CSC 791 - Natural Language Processing

Term Project Report R3

Team Members:
- **Harsh Kachhadia** (hmkachha)(200370255)(791 student)
- **Karan Shah** (krshah3)(200365051)(791 student)

Project Title: Toxic Comment Classification

Project Description:

❖ Motivation

Major motivation for this project comes from the fact that, with the rising number of social media users or blog readers, instead of everyone sharing their thoughts openly with others, people are hesitating from opening in front of others, because of rising toxicity in comments of people on other thoughts or posts. This fear of abuse and harassment stops many people from expressing themselves and thus they give up on getting help from others. Platforms also struggle to efficiently promote healthy conversations, resulting in many users to limit or completely shut down user comments.

❖ Aim:

The primary aim of this project will be to build a model which will be able to accurately classify comments either from social media or any blog into one or more of the prespecified categories such as:

- toxic
- severe_toxic
- obscene
- threat
- insult
- identity_hate

A single comment can belong to multiple categories. This is a task of multi-label text classification.

Extension to the project: A possible extension to this project will be to integrate this trained model into a Python Flask Web API, which can later be used for other software or users to detect comment toxicity by making an API call.

❖ Hypothesis:

For this project, we are assuming the following points as our hypotheses, which will be tested based on results obtained from work done in this multi-label text classification project.

- For a multi-label text classification task, which should be preferred - Machine Learning Algorithms or Deep Learning Neural Networks?
- Do word-embedding techniques play an important role in text classification tasks? If yes, are all word embedding techniques same or is it like some perform better than others?

❖ Impact on Real Life:

a. **Stakeholders:** All the users on social media or blogs or any platform where comments can be made. Stakeholders also include the platform that facilitates such conversations such as social media sites.

b. **Usage Scenarios:** This type of software can be utilized by platform providers to remove hate comments from posts or discussions, in order to provide a healthy and safe online environment.

c. **Intended Users:** The intended users for this software will be the social media companies, who will utilize the Web API to classify the comments of users on posts and remove the indecent comments. This will protect the users on the platform from hate content and will support the social media companies' mission of providing a safe and decent social environment.

❖ Datasets: Link

We will be using the **toxic comment dataset** from Kaggle. This dataset contains a **large number of Wikipedia comments** which have been **labeled by human raters** for toxic behavior into the categories mentioned below..

a. Data Exploration:

The training set is of the size **159571 rows** and consists of the text scraped from various sources. It also contains labels for identifying whether the text belongs to any one of the following classes. Note that they are not mutually exclusive.

- Toxic
- Severely Toxic
- Obscene
- Threat
- Insult
- Identity Hate

Out of these 159571 rows, **16225 rows contain such types of toxic comments**.
The remaining **143346 are clean comments**.

