Machine Learning Engineer Nanodegree

Capstone Project

Nishkarsh Vardhan
May 26th, 2018

I. Definition

Project Overview

Toxic Comments are frequent nowadays, with ease of accessibility for internet around the world. Hatred among people are pretty common due to some cultural sentiments or even they try to bully others view to show their dominance all over the public space. In today's scenario it's very important to understand the views and values of others. With help of discussions we can solve major problems very easily. Sometimes these discussion also lead to innovations and revolutions which can create better future.

In this project, I am trying to determine the toxic comments on Wikipedia page. This will help in figuring out and eradicating toxic words from comments on any online pages or tweets later on. I will be using baseline to be a linear model like Linear Regression or Logistic Regression. Then will move to advance model like Recurrent Neural Network based LSTM(Long Short Term Memory) model. Comparisons will then be made on existing models and predictions will be made on test data set with proper evaluation and accuracy.

This project will use dataset from Kaggle available at -

https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data.

This is a data collected by Google and Jigsaw, it's a data with large number of Wikipedia comments which have been labeled by human raters for toxic behavior.

Problem Statement

Toxic Comment Classification is the major goal for this project. We are provided with a large number of Wikipedia comments which have been labeled by human raters for toxic behavior. The types of toxicity are:

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

We need to create a model which predicts a probability of each type of toxicity for each comment.

I will use Supervised learning and Neural Networks to train the model and help to detect accurateness of toxic comments.

I will start by working with basic models like supervised models to check if I am getting correct output then will tune the model in more elaborate manner with Neural Networks as a part of unsupervised learning. This Problem is frequent nowadays so it can be easily worked upon and is quantifiable also. Basically, I will quantify with my models the toxicity levels in Wikipedia comments for each labels mentioned above.

So, as discussed earlier I will start with the linear model like Logistic Regression to create a baseline as it will cover log loss function internally. I will also try to compare with Linear Regression model if required (used in Perspective API from Google). After that I will apply deep learning model on problem with Recurrent Neural Network based LSTM(Long Short Term Memory) model, this will help me in proper classification of complete sentences.

Neural Nets are effective way to actually figure out the exact words which are toxic. So, with the help of word embedding for which I am using GloVe(glove.6B.50d) (Global Vector for Word Representation, ref - https://nlp.stanford.edu/projects/glove/) I will cover complete model can be said as LSTM with Glove model. Anticipated solution will be to effectively find and clearly out toxic comments from online page. This will send warning to person involved in that toxicity. So it will help in warning as well as alerting the public about the hatred or bullying.

Metrics

I will be using two baselines in order to get a better understanding of the problem from both a general machine-learning perspective(Logistic Regression) and a deep-learning perspective.(RNN based LSTM with GloVe).

Evaluation metric which I have used in project is accuracy score. I chose this because it computes subset accuracy that is the set of labels predicted for a sample must exactly match the corresponding set of labels. Below is the code for accuracy score:-

sklearn.metrics.**accuracy_score**(*y_true, y_pred, normalize=True, sample_weight=None*)

All the values are predicted based on accuracy score only. Reason for usage of accuracy score is log loss value is pretty much reduced and there will be improved validation score for every model.

Apart from above metric there is one more metric which I used in my deep learning model which is binary_cross entropy metric. Each class is solved simultaneously using the binary_crossentropy metric, performance gains will be achieved not only by increased model complexity but also by a more complex objective function. Binary cross entropy for multi-label classification can be defined by the following loss function:

$$-1N\sum i=1N[yilog(y\hat{}i)+(1-yi)log(1-y\hat{}i)]$$

The log_loss values were pretty much precise for these metrics that is why I used them. In future, I can try for ROC-AUC metric too which seems to predict a good score for these type for problems.

