# Abusive language detection using Deep learning and the derivation of context for this phenomenon

1st Ryan Roy
*ANLP Final Project*
*Indiana University - Bloomington*
ryroy@iu.edu

2nd Shruti Padole
*ANLP Final Project*
*Indiana University - Bloomington*
spadole@iu.edu

***Abstract*—This project attempts to design, and evaluate the performance of deep-learning algorithms and models (Neural networks, Recurrent neural network, LSTMs, K-means clustering) and NLP techniques like dense encoding of words to tackle the much relevant and pressing problem of abusive speech detection on social media platforms. It summarizes some of the most relevant efforts that have been made in this direction in the past, and uses their insights and design-principles to make the process smarter and more effective. It concludes with how the trained models can be exploited further in the interest of investigating the context and nature of this abusive behaviour.**

***Index Terms*—Abuse detection, Deep learning , RNN, Neural networks, Twitter database, Dense-encoding, K-means clustering**

## I. Introduction

The latest estimate is that about 4.2 billion people in the world use social media. This leads to the creation of a humongous amount of content on these platforms.

Unfortunately a lot of this content is toxic in nature and deeply problematic and threatens to adversely affect the safety and well being of all users.

Given that the scale of this problem is too big and too complex (as the interpretation of what is abusive and non-abusive could vary albeit only slightly) from individual to individual - it is inevitable to be drawn to the conclusion that a significant effort needs to be put into the research and development of a much more comprehensive and automated solution to take care of this. As in a lot of industries in the recent past - deep learning comes once again to our rescue.

### A. Goal

1) Exploration of what features can be harvested with respect to solving the abuse-detection problem from the context of the twitter dataset? What steps are involved in the generation of these features? What are some of the data-sets available online for the generation of features? Do hashtags deserve more importance than the words when it comes to the classification problem ?
2) Building and evaluating the performance of a baseline model for the task. Exploration of improvisations that can be made to the baseline to improve its performance. Incorporating insights and architectural features of state-of-the-art models. Exploration of hybrid models that take advantage of deep learning and machine learning.
3) From a business point of view, is there a way to make the whole process more proactive and less dependent on laborious human annotation ? How could we derive more information about the context of the abuse by exploiting the trained model?

## II. Literature Review

There has been a lot of pertinent work and research on this topic.In the following section we briefly talk about some of the most important works that we took inspiration from and their impact on our project

One of the most cited papers that we could come across for this task was the one titled “A Unified Deep Learning Architecture for Abuse Detection” [1].Of all the papers that we looked at - this one seemed to have the best results in terms of classification accuracy. This paper talked about a very interesting approach wherein it made use of two distinct neural networks - the first one was an RNN which was used to evaluate the text of the tweet/comment itself and to derive some insight about whether the words itself indicated a pattern of abuse/hate. The second neural network was a general feed forward network whose purpose was to evaluate the meta-data associated with the text like ”author-id”,timestamp etc. The reason why an RNN was not used here was because there was no sequential relationship between the various elements of metadata. Finally the output from the two aforementioned networks was fed into a final dense layer so that it could figure out on its own how much weightage was to be attributed to the meta-data and the text.

The second most important paper that we consulted was the one titled ”Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter” [2]. This served as our base paper and it laid out very efficiently an holistic way to approach this problem, analyse the result to derive some context to it and to summarize what worked and what did not.

The paper ”Detecting Offensive Language in Social Media to Protect Adolescent Online Safety” [3] delved a little bit deeper into the matter and tried to look at things not just from a syntactical point of view but also from a semantic point of view. One very interesting idea that makes this paper unique was the discussion about ”intensifiers”. This refers to a special type of semantic dependency between two words in a tweet in

which the dependent word that amplifies the meaning of the independent word. Assuming that the inherent nature of abuse is loud and exaggerated, capturing this kind of dependency would seem quite useful.