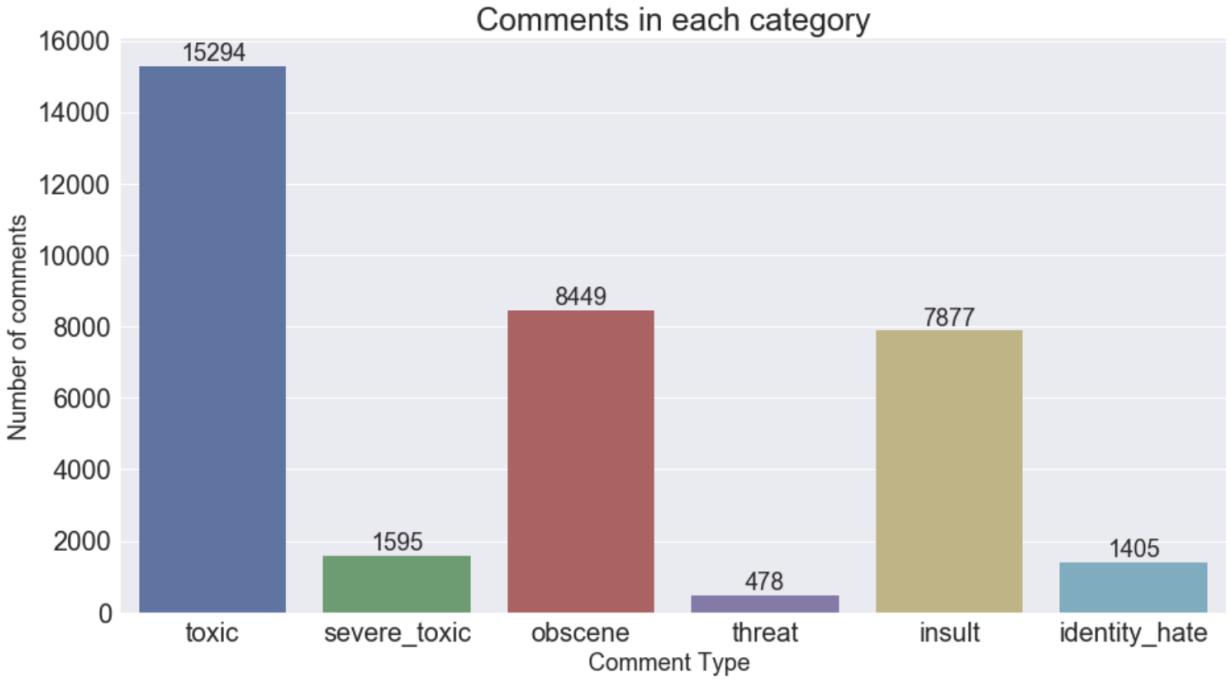

Figure 1: Distribution of comments in different classes (a comment may belong to multiple classes.)

Talking about the testing dataset, there are two files that can be used for testing purposes present in the kaggle data - **test.csv** and **test_labels.csv**. These files contain the **testing data** and their **corresponding labels**. Overall the **testing data has 153164 rows**. Out of which some of the data does not have corresponding labels. Thus, **we removed such rows containing the -1 label**. This brings the **testing dataset size down to 63978 rows** which can be used for testing purposes in our project.

As can be seen, each of these classes are **important to identify for a system looking to make content clean for the user**s. The table below is an example of a few rows of the data.

1	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK
2	You are gay or antisemmitian? Archangel WHite TigerMeow! Greetingshhh! Uh, there are two ways, why you do erased my comment about WW2, that holocaust was brutally slaying of Jews and not gays/Gypsys/Slavs/anyone... 1 - If you are anti-semitian, than shave your head bald and go to the skinhead meetings! 2 - If you doubt words of the Bible, that homosexuality is a deadly sin, make a pentagram tatoo on your forehead go to the satanistic masses with your gay pals! 3 - First and last warning, you fucking gay - I won't appreciate if any more nazi shwain would write in my page! I don't wish to talk to you anymore! Beware of the Dark Side!

As can be observed, they are **pretty disturbing comments and need to be flagged**.

❖ Implementation/Methodology

Let us break down the implementation into a few sections and briefly discuss each of them separately.

a. Dataset Preprocessing

For the problem at hand, before we feed our data into the model, we **need to preprocess** it, in order to **make it suitable for training** and for efficient model learning.

Following are the preprocessing steps involved:

1. Removing Punctuations, Special Characters, Digits
2. Removing Stopwords
3. Lowercasing
4. Tokenization

Before starting with data preprocessing, we checked whether the data has any **rows with missing values or NaN values** using **isnull()** and **isnan()** methods. We didn't find any such rows.

After checking for the missing values, we **cleaned the text by eliminating punctuations, special characters, and digits using regex and predefined punctuations**. We also **removed the stopwords with the help of NLTK stopwords** bag of words to make the text more specific. We even **lowercase the sentences** for simplicity. And at last, those **sentences were tokenized** in order to turn them into smaller units

This data preprocessing was performed on both, 'comment_text' columns present in training and testing data. **The labels were not processed as they were binary labels**.