Details Explaination of Algorithm, Technique and metrics :-

For Supervised Learning model like Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

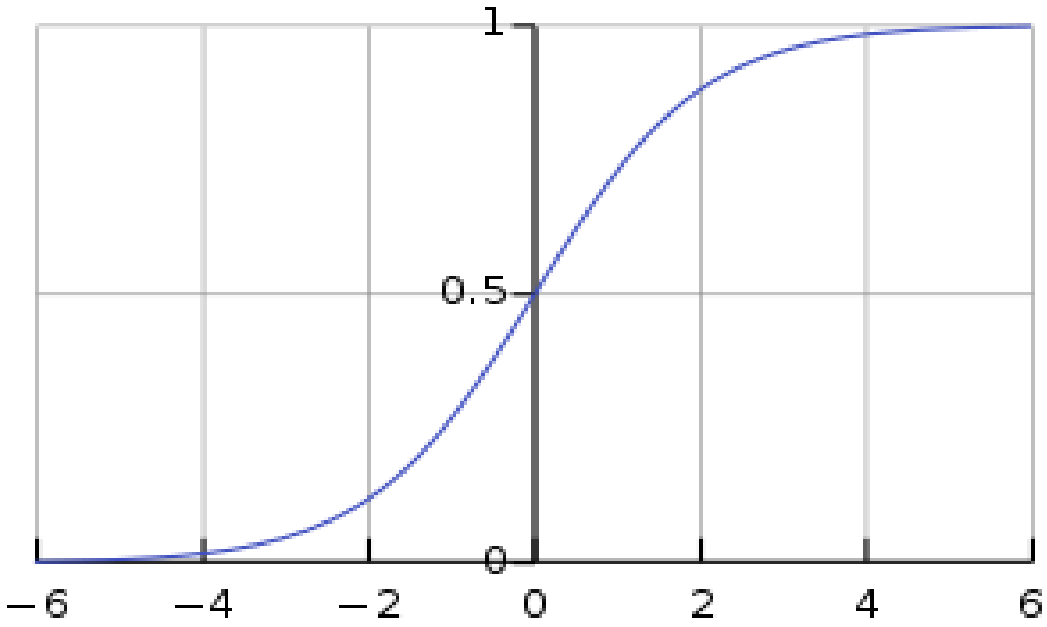
$$f(x) = L / (1 + e\hat{}(-k(x-x0)))$$

e=natural logarithm base

x0 = x-value of sigmoid's midpoint.

L=curve's maximum value

k=steepness of curve



This implementation can fit binary, One-vs- Rest, or multinomial logistic regression with optional L2 or L1 regularization.As an optimization problem, binary class L2 penalized logistic regression minimizes the following cost function:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1).$$

Similarly, L1 regularized logistic regression solves the following optimization problem

$$\min_{w,c} ||w||_1 + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1).$$

The solvers implemented in the class Logistic Regression are “liblinear”, “newton-cg”, “lbfgs”, “sag” and “saga”:

Here I am using sag solver. The “sag” solver uses a Stochastic Average Gradient descent. It is faster than other solvers for large datasets, when both the number of samples and the number of features are large.

For deep-learning perspective LSTM with GloVe model.The formula for minimizing the mean binary cross entropy loss across the training set.

In binary classification, where the number of classes M equals 2, cross-entropy can be calculated as:

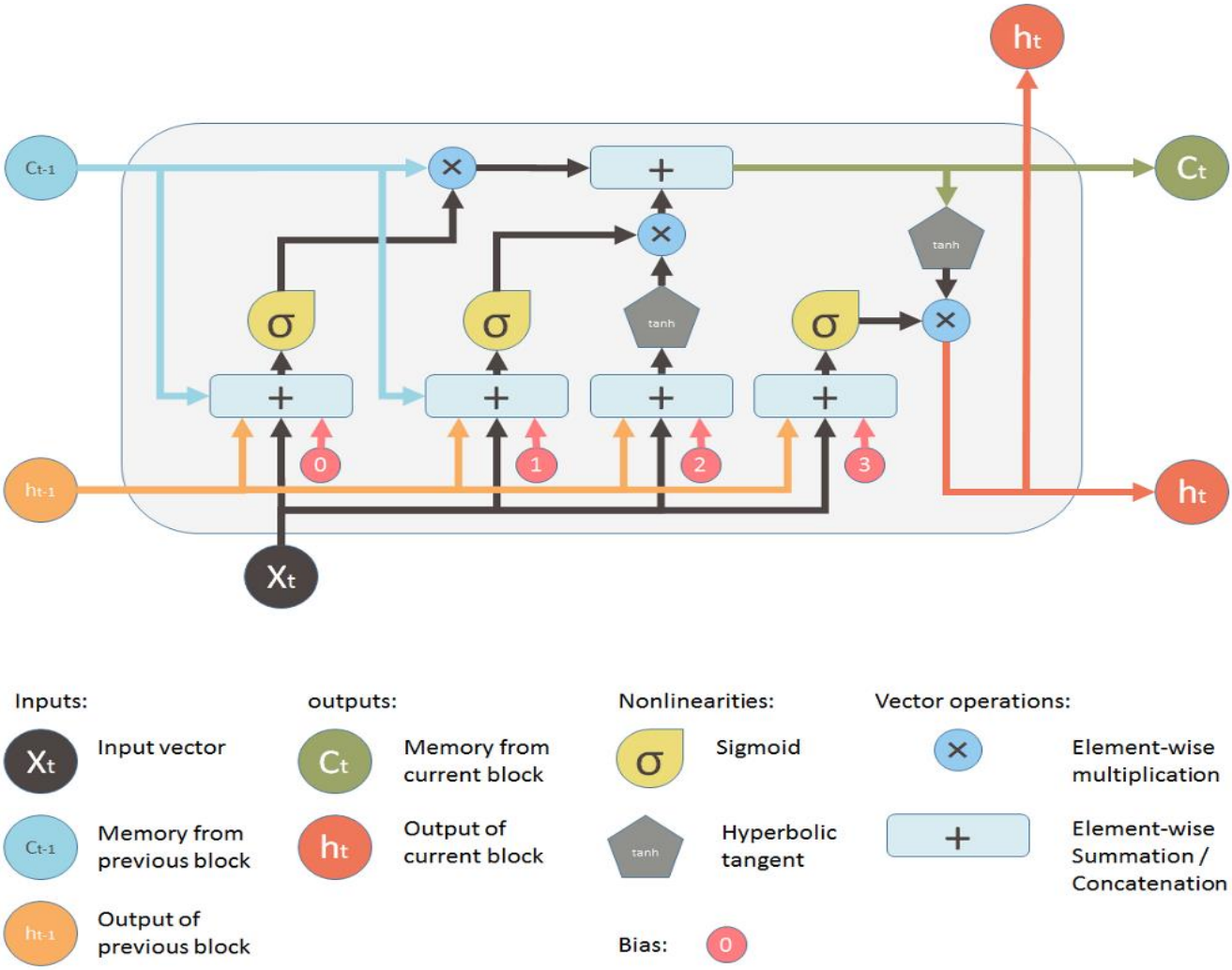
$$-(y\log(p)+(1-y)\log(1-p))$$

If M >2 (i.e. multiclass classification), we calculate a separate loss for each class label per observation and sum the result.

$$-\Sigma c=My(o,c)\log(p(o,c))$$

where -

M - number of classes (dog, cat, fish)
log - the natural log
y - binary indicator (0 or 1) if class label cis the correct classification for observation o.
p - predicted probability observation o is of class c
For primary model, I used a recurrent neural network (RNN) with a long short-term memory (LSTM) cell . I will use word based embedding and will test for word-level embedding on the LSTM for efficiency reasons.
Recurrent Neural Network with Long Short-Term Memory cell LSTMs in particular outperform RNNs with GRU cells on problems where understanding a broader, long-term context is important, and the LSTM cell also prevents the problem of a vanishing gradient. LSTMs are commonly used in sentiment classification because of their ability to use longer-term contexts. LSTMs uses sigmoid functions for the classification.