The paper "Multiple Word Embeddings for Increased Diversity of Representation" [4] talks about the usage of multiple embeddings to accentuate the efficiency of the classification task and is relevant to the work being carried out by the authors as it involves deriving useful cues from more than one subtype of sequential data forms. For instance the sequential data found in tweets and the sequential data found in the "user-bio" and it could thus be assumed in the context of this word that it would be very useful to use multiple word embedding layers instead of one.

## III. Methodology

### A. DataSet

The collection of relevant data is always a challenge and do have done so successfully is often the most crucial task involved in the process of research. The basic level access to the twitter API is one that helps you extract the tweet-content and the author-id using just the tweet-id. There was a predefined dataset that was developed by the authors of "[2]" which had labels corresponding to about 10k tweets. The tweets were missing but the tweet-ids could be used to extract the tweets using just one of the basic-level access.

The authors also contacted twitter requesting them the access to "elevated level access" that granted them additional data-points for every tweet that involved elements of metadata like "Geo-location","time when tweet was sent", and information about the user like "User-bio", "Retweets", "No. of followers" and so on for the aforementioned 10k tweets.

However this would still turn out to be a rather small dataset as the models that were being used to train this data on were rather powerful and would presumably perform the undesirable action of memorizing the dataset instead of learning from it. A dataset called "MeTooHate" containing 800k tweets, a basic set of meta-data elements for all tweets, and their binary labels (1 for hateful and 0 for neutral) were discovered scouring the internet. The size made it the default choice for further analysis. Two other datasets for the purposes of validation were called Founta and Razavi.They too contained just the bare-bone information such as tweet-text data and hate vs non-hate binary labels.

### B. Model

*1) Base LSTM model:* The authors begin by loading the 800k tweets long "MetooHate" dataset from the csv file into a pandas data-frame. Then the next step is to pre-process the tweet text to bring it into a more standard format. The authors build their own vocabulary using all the tweets and the time taken to complete this task is comparable to the time taken to train the model. The model has a tf.keras.layers.Emebedding layer as its first layer and the inputs after being tokenized using the vocabulary passes through this layer first. A predefined function is used for both the tokenization and the padding steps involved. The model itself consists of an embedding layer, followed by an LSTM layer with the width of 16 nodes, it then has a dense fully connected layer that has 8 nodes and followed by another dense layer with just the single node with a sigmoid activation function. The loss function used is "binary cross entropy" as the task is binary classification. The model is trained for approximately 30 epochs. This model serves as the base model for all the purposes of this paper and the insights about data drawn from this model comes in handy for the design of all other models.

*2) Improvised BiLSTM model:* One major observation from the base model was that the model was seemingly over fitting pretty fast. So one major change that we made to the second model other than making the LSTM layer a BiLSTM layer was to make the model deeper and leaner (lesser nodes per layer). Gradient clipping was introduced to work around the exploding gradient problem that was observed in the first model. Also the no. of embedding dimensions was reduced from 50 to 5 in this model. The rest of the steps involved in data-preparation was the same and the model was trained for about 10 epochs and early stopping was used so that it does not over-fit again.

```
Model: "sequential_19"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_20 (Embedding)     (None, None, 5)           686780
_________________________________________________________________
bidirectional_17 (Bidirectio (None, 10)                440
_________________________________________________________________
dense_48 (Dense)             (None, 4)                 44
_________________________________________________________________
dense_49 (Dense)             (None, 8)                 40
_________________________________________________________________
dense_50 (Dense)             (None, 16)                144
_________________________________________________________________
dense_51 (Dense)             (None, 5)                 85
_________________________________________________________________
dense_52 (Dense)             (None, 1)                 6
=================================================================
Total params: 687,539
Trainable params: 687,539
Non-trainable params: 0
```

Fig. 1. Architecture of the BI-LSTM model

*3) Mesh model:* The authors also build a third proof-of-concept model that takes advantage or other non-linguistic data-points associated with the data like time, space, user, hashtags, user-bio etc. This is inspired by the model developed as part of the work - "A Unified Deep Learning Architecture for Abuse Detection"[1]. The only change here is that instead of having 2-neural network based mesh model, one dealing with meta-data and one dealing with text data, we have 4 neural networks based mesh model, one to deal with the text data, one to deal with the hashtags, one to deal with the user-bio statement and one to deal with other elements of the metadata.

