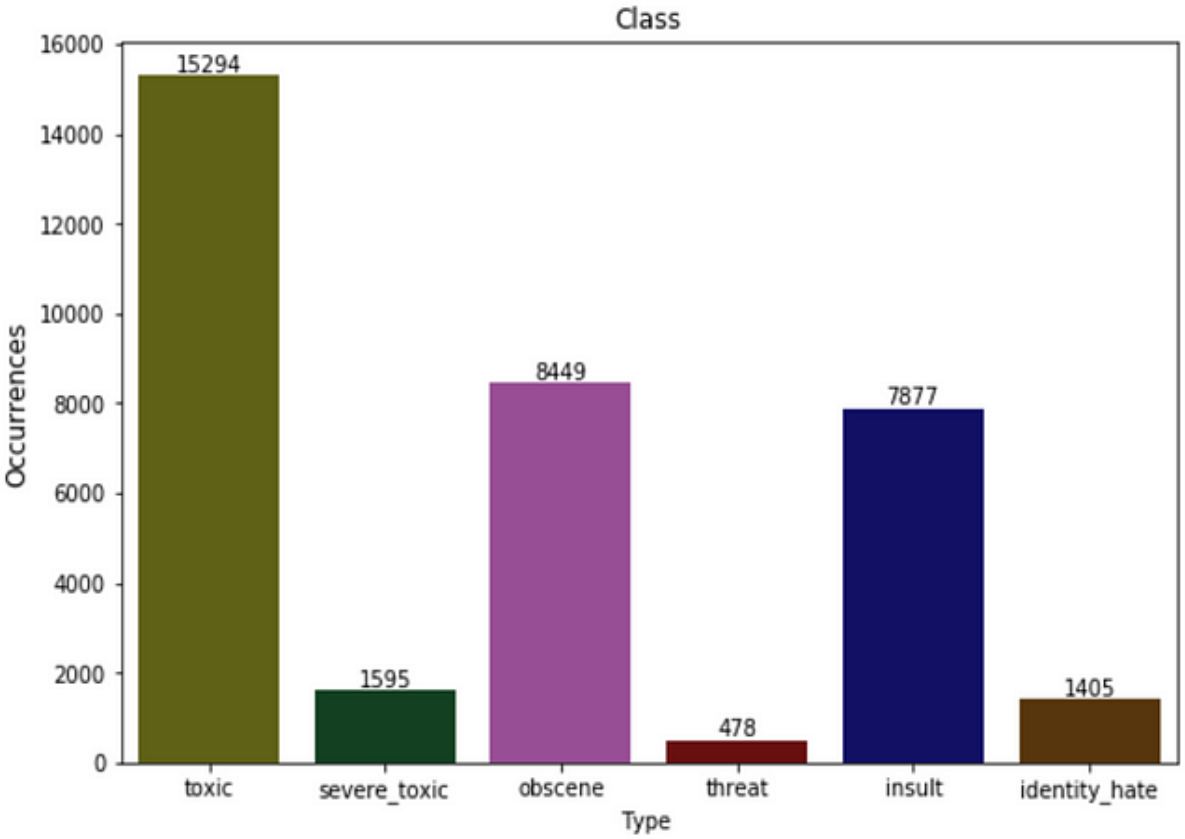


Figure 3. toxic levels in the comments classification in train dataset

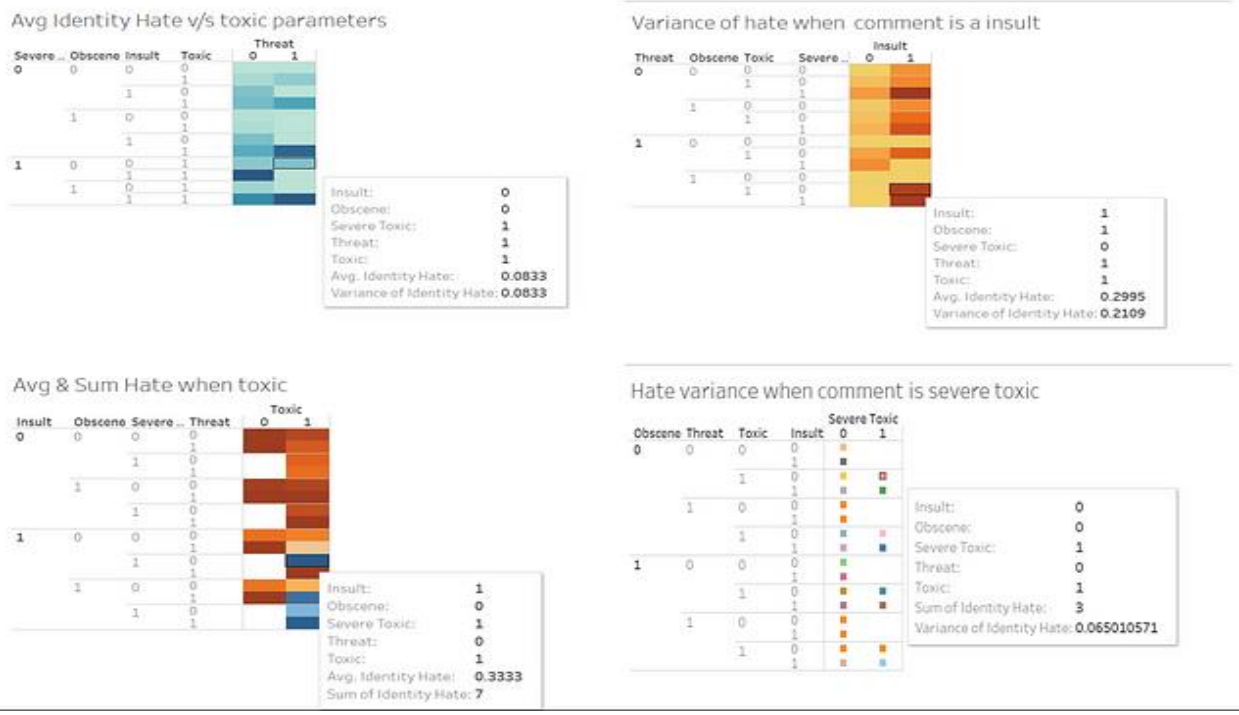


Figure 4. Variance with comment's toxic category on Tableau

Step 6: Padding.

Comments found on online forums or social media platforms are variable in length, some are one-word replies while others are vastly elaborated thoughts. Variable-length sentences are converted into variable-length sequence vectors and we cannot pass vectors of inconsistent lengths to our deep-learning model. To circumvent this issue, we use Padding. With the help of padding, we can make the shorter sentences as long as the others by filling the shortfall by zeros, and on the other hand, we can trim the longer ones to the same length as the short ones [3]. I used the "pad_sequences" function from the "Keras" library and, I fixed the sentence length at *200* words and applied ***post*** padding ***(i.e. for shorter sentences, 0's will be added at the end of the sequence vector)***. As soon as we are done with the padding of our sequence vectors, we can start creating our deep-learning models.

Model Creation & Model Assessment

Step 1: Split Training Data into Train-Set and Validation-Set.

Since we have completed the data pre-processing and feature engineering part of our project, we move on to the model creation and model assessment part of the project. Before trying to fit a deep learning model on the training data, I randomly split the data into train-set and validation-set. The validation set accounts for 20% of the training data.

Step 2: Import fastText's pre-trained word embeddings.

As mentioned earlier, in the ***Problem Statement,*** I wanted to use pre-trained word embeddings from ***fastTex*t** to harness the power of **Transfer Learning**. To do so, I load the fastText word embeddings into my own environment, and then, I create an embedding matrix by assigning the vocabulary with the pre-trained word embeddings.

Step 3: Model Creation (LSTM).

It is now time to choose a deep-learning model and train the model using the train-set and the validation-set. Since we are working on a Natural Language Processing use-case, it is ideal that we use the Long Short Term Memory model (LSTM). LSTM networks are similar to RNNs with one major difference that hidden layer updates are replaced by memory cells. This makes them better at finding and exposing long-range dependencies in data which is imperative for sentence structures [5].

i. Firstly, I imported the "Talos*"* library since it will help us perform hyperparameter tuning as well as model evaluation. Using the "Scan" function, I performed a GridSearchCV and found the best parameters that would give me the highest accuracy.

ii. Next, using the best hyper-parameters, I defined the required number of layers for the LSTM model, compiled the model, and lastly trained the model using the train-set and validation-set.

Step 4: Model Creation (LSTM-CNN).