Following formulas internally used for RNN based LSTM model :-

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t &= f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
h_t &= o_t tanh(c_t)
\end{aligned}
$$

Where internally it describes how a layer of memory cells is updated at every timestep t. In these equations:
i(t) - input gate at time t.
x(t) - is the input to the memory cell layer at time t.
h(t) - hidden layer output at time t.
o(t) - final output at time t.
f(t) - the activation of the memory cell's forget gates at time t.
c(t) - the memory cells' new state at time t
W(xi),W(hi),W(ci),W(xf),W(hf),W(cf),W(xo),W(ho),W(co) - represent weight matrices.
b(i),b(f),b(v),b(0) - are bias vectors.

Apart from above metrics calculated Average, Mean and Standard Deviation using inbuilt libraries.

II. Analysis

Data Exploration

This project will use dataset from Kaggle available at "https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data".
This is a data collected by Google and Jigsaw, it's a data with large number of Wikipedia comments which have been labeled by human raters for toxic behavior.
Data set comprises of 2 files, these are:-
train.csv - the training set, contains comments with their binary labels
test.csv - the test set, you must predict the toxicity probabilities for these comments.
To deter hand labeling, the test set contains some comments which are not included in scoring.
Here, we are provided with a large number of Wikipedia comments which have been labeled by human raters for toxic behavior. The types of toxicity are:

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

Requirement is to create a model which predicts a probability of each type of toxicity for each comment.

Data provided consists of below values:-

Read and Segregate the data

	category	no_of_comments
0	toxic	15294
1	severe_toxic	1595
2	obscene	8449
3	threat	478
4	insult	7877
5	identity_hate	1405

This is how a classification done and are present in training data and we have to refine this data to figure predict the correct classification based upon our applied models.

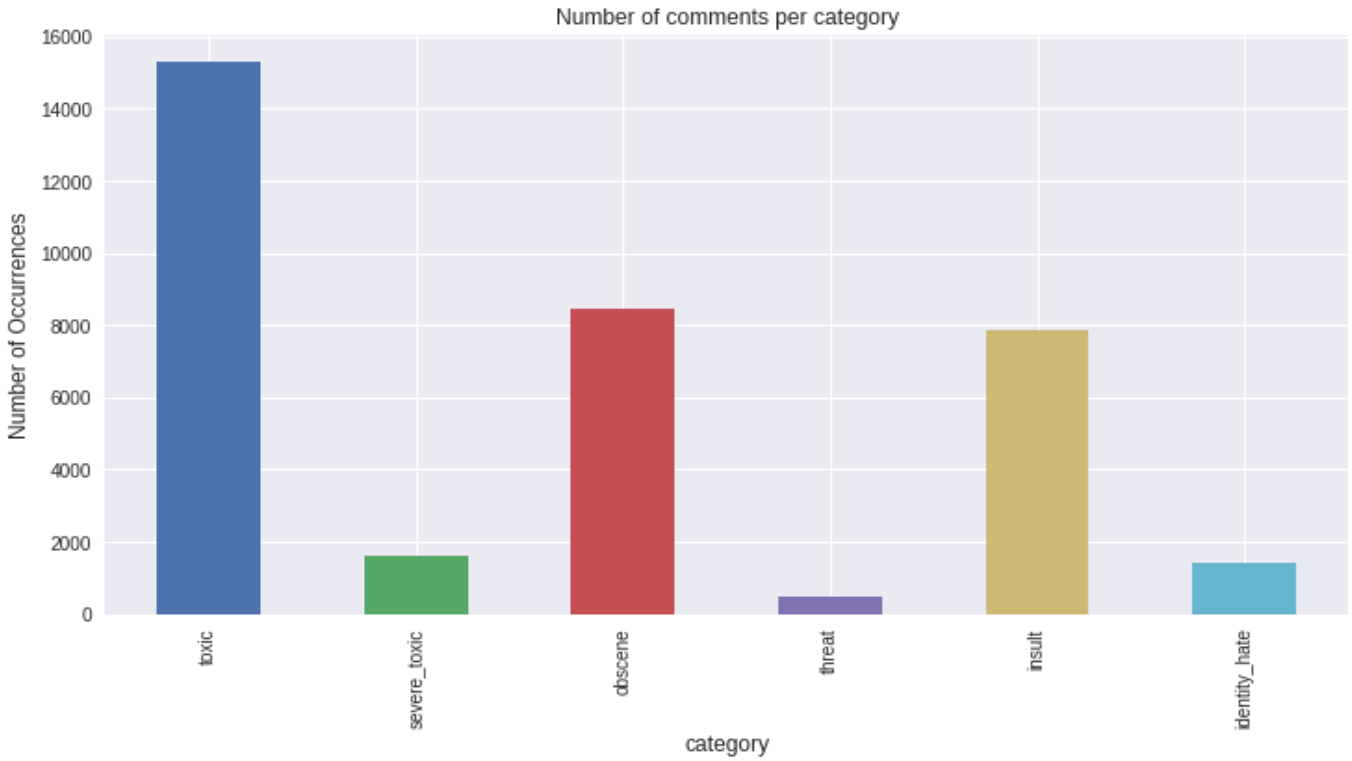
Now, Let's elaborate more about the dataset.