*4) K-means clustering of Dense-encoded word vectors:* The dense-layer encodings carry within themselves a task-contextualized representation of all the words within the vocabulary. So in one sense they represent in a higher dimensional space the similarity between these words or the

Model: "model_89"

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_90 (InputLayer) | [(None, 75)] | 0 | |
| input_91 (InputLayer) | [(None, 75)] | 0 | |
| embedding_83 (Embedding) | (None, 75, 10) | 202180 | input_90[0][0] |
| embedding_84 (Embedding) | (None, 75, 10) | 49010 | input_91[0][0] |
| bidirectional_41 (Bidirectional | (None, 20) | 1680 | embedding_83[0][0] |
| bidirectional_42 (Bidirectional | (None, 20) | 1680 | embedding_84[0][0] |
| dense_353 (Dense) | (None, 4) | 84 | bidirectional_41[0][0] |
| dense_358 (Dense) | (None, 4) | 84 | bidirectional_42[0][0] |
| input_92 (InputLayer) | [(None, 75)] | 0 | |
| input_93 (InputLayer) | [(None, 75)] | 0 | |
| dense_354 (Dense) | (None, 8) | 40 | dense_353[0][0] |
| dense_359 (Dense) | (None, 8) | 40 | dense_358[0][0] |
| dense_363 (Dense) | (None, 64) | 4864 | input_92[0][0] |
| dense_367 (Dense) | (None, 64) | 4864 | input_93[0][0] |
| dense_355 (Dense) | (None, 16) | 144 | dense_354[0][0] |
| dense_360 (Dense) | (None, 16) | 144 | dense_359[0][0] |
| dense_364 (Dense) | (None, 32) | 2080 | dense_363[0][0] |
| dense_368 (Dense) | (None, 32) | 2080 | dense_367[0][0] |
| dense_356 (Dense) | (None, 10) | 170 | dense_355[0][0] |
| dense_361 (Dense) | (None, 10) | 170 | dense_360[0][0] |
| dense_365 (Dense) | (None, 10) | 330 | dense_364[0][0] |
| dense_369 (Dense) | (None, 10) | 330 | dense_368[0][0] |
| dense_357 (Dense) | (None, 1) | 11 | dense_356[0][0] |
| dense_362 (Dense) | (None, 1) | 11 | dense_361[0][0] |
| dense_366 (Dense) | (None, 1) | 11 | dense_365[0][0] |
| dense_370 (Dense) | (None, 1) | 11 | dense_369[0][0] |
| tf.concat_17 (TFOpLambda) | (None, 4) | 0 | dense_357[0][0]<br>dense_362[0][0]<br>dense_366[0][0]<br>dense_370[0][0] |
| dense_371 (Dense) | (None, 64) | 320 | tf.concat_17[0][0] |
| dense_372 (Dense) | (None, 1) | 65 | dense_371[0][0] |

Total params: 270,403
Trainable params: 270,403
Non-trainable params: 0

Fig. 2. Architecture of the mesh model

lack thereof with respect to a hate-no hate context. This encodings serve as a powerful tool to further investigate what patterns form within these higher dimensional spaces. The authors perform a Eigen-decomposition on the embedding layer weights to isolate the maximum information containing dimensions and then perform a k-means clustering on the result to evaluate if sub-groups of words emerge and if so - what do they collectively represent in a semantic sense?