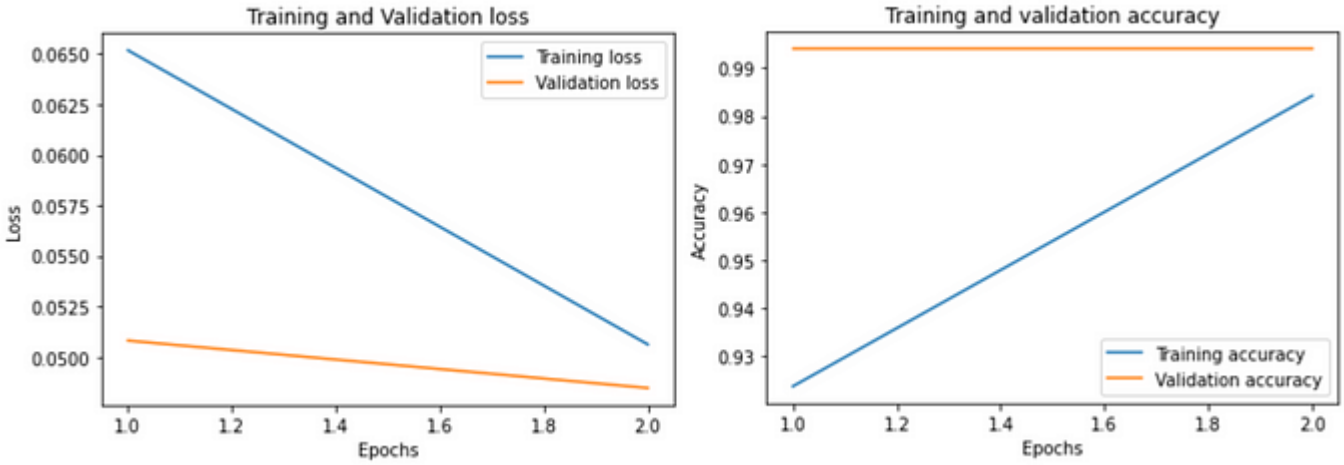
During the research phase of my project, I came across papers that achieved Toxic Comment Classification using a hybrid model (***i.e. an LSTM and CNN model that worked together***). Such architecture, for a deep-learning model, intrigued me. LSTM can effectively preserve the characteristics of historical information in long text sequences whereas CNN can extract the local features of the text [6]. Combining the two traditional neural network architectures will help us harness their combined capabilities. Therefore, I decided to implement an LSTM-CNN hybrid model as a part of my project. The goal was to compare the performance of both the deep-learning architectures and ascertain the best deep-learning model for my project.

Similar to the process followed in Step 3, I discovered the best hyper-parameters for my **hybrid** model using "Talos". Once the operation was completed, I evaluated the results and picked the hyperparameters which gave me the highest accuracy. Finally, I trained my hybrid model using the train-set and the validation-set.

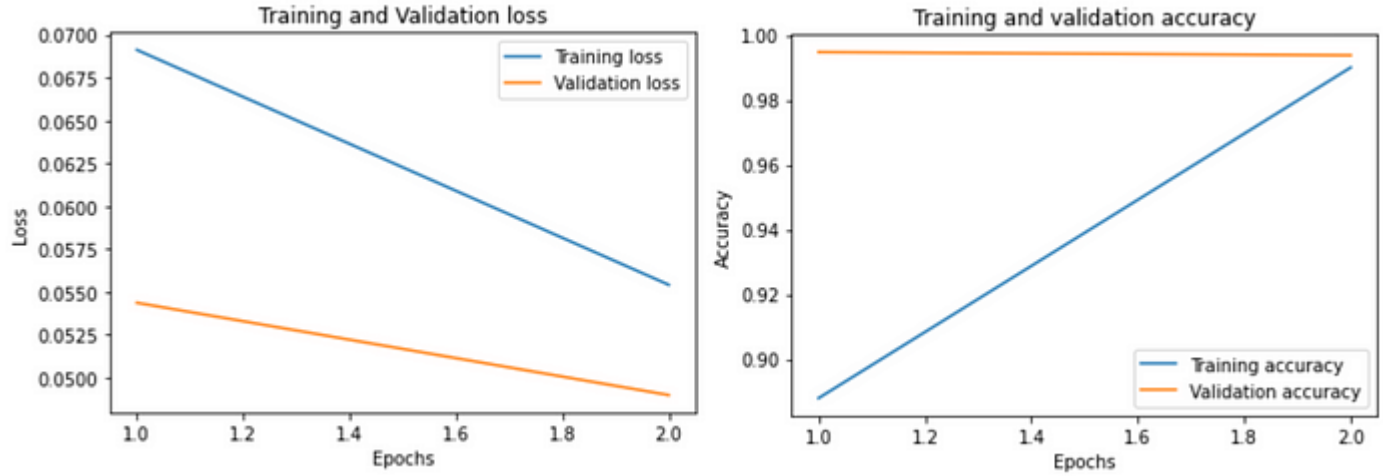
Common symptoms include blurred vision, floaters, dark or empty areas in the vision, and vision loss. Risk factors include poor blood sugar control, duration of diabetes, high blood pressure, and high cholesterol.