It has total 6 target variables namely – toxic, severe_toxic, obscene, threat, insult and identity_hate. As a calculation and study part it's easily observable that data has following information regarding unlabeled comments:-
1) Finding comments from data which are not labelled at all!!
Percentage of comments that are not labelled:89.83211235124176
Number of missing comments in comment_text:
0
2) Let's find the mean, standard deviation and max of the above data.
(394.0732213246768, 590.7202819048919, 5000)
3) Data basically consists of total 10734904 words.532299 unique words.10 Most common words in the dataset:"the" "to" "of" "and" "a" "I" "is" "you" "that" "in"
4) Total Classified comment_text 35098 words.
5) Sample data looks like below:-
Checking data Comment
#1: Explanation
Why the edits made under my username Hardcore Metallica Fan were reverted? They weren't vandalisms, just closure on some GAs after I voted at New York Dolls FAC. And please don't remove the template from the talk page since I'm retired now.89.205.38.27
Label #1: [0 0 0 0 0 0]
Comment #2: D'aww! He matches this background colour I'm seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC)
Label #2: [0 0 0 0 0 0]
Comment #3: Hey man, I'm really not trying to edit war. It's just that this guy is constantly removing relevant information and talking to me through edits instead of my talk page. He seems to care more about the formatting than the actual info.
Label #3: [0 0 0 0 0 0]
Comment #4: " More I can't make any real suggestions on improvement - I wondered if the section statistics should be later on, or a subsection of ""types of accidents"" -I think the references may need tidying so that they are all in the exact same format ie date format etc. I can do that later on, if no-one else does first - if you have any preferences for formatting style on references or want to do it yourself please let me know.