## IV. Results

### A. Base model and improvised LSTM model

The base model was prone to over-fitting and the validation accuracy was declining and the training accuracy was nearing 1 after the first 10-15 epochs.

```
Epoch 8/10
697/697 - 22s - loss: 0.0045 - accuracy: 0.9991 - precision: 0.9991 - recall: 0.9991 - val_loss: 0.9013 - val_accuracy: 0.8495
- val_precision: 0.8508 - val_recall: 0.8491
Epoch 9/10
697/697 - 22s - loss: 0.0043 - accuracy: 0.9989 - precision: 0.9989 - recall: 0.9989 - val_loss: 0.9717 - val_accuracy: 0.8459
- val_precision: 0.8465 - val_recall: 0.8455
Epoch 10/10
697/697 - 22s - loss: 0.0064 - accuracy: 0.9983 - precision: 0.9983 - recall: 0.9983 - val_loss: 0.9999 - val_accuracy: 0.8451
- val_precision: 0.8470 - val_recall: 0.8443
```

Fig. 3. Accuracy,Precision and recall of Base model

Once a BiLSTM layer was introduced in the place of the LSTM layer and some of the insights mentioned in the section above this were applied to the model, we could bring it to give an accuracy of 96 percent as validation accuracy or test accuracy which is 96 percent of 800k tweets predicted right.

### B. State of art inspired mesh model

Since the data wasn't available to train the model on the bigger dataset (800k), we resorted to training it on the smaller

```
Epoch 6/8
15346/15346 [==============================] - 245s 16ms/step - loss: 0.3576 - accuracy: 0.8847 - precision: 0.0000e+00 - recal
l: 0.0000e+00 - val_loss: 0.3585 - val_accuracy: 0.8842 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 7/8
15346/15346 [==============================] - 245s 16ms/step - loss: 0.2680 - accuracy: 0.9063 - precision: 0.8234 - recall:
0.2393 - val_loss: 0.1675 - val_accuracy: 0.9585 - val_precision: 0.8088 - val_recall: 0.8400
Epoch 8/8
15346/15346 [==============================] - 245s 16ms/step - loss: 0.1947 - accuracy: 0.9344 - precision: 0.8575 - recall:
0.5174 - val_loss: 0.1585 - val_accuracy: 0.9614 - val_precision: 0.8974 - val_recall: 0.7530
```

Fig. 4. Accuracy,Precision and recall of improvised BiLSTM model

one which had about 10k tweets but we were able to extract the additional data points like username, hashtags and so on. Despite its small size, we were able to build a functional proof of concept mesh model and acquire an accuracy of 87 percent on unseen test data.

**Testing the performance on unseen test data**

```
metrix = model.evaluate([inputA_test,inputB_test,inputC_test,inputD_test],y_test)
```

```
65/65 [==============================] - 1s 3ms/step - loss: 0.8658 - accuracy: 0.8754
```

A test-accuracy of 87 percent is obtained!

Fig. 5. Accuracy of mesh model

### C. K-means clustering of dense-layer encodings

In this section, we discuss the process outcomes of the K-means clustering process. For the purpose of being able to plot it onto a graph - we first did an Eigen-decomposition and reduced the dimensionality of the vectors from 5 to 2. Later we used a standard K-means algorithm to assign each vector to a randomly selected centroid and then recalculate the position of the centroid based on all the vectors that were in that specific cluster. 4 clusters were formed and the results can be seen in the diagrams below.

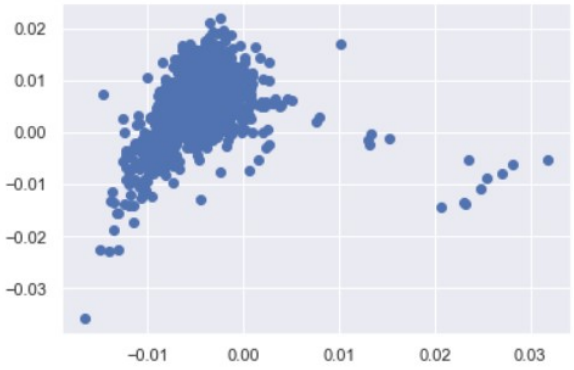


Fig. 6. Before the clusters are formed

Once the clusters are formed, we can use their index-numbers to retrieve the words from our original vocabulary and assess their similarity with respect to the task at hand