Class Imbalance Problem:

As we can witness the imbalance between the number of types of comments(clean and toxic), where there are **16225 toxic comments** and **143346 clean comments.** Also inside the 16225 toxic comments, toxicity classes are also imbalanced. There are more of 'toxic' label comments than in other label comments as we can see in Figure 1. We tried to solve this problem during the baseline model implementation, but since this is a multi-label classification task, publicly available libraries such as **'imblearn'** are unable to up-sample or down-samples, as they are built specifically for single label classification tasks.

Along with that, we tried up-sampling the data using SMOTE technique but it would not have been useful, since we already have ~150000 data points in training data, and we were facing google colab session crash issue, we decided not to go for solving class imbalance problem and simply move forward with the original dataset.

b. Baseline Model

For the baseline model, we have decided to **use a machine learning model**. This will help us learn 2 things. 1) Which ML or DL model is better for this problem in terms of training and how much data is needed 2) Which of ML or DL gives us better results in terms of metrics such as accuracy, etc.

The baseline model flow will be as follows:

Word Embedding -> TF-IDF:

First of all the training text and testing text will be **vectorized** and **word embeddings** will be generated using the **TF-IDF vectorizer** from the sklearn library in python. The data will further be fitted using the fit_transform method of TF-IDF vectorizer from sklearn.

This fitted training data will be used for model training, while the fitted testing data will be used for evaluation purposes.

Model Selection:

The machine learning algorithm that we have used here is: **Logistic Regression** and we will be using the in-built implementation of Logistic Regression algorithm from sklearn library. We selected the Logistic Regression algorithm after comparing it with different Machine Learning algorithms such as Multinomial Naive Bayes, SVM, etc. Out of these 3 algorithms, Logistic Regression algorithms perform the best in terms of metric such as accuracy, precision, recall, and f1-score.

Model Training and Hyper-parameter Tuning:

Since the in-built machine learning models in sklearn library such as the one that we are using(Logistic Regression) are for **single label classification purposes**, in order to proceed with our **multi-label classification task**, we decided to used the **ClassifierChain** from the **skmultilearn** library in python. This ClassifierChain, trains inbuilt ML models of sklearn on multilabel data by using the

concept of **model chaining**. This trains models on each class of dataset individually and then combines them.

Problems during model training:

The ClassiferChain approach was a better way to solve our problem of training on multilabel dataset, but this resulted in problems such as **Session Crash ON Google Colab** and **Kernel Crash when trained locally on PC.**

As a result we tried to lower the count of the dataset, by **not training on rows that are clean comments**. This means we tried to train the ClassifierChain model on **16225 rows** which actually are **labelled (non clean comments)**. But the problem still persisted. Thus we had to go for a different route.

Solution to the above problem:

As a result at the end, we had to train each Logistic Regression model individually with each class. This means that different models had to be trained for each class individually. This resulted in **6 logistic regression models** each trained on **159571 data points**. This means that we trained each model on the same text of training dataset but the fitting labels were individually different.

Hyperparameter Tuning:

During their training, we experimented(Hyper-parameter tuning) with the following :

1. Random state values
2. Types of solver
3. max_iters values (Number of iterations)
4. Number of units in LSTM Layers, Bidirectional Layers, Dense Layers

This experimentation was to achieve the best performance out of all the tried models..

Model Evaluation:

For evaluation purposes, we used the **test dataset** that contained **63978 rows of comments(text).** We obtained model predictions from each of the 6 trained models and stacked them column wise in an numpy ndarray in order to obtain an output structure similar to the actual labels. With this ndarray, we obtained the combined accuracy of these 6 models.

The accuracy that we obtained is as follows:

Accuracy: 89.5%

We aim to improve the accuracy further as discussed in the proposed model.

c. Proposed Model

For the baseline models, there were two problems. The following were the problems and their solutions which were achieved in the proposed model.

1. The simple vectorizer (TF-IDF) could not encode the meaning of text and even if so, it ignores n-grams. The sequential nature is not recorded.
2. The basic machine learning models do not look at text data as sequences rather than mutually exclusive data points.