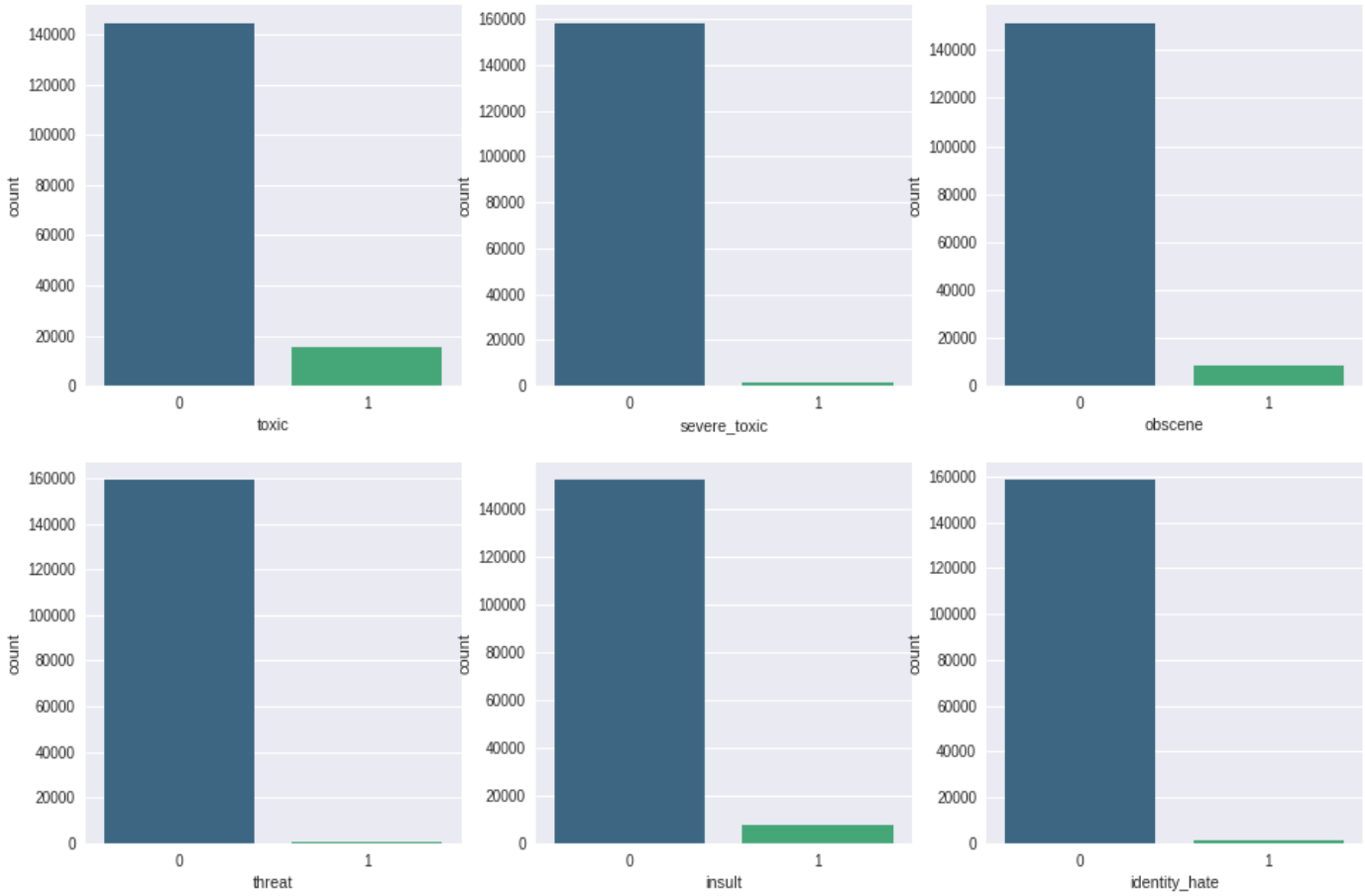
There appears to be a backlog on articles for review so I guess there may be a delay until a reviewer turns up. It's listed in the relevant form eg Wikipedia:Good_article_nominations#Transport "
Label #4: [0 0 0 0 0 0]
Comment #5: You, sir, are my hero. Any chance you remember what page that's on?
Label #5: [0 0 0 0 0 0]

Exploratory Visualization

Visualization of data based on category can be seen below. We can clearly see number of occurrences for toxic comments is higher than others.



Below is the graph for individual category of comments to frquency it occurred.



Algorithms and Techniques

Algorithms used for solving this classification problem initially is Supervised learning using Logistic Regression to create a benchmark model then to classify it further used Recurrent Neural Network based LSTM (Long Short Term Memory) Model. Logistic Regression is used to create the benchmark as it is a supervised learning model. I will be using scikit learn library to fit the Logistic Regression. For Deep Learning model which is based on Recurrent Neural Network to train and fit the model I have used LSTM API from Keras. Library.

Techniques used for solving the problem is basically stepwise we can see below :-

Stage 1:-

First Stage of the project will be to download and preprocess the data provided from wikipedia comments. Comments might contain multiple acronyms, emoticons and unnecessary data like urls, images etc. So, we need to first preprocess the data to represent correct emotions of public. Now, here I will do filtering like using tokens (splitting individual words based on space and symbols) for list of words. Remove unnecessary words which do not show any emotions. Remove unnecessary symbols and common positive words as per understanding.