Step 5: Evaluate the Model Accuracy and Model Loss during the training phase.

As we have completed the training of both our deep-learning models, we should now visualize their accuracy and loss values during the entire training process. Ideally, the loss value for any deep-learning model should decrease as the number of epochs increases, on the other hand, the accuracy should increase as the number of epochs increases. This gives us a fairly decent idea about the quality of our deep-learning model, and whether it has been appropriately trained. Trends in the accuracy and the loss values during every epoch can be seen in the images below.



Loss and Accuracy values for the LSTM model over a span of 2 epochs. (Image by author)



Loss and Accuracy values for the LSTM-CNN model over a span of 2 epochs. (Image by author)

Step 6: Calculating Model Accuracy using Test-set

Evaluating the model based on accuracy and loss values gave me promising results. It gave me the confidence to assess the performance of my deep learning models using the test set. As mentioned earlier in this blog, the test-set was procured from *Kaggle* and it was passed through the same data pre-processing and feature engineering steps as the training data. Since I now have the processed test data available with me, I used the "predict" function to generate outputs for the inputs present in the test data.

As soon as the above process was completed for both my deep learning models, I uploaded the respective ".csv" output files to the Kaggle Competition and submitted them to generate the final accuracy scores. The maximum accuracy scores for both my deep learning models can be seen in the image below.

Deep Learning Model Accuracy Scores		
Model	Score Type	
	Private	Public
LSTM	97.70%	97.40%
LSTM-CNN	97.10%	97.07%

Comparison of Accuracy scores for the traditional LSTM Model and the hybrid LSTM-CNN Model. (Image by author)

Conclusion

After evaluating the results procured during the training phase of my project and the results that I received from the competition website, I can claim that the **traditional LSTM Model** performs **better** than the **hybrid LSTM-CNN Model.** The hybrid model loses marginally to the traditional deep-learning model which states that the traditional LSTM model is the right choice for the Toxic Comment Classification use-case.

Reference: -

1. van Aken, B., Risch, J., Krestel, R., L¨oser, A.: Challenges for toxic comment classification: An in-depth error analysis. In: Proceedings of the Workshop on Abusive Language Online (ALW@EMNLP), pp. 33–42 (2018)
2. Alber, M., Lapuschkin, S., Seegerer, P., H¨agele, M., Sch¨utt, K.T., Montavon, G., Samek, W., M¨uller, K.R., D¨ahne, S., Kindermans, P.J.: innvestigate neural networks! arXiv preprint arXiv:1808.04260 (2018)
3. Ambroselli, C., Risch, J., Krestel, R., Loos, A.: Prediction for the newsroom: Which articles will get the most comments? In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), pp. 193–199. ACL (2018)
4. Arras, L., Horn, F., Montavon, G., M¨uller, K.R., Samek, W.: Explaining predictions of non-linear classifiers in nlp. In: Proceedings of the Workshop on Representation Learning for NLP, pp. 1–7. Association for Computational Linguistics (2016)
5. Arras, L., Montavon, G., M¨uller, K.R., Samek, W.: Explaining recurrent neural network predictions in sentiment analysis. In: Proceedings of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pp. 159–168. Association for Computational Linguistics, Copenhagen, Denmark (2017)
6. Bach, S., Binder, A., Montavon, G., Klauschen, F., M¨uller, K.R., Samek, W.: On

pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one 10(7) (2015)
7. Badjatiya, P., Gupta, S., Gupta, M., Varma, V.: Deep learning for hate speech detection in tweets. In: Proceedings of the International Conference on World Wide Web (WWW), pp. 759–760. International World Wide Web Conferences Steering Committee (2017)
24 Julian Risch and Ralf Krestel
8. Berry, G., Taylor, S.J.: Discussion quality diffuses in the digital public square. In: Proceedings of the International Conference on World Wide Web (WWW), pp. 1371–1380. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2017)
9. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics (TACL) 5(1), 135–146 (2017)
10. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research 16(1), 321–357 (2002)
11. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder–decoder for statistical machine translation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734. Association for Computational Linguistics (2014)
12. Davidson, T., Warmsley, D., Macy, M., Weber, I.: Automated hate speech detection and the problem of offensive language. In: Proceedings of the International Conference on Web and Social Media (ICWSM), pp. 512–515 (2017)
13. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
14. Diakopoulos, N.: Picking the nyt picks: Editorial criteria and automation in the curation of online news comments. International Symposium on Online Journalism (ISOJ) 6(1), 147–166 (2015)
15. Diakopoulos, N., Naaman, M.: Towards quality discourse in online news comments. In: Proceedings of the Conference on Computer Supported Cooperative Work (CSCW), pp. 133–142. ACM (2011)
16. Djuric, N., Zhou, J., Morris, R., Grbovic, M., Radosavljevic, V., Bhamidipati, N.: Hate speech detection with comment embeddings. In: Proceedings of the International Conference on World Wide Web (WWW), pp. 29–30. International World Wide Web Conferences Steering Committee (2015)
17. Gal´an-Garc´ıa, P., Puerta, J.G.d.l., G´omez, C.L., Santos, I., Bringas, P.G.: Supervised machine learning for the detection of troll profiles in twitter social network: Application to a real case of cyberbullying. Logic Journal of the IGPL 24(1), 42–53 (2016)
18. Gamb¨ack, B., Sikdar, U.K.: Using convolutional neural networks to classify hate-speech. In: Proceedings of the Workshop on Abusive Language Online (ALW@ACL), pp. 85–90 (2017)
19. G´omez, V., Kaltenbrunner, A., L´opez, V.: Statistical analysis of the social network and discussion threads in slashdot. In: Proceedings of the International Conference on World Wide Web (WWW), pp. 645–654. ACM (2008)
20. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2013)
21. Guberman, J., Schmitz, C., Hemphill, L.: Quantifying toxicity and verbal violence on twitter. In: Proceedings of the Conference on Computer Supported Cooperative Work (CSCW), pp. 277–280. ACM, New York, NY, USA (2016)
22. Hardaker, C.: Trolling in asynchronous computer-mediated communication: From user discussions to academic definitions. Journal of Politeness Research. Language, Behaviour, Culture 6, 215–242 (2010)
23. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation 9(8), 1735–1780 (1997)

24. Howard, J., Ruder, S.: Universal language model fine-tuning for text classification. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), pp. 328–339. Association for Computational Linguistics, Melbourne, Australia (2018)
Toxic Comment Detection in Online Discussions 25
25. Kindermans, P.J., Sch¨utt, K.T., Alber, M., M¨uller, K.R., Erhan, D., Kim, B., D¨ahne, S.: Learning how to explain neural networks: PatternNet and PatternAttribution. arXiv preprint arXiv:1705.05598 (2017)
26. Kolhatkar, V., Taboada, M.: Constructive language in news comments. In: Proceedings of the Workshop on Abusive Language Online (ALW@ACL), pp. 11–17 (2017)
27. Kolhatkar, V., Taboada, M.: Using new york times picks to identify constructive comments. In: Proceedings of the Workshop: Natural Language Processing meets Journalism@EMNLP, pp. 100–105 (2017)
28. Kumar, S., Shah, N.: False information on web and social media: A survey. arXiv preprint arXiv:1804.08559 (2018)
29. Lampe, C., Resnick, P.: Slash (dot) and burn: distributed moderation in a large online conversation space. In: Proceedings of the Conference on Human Factors in Computing Systems (CHI), pp. 543–550. ACM (2004)
30. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems (NIPS), pp. 3111–3119 (2013)
31. Mnih, V., Heess, N., Graves, A., Kavukcuoglu, K.: Recurrent models of visual attention. In: Advances in Neural Information Processing Systems (NIPS), pp. 2204–2212 (2014)
32. Napoles, C., Pappu, A., Tetreault, J.R.: Automatically identifying good conversations online (yes, they do exist!). In: Proceedings of the International Conference on Web and Social Media (ICWSM), pp. 628–631 (2017)
33. Napoles, C., Tetreault, J., Pappu, A., Rosato, E., Provenzale, B.: Finding good conversations online: The yahoo news annotated comments corpus. In: Proceedings of the Linguistic Annotation Workshop (LAW), pp. 13–23 (2017)
34. Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y., Chang, Y.: Abusive language detection in online user content. In: Proceedings of the International Conference on World Wide Web (WWW), pp. 145–153. International World Wide Web Conferences Steering Committee (2016)
35. Park, D., Sachar, S., Diakopoulos, N., Elmqvist, N.: Supporting comment moderators in identifying high quality online news comments. In: Proceedings of the Conference on Human Factors in Computing Systems (CHI), pp. 1114–1125. ACM (2016)
36. Park, J.H., Fung, P.: One-step and two-step classification for abusive language detection on twitter. In: Proceedings of the Workshop on Abusive Language Online (ALW@ACL), pp. 41–45. Association for Computational Linguistics, Vancouver, BC, Canada (2017)
37. Pavlopoulos, J., Malakasiotis, P., Androutsopoulos, I.: Deeper attention to abusive user content moderation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1125–1135. Association for Computational Linguistics, Copenhagen, Denmark (2017)
38. Pavlopoulos, J., Malakasiotis, P., Bakagianni, J., Androutsopoulos, I.: Improved abusive comment moderation with user embeddings. In: Proceedings of the Workshop on Natural Language Processing meets Journalism (co-located with EMNLP), pp. 51–55. Association for Computational Linguistics, Copenhagen, Denmark (2017)
39. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
40. Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguis-