In order to tackle the above problems, in our proposed model, we plan to use advanced word embedding techniques such as Keras Inbuilt Word Embedding and Glove Embedding, where we try to solve the problem 1 mentioned above. Along with that, in order to tackle problem 2, we have implemented advanced deep learning neural networks such as LSTM (Long Short Term Memory) and BiDirectional LSTM.

Feature Additions:

We had planned to add two additional features:
Sentiment Analysis of comments
Length of Sentences

Sentiment Analysis of comments: We added this feature by using the TextBlob library sentiment analysis along with the comments. This feature did not turn out to be that useful and contributing, as later we discovered that most of the comments are Negative by sentiment, and thus, did not add much to the knowledge base. As a result, we removed this feature.

Length of Sentences: We added this feature, but this one too did not contribute much as in this case, semantics and words of comments matter more as compared to the length of sentences. Thus, we decided not to proceed with this feature and just proceed with the original dataset.

Word Embedding:

For our proposed model, we have decided to try and compare two word embedding techniques: 1) Keras inbuilt word embedding 2) Glove embedding technique. These are some of the most popular embedding techniques. We decided to use a modified word embedding to account for bigrams and trigrams in the text. Using a more complex encoding technique allowed us to create a better language model.

Model Selection:

For our proposed model, we decided to use two advanced neural network models - 1) LSTM 2) BiDirectional LSTM. As a classifier, we decided to use LSTM and BiDirectional LSTM to create a language model and classify the sentence. Having trained such advanced classifiers will learn about the semantics and vocabulary of the dataset in detail as compared to the Machine Learning model -

Logistic Regression. We could have aimed for more complex neural networks, but these two neural networks gave better results, thus, there was no need to move to other neural networks.

Model Training and Hyper-parameter Tuning:

We contrasted and compared the performance of three different models using different embedding techniques and tweaking hyperparameters to enhance performance. The models are as follows:

1. Keras Embedding with Single Layer LSTM
2. Keras Embedding with Bidirectional LSTM
3. GloVe(300 dimension) embedding with Bidirectional LSTM

During their training, we experimented(Hyper-parameter tuning) with the following :

1. Dropout Rates
2. BatchNormalization
3. Number of LSTM Layers, Bidirectional Layers, Dense Layers, Dropout Layers
4. Number of units in LSTM Layers, Bidirectional Layers, Dense Layers

This experimentation was to solve the problem of underfitting due to a large number of features by increasing the model complexity.

We also experimented with the “Number of epochs” for our models. From the experimentation, we found out that the results turned out to be quite similar to each other when we changed the number of epochs ranging from 2 to 5. Thus, since results were quite similar for the number of epochs, we decided to go till 2-3 epochs for our models, since the model training times were quite high and we also had to keep in mind the 12-hour GPU usage limit of google colab.

In the third model, we also implemented a learning rate scheduler that increased the learning rate after every epoch to give the model time to map the features well. The third model was also set to train the embedding layer on top of the pretrained GloVe model to increase the performance.

Challenges during Model Training:

The most common type of issue that we faced during model training and hyper-parameter tuning was the message on Google Colab which stated that the GPU runtime of Google Colab was available for continuous use for only 12 hours of a day. Since, we were trying out models with different parameters in order to find the best model, we had to do a lot of model training, and as a result, we used to reach the max limit on GPU usage time.

Model Evaluation:

For evaluation purposes, we decided to use the metric “Accuracy” as this is the most common metric used in cases of multi-label classification tasks. For evaluation purposes, we used the **test dataset** that contained **63978 rows of comments(text).** We obtained model predictions from each of the 3 trained models.