Stage 2:-

Second Stage of the project will be filtering and tokenizing the words. So here what I will do is completely filter out all unnecessary words and labels, after that make them all into lower case so that my models

should give perfect accuracy scores at the end and not getting confused between same words. So, Tokenizing the limited words will be easier. I will allocate unique token to each word. Then I will be using GloVe(glove.6b.5d) data to basically embed my complete data . Glove(Global Vectors for Word Representation) is basically a unsupervised learning algorithm for obtaining vector , this will define each word into vectors. Training in GloVe is performed on aggregrated global word-word occurrence statistics from corpus, and the resulting representations showcase interesting linear substructures of word vector space.

Stage 3:-

Third Stage of the project will be to implement the above values and plugin to our models for learning purpose. Here first I will implement Linear Regression and based on the results, I will compare and train my deep-learning model which is RNN based LSTM. Long Short-term memory cell will be helpful if words are recurring more frequently in comments which actually happens a same word keeps on coming to hurt the sentiments. Logistic Regression will use supervised model so we will get approximate values but with Recurring Neural Network we will get exact values and words that hurt sentiments.

Recurrent Neural Network is a kind of self learning model from previous input. There is a special kind of model which takes care of learning based on memory which is LSTM(Long Short Term Memory) model. I have extended my neural network model further to prove that this model is able to detect toxicity level and also rightly classify the toxicity level which is not provided by any model.

Stage 4:-

Fourth Stage of the project will be to test the data based on given training values. I will try to test with limited set of data on trained model and will check if both the models are providing sufficient results or not. Also with random sentence I will check how much toxicity level it has.

Benchmark

A baseline mode was provided based on earlier perspective API(from Google) results. I have generated those baseline results and the values obtained from data are below:-

Finding the base level accuracy for each label from data

Base Level Accuracy for Each Label

Category	Accuracy Level Percent
Toxic	90.41555169799024
Severe_toxic	99.00044494300343
Obscene	94.7051782592075
Threat	99.70044682304429
Insult	95.06363938309592
Identity_hate	99.11951419744189

The above data is basically a linear regression model already given as a part of test data. Benchmark model is given so that we can compare our result and provide the best performing model for the problem.

III. Methodology

Data Preprocessing

The project starts with download and preprocess of data provided in Wikipedia comments by Kaggle. I started by storing the target in a variable and splitting the train and test datasets. Comments might contain multiple symbols, punctuations and unnecessary data like URLs, etc. So, we need to first preprocess the data to filter out correct words. First is to divide the comments(comment_text) into category wise(column wise), then remove the special characters and common words known using regular expressions. After that all characters were made into lowercase characters. Remove unnecessary symbols and common positive words as per understanding like Remove irrelevant characters (!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n) and convert all letters to lowercase (HeLlO -> hello). This should give perfect accuracy scores at the end and not getting confused between same words. So, Tokenizing the limited words will be easier.

After filtering of data from categories and removing common words and symbols, I tokenized the words using TfidfVectorizer. This process transforms text to feature vectors that is used as input to estimator.

Now, after this let's train our model to learn the vocabulary from the feature vectors, then use it to create a document-term matrix. Then the test data is transformed into document-term matrix as well.

The data representation for vocabulary is one-hot encoding where every word is transformed into a vector with a 1 in its corresponding location. For example, if word vector is [hello, what, is, your, name] and the word we are looking at is "your", the input vector for "your" would just be [0, 0, 0, 1]. This works fine unless vocabulary is huge - in this case, it would end up with word vectors that consist mainly of a bunch of 0s. To get rid of this I used GloVe(Global Vectors for Word Representation) algorithm. GloVe(Global Vectors for Word Representation) is basically a unsupervised learning algorithm for obtaining vector , this will define each word into vectors. Training in GloVe is performed on aggregated global word-word occurrence statistics from corpus, and the resulting representations showcase interesting linear substructures of word vector space. This is also called as Embedding of words.