tics (NAACL), pp. 2227–2237. Association for Computational Linguistics, New Orleans, Louisiana (2018)
26 Julian Risch and Ralf Krestel
41. Pitsilis, G.K., Ramampiaro, H., Langseth, H.: Effective hate-speech detection in twitter data using recurrent neural networks. Applied Intelligence 48(12), 4730–4742 (2018)
42. Qian, J., ElSherief, M., Belding-Royer, E.M., Wang, W.Y.: Leveraging intra-user and inter-user representation learning for automated hate speech detection. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), pp. 118–123 (2018)
43. Risch, J., Krebs, E., L¨oser, A., Riese, A., Krestel, R.: Fine-grained classification of offensive language. In: Proceedings of GermEval (co-located with KONVENS), pp. 38–44 (2018)
44. Risch, J., Krestel, R.: Measuring and facilitating data repeatability in web science. Datenbank-Spektrum DOI 10.1007/s13222-019-00316-9
45. Risch, J., Krestel, R.: Aggression identification using deep learning and data augmentation. In: Proceedings of the Workshop on Trolling, Aggression and Cyberbullying (TRAC@COLING), pp. 150–158 (2018)
46. Risch, J., Krestel, R.: Delete or not delete? semi-automatic comment moderation for the newsroom. In: Proceedings of the Workshop on Trolling, Aggression and Cyberbullying (TRAC@COLING), pp. 166–176 (2018)
47. Rizos, G., Papadopoulos, S., Kompatsiaris, Y.: Predicting news popularity by mining online discussions. In: Proc. of the Int. Conf. on World Wide Web Companion (WWW), pp. 737–742. International World Wide Web Conferences Steering Committee (2016)
48. Schabus, D., Skowron, M.: Academic-industrial perspective on the development and deployment of a moderation system for a newspaper website. In: Proceedings of the Language Resources and Evaluation Conference (LREC), pp. 1602–1605 (2018)
49. Schabus, D., Skowron, M., Trapp, M.: One million posts: A data set of german online discussions. In: Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR), pp. 1241–1244 (2017)
50. Schmidt, A., Wiegand, M.: A survey on hate speech detection using natural language processing. In: Proceedings of the International Workshop on Natural Language Processing for Social Media, pp. 1–10 (2017)
51. Stroud, N.J., Van Duyn, E., Peacock, C.: News commenters and news comment readers. Engaging News Project pp. 1–21 (2016)
52. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems (NIPS), pp. 5998–6008 (2017)
53. Wang, C.: Interpreting neural network hate speech classifiers. In: Proceedings of the Workshop on Abusive Language Online (ALW@EMNLP), pp. 86–92. Association for Computational Linguistics, Brussels, Belgium (2018)
54. Waseem, Z.: Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In: Proceedings of the Workshop on NLP and Computational Social Science, pp. 138–142. Association for Computational Linguistics, Austin, Texas (2016)
55. Waseem, Z., Davidson, T., Warmsley, D., Weber, I.: Understanding abuse: A typology of abusive language detection subtasks. In: Proceedings of the Workshop on Abusive Language Online (ALW@ACL), pp. 78–84. Association for Computational Linguistics, Vancouver, BC, Canada (2017)
56. Waseem, Z., Hovy, D.: Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In: Proceedings of the Student Research Workshop@NAACL, pp. 88–93. Association for Computational Linguistics, San Diego, California (2016)
57. Wei, Z., Liu, Y., Li, Y.: Is this post persuasive? ranking argumentative comments in online forum. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), vol. 2, pp. 195–200 (2016)