After which accuracies for all the 3 models were obtained. They have been discussed in detail in the **Results and Findings** section.

d. Results and Findings

From the journey so far, we have obtained results in the form of accuracies and losses for all the models from both baseline models and proposed models. As for evaluation purposes, as discussed above, we had decided to use the metric “Accuracy” which is the most common metric used in cases of multi-label classification tasks. Thus, the primary criteria for comparison that we will be using is Accuracy, which has been presented in a tabular manner below:

Results:

Model	**Accuracy**
TF-IDF Embedding with Logistic Regression (Baseline Model)	89.5%
Keras Embedding with Single Layer LSTM	88.6%
Keras Embedding with Bidirectional LSTM	88.1%
GloVe(300 dimension) embedding with Bidirectional LSTM	**94.86%**

From the above table, we can compare all the models' performance at once, and clearly see that the model **“GloVe(300 dimension) embedding with Bidirectional LSTM”** performs best for this multi-label classification task and is our final proposed model for this task.

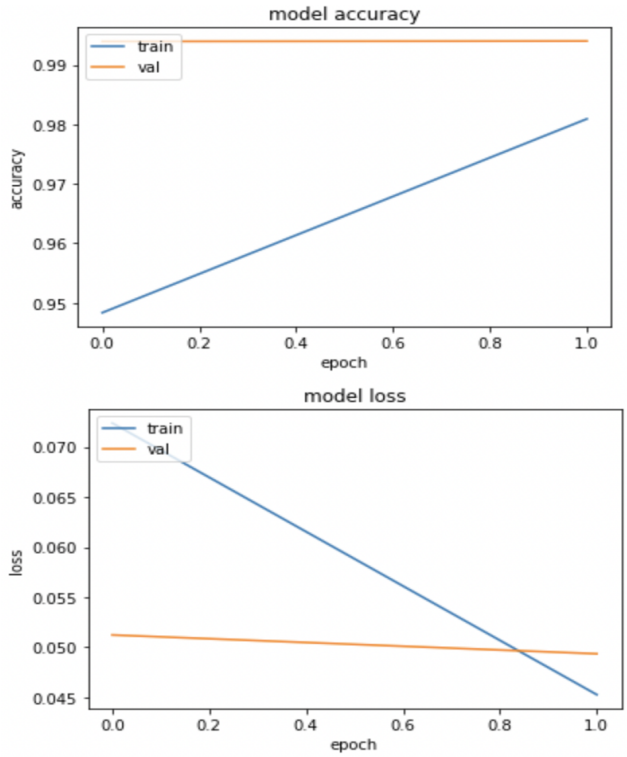

Figure 2: Model Accuracy and Model Loss of "Keras Embedding with Single Layer LSTM"

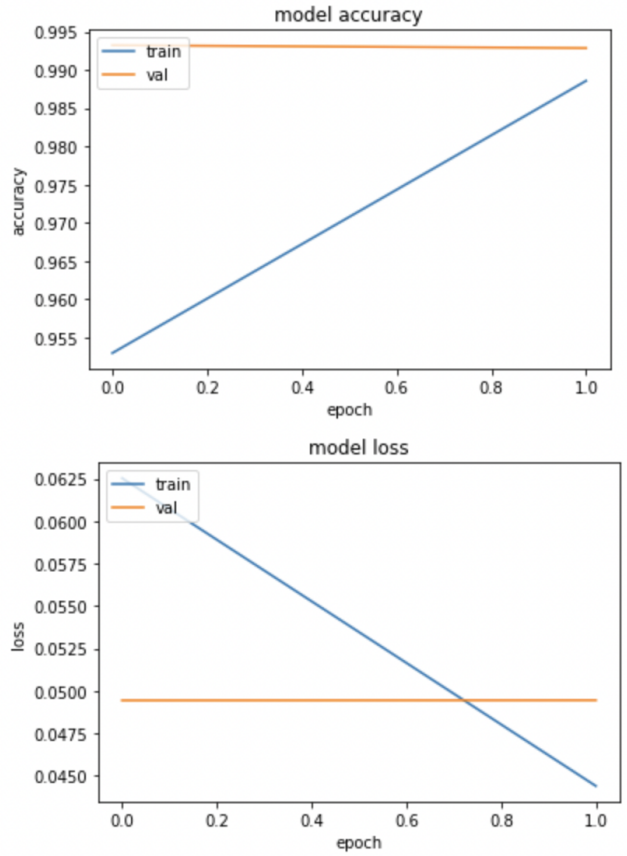

Figure 3: Model Accuracy and Model Loss of "Keras Embedding with Bidirectional LSTM"