### D. Benefits

The BiLSTM based model is easy to create and can be trained relatively easily.It will only get better and better with more data. It can be extended to other binary classification tasks as well by giving it the appropriate data. It forms a very good baseline for text classification tasks associated

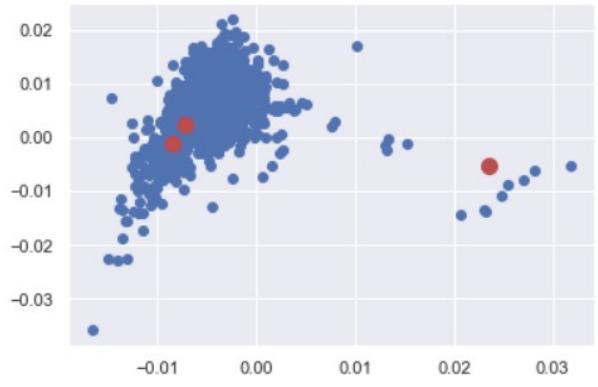


Fig. 7. Calculation of centroids

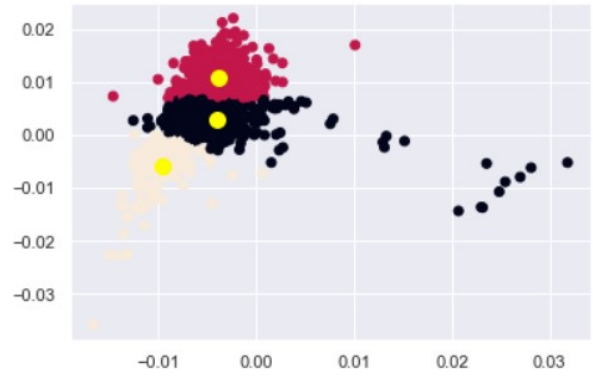


Fig. 8. Final clusters

```
In [254]: group_1_words[:10]
          'eg:',
          'hollywoodpost-',
          'prison-rape',
          'imminently',
          'hortons',
          'mortgage',
```

Fig. 9. Group 1 words

```
In [255]: group_2_words
          'shading',
          'assoviated',
          '&amp;war',
          'naftas',
          'hidden)',
          '18333133149',
```

Fig. 10. Group 2 words

with tweets as it draws up important conclusions about the following decision-points

- Dimensionality of dense-layer encodings
- Architectures that work fine and others
- Loss function, optimizers, number of nodes per layer
- Process of tokenization and sequence padding
- Right number of epochs to train it for

The data obtained here can be extrapolated in order to develop the baseline for tasks that are much more complex and data-intensive than this by taking these decisions as a reference point.

From an industry perspective, the clustering of dense layer encodings can be seen to have the following advantages:

- Alternative to human annotation based approach which is taxing, unreliable
- Closed loop design where the output can be taken into consideration for the input
- Communication with the model is established and the process is no longer a giant black-box
- Can give the edge to business helping them be proactive instead of reactive when dealing with abusive tweets and abusive authors

## ACKNOWLEDGMENT

We would like to express our gratitude to our professor Zeeshan Sayyed and Sandra Kübler for providing the the platform to showcase our project and skills.Their guidance and insights have been of extreme significance for this project.

## REFERENCES

[1] Antigoni-Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, Ilias Leontiadis, “A Unified Deep Learning Architecture for Abuse Detection” WebSci ’19, June 30–July 3, 2019, Boston, MA, USA

[2] Zeerak Waseem, Dirk Hovy, “Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter”

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

[4] Brian Lester, Daniel Pressel, Amy Hemmeter, Sagnik Ray, Choudhury and Srinivas Bangalore, ”Multiple Word Embeddings for Increased Diversity of Representation”