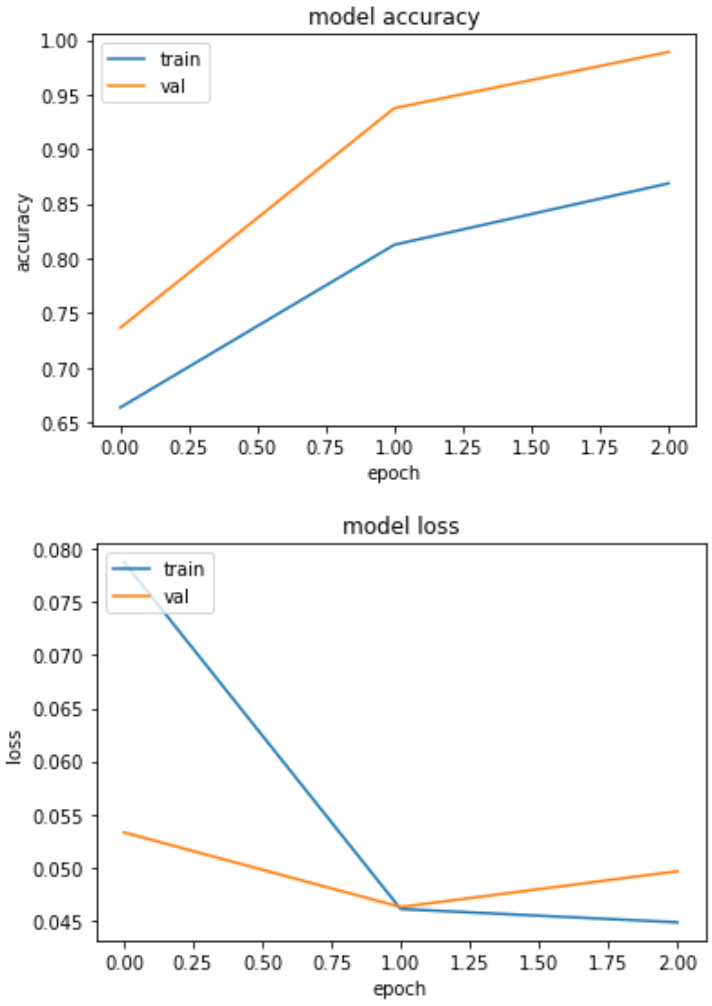

Figure 4: Model Accuracy and Model Loss of "GloVe(300 dimension) embedding with Bidirectional LSTM"

Findings:

- Deep Learning neural networks as compared to Machine Learning algorithms perform better in terms of learning the semantics and vocabulary of the input text and thus, result into a better and efficient model.
- We found that word-embedding techniques play an important role in finding and selecting the important components of text that we feed into our models.
- Also, models with GloVe embedding technique performed the best. This indicates the importance of a better word embedding technique and how much it is important to learning of a model and its performance.

These findings also answer the hypotheses questions that we had since the beginning of the project.

e. Conclusion

- In conclusion, we understood that toxic comment classification is a very nuanced task in the field of NLP and would require complex language models to label inputs. It is all the more critical to have high accuracy especially in the modern world where social media has seeped into the fabric of how people interact with the world.

- Our models performed well on the training set as well as on the given testing set, however we must make sure that it also scales well with comments and text from other domains. The project can also be expanded to include classification of toxic audio, videos, and emoticons as well. After doing this project, we understood the importance and need of a robust system to make the internet a safe place for us and our children to interact with.

Term Paper

We have discussed our project in the report above and will be using this paper to illustrate various limitations that we faced throughout the process.

Limitations with Problem Formulation and Dataset:

1. While going through the problem statement and data comments, we discovered that the problem statement stated here along with its dataset specifically takes the assumption that most of the time, the toxicity will be in terms of abusive words or curse words. But it is not always the case, since many times people try to abuse others on internet by using 1) subtle comments of negative nature 2) comments without curse or abusive words 3) comments with just curse emoticons, which might not classified by the models trained on this dataset due to the above mentioned assumption.

   This points out to the fact that there can be more classes of toxicity such as 'taunts', 'abusive emoticons' and 'hateful' which also impact the mental health of social media users, but only 6 of them are being targeted by this problem which is a limitation.

2. Along with the limitation in the problem statement, the dataset, also brings a limitation along with it. The limitation is the class-imbalance problem. This problem has been discussed in detail in the "class imbalance problem" section of "implementation/methodology" section. This limitation simply causes the model to learn more about a specific class than other classes. This might result in the trained model to be more biased or in simple language have more chance of predicting that particular class.

3. The dataset also carries a limitation with it in the nature of its source of comments. It is specifically mentioned on Kaggle that comments have been taken from Wikipedia, which is not that good source for sourcing such toxic and hateful comments. Instead, Twitter 'tweets' or instagram comments could also have been targeted as they are some of the most popular and used social media platforms.

   Also the models trained in this dataset, when applied to sites like twitter or instagram, might not be that efficient in catching toxic behavior as it might happen that the nature of the cluster of comments on wikipedia is different from the ones commonly used such as twitter or instagram.

Limitations with the Methods:

1. Limitations arose in our methods of implementing using pre-trained GloVe embeddings or those that the Keras library presented instead of training a corpus specific embedding layer. Even though the pertained vectors are powerful and cover a wide range of words with multiple meanings, the loss decay that can be observed after training embeddings from scratch is much higher as the model understands the context of the word. Especially in applications like classifying for toxicity or sentiment where a specific meaning or usage of a word can change a lot.

   For instance, the sentence “{a particular race} are swine” is obviously a toxic comment, however, the embedding of swine should reflect negative connotations and not just mean pig. A corpus specific trained embedding would converge faster as compared to pertained ones in identifying such

examples.

2. Although the LSTM recognizes sequences of text and builds a model over the vectorized corpus, adding more complex mechanisms like attention adds more information about the context especially when the comments are long and spread out over multiple sentences. Essentially, in the encoder decoder architecture of the LSTM, an extra attention layer captures the hidden state information over propagation in both directions and compresses it in a context vector. Intuitively, this makes the model understand global contexts and helps it learn the toxic nature more effectively.

Limitations with Hypothesis and Findings:

1. The first limitation that arises for the results we obtained is that, the results are obtained by training neural network models for 2-5 epochs as compared to 15-20 epochs in general. This was due to google colab GPU limitations, the place where majority of this project work is done. Since we had a lot of data, and word embedding was also performed, per epoch time was quite high, which can only be brought down by the use of GPU. Also, google colab allows only 12 hours of continuous GPU usage, which limited our hyper-parameter tuning opportunities and thus, we preferred to go for 2-5 epochs only. But the good thing is, we observed a similar or constant model performance in terms of accuracy when we changed the number of epochs from 2 to 5. Thus, though we faced hardware limitations, results were quite acceptable.

2. One of our findings discusses that deep learning neural networks usually perform better than machine learning models in this type of tasks. This finding was derived from the results that we obtained from this project work. But the catch here is, while training machine learning models, we faced "Session Crash" problem on google colab due to excessive RAM usage (colab provides 12GB RAM), which didn't allow us to use the ClassifierChain from sk-multi learn python library. If we had been able to use this, machine learning model results might have been different, which in turn might have altered the results and findings.

The prerequisites for taking your findings and applying them in practice

As such, there are no prerequisites, with the findings and applying them. Findings from this project are quite broad in nature, by which we mean that they are more like suggestive findings which can be used by anyone working on heavy text classification tasks like this in order to improve model performance and achieve better results.