After preprocessing, vocabulary size drops to compact size of 210,337 with a max comment size of 371 words and an average comment size of about 68 words per sentence. This data is pretty much in good shape to get trained.

Implementation

Algorithms and techniques implemented for the given data are Supervised Learning based Logistic Regression and Deep Learning based RNN with LSTM model.

For Logistic Regression solver used is sag which is used for huge data and process them pretty fast. For RNN based LSTM mode I have used Keras library it has LSTM API to fit the model and the solver used is ADAM. Based on data we got vectors from embedding with the help of GloVe it was quite fast to classify the data correctly with precision.

Let's discuss about models in detail -

1) Logistic Regression :-

After Data preprocessing we directly trained our model with "sag" solver of logistic regression. SAG – Mark Schmidt, Nicolas Le Roux, and Francis Bach

Minimizing Finite Sums with the Stochastic Average Gradient https://hal.inria.fr/hal-00860051/document, itis much faster in terms of convergence.

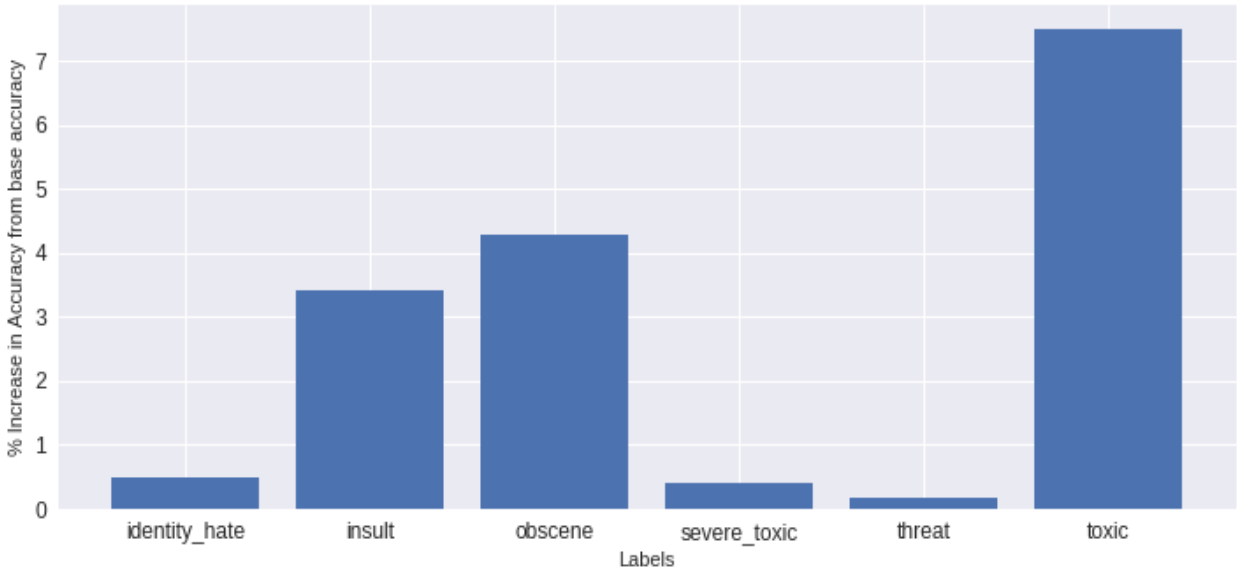
Since the target variable is more than one, we will use for loop and apply our model and then we will see the prediction and accuracy of our model. API used for Logistic Regression is:-

class sklearn.linear_model.LogisticRegression(penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='liblinear', max_iter=100, multi_class='ovr', verbose=0, warm_start=False, n_jobs=1)

Logistic Regression used as supervised learning model and comparison made using existing baseline values (from given data). After training the data with Logistic Regression we can see a increase in accuracy.

Catergory	Baseline Percent	Logistic Regression Percent	Increase in percent
Toxic	90.41555169799024	97.91127460503475	7.4957229070445095
Severe_toxic	99.00044494300343	99.41154721095938	0.41110226795595395
Obscene	94.7051782592075	98.99104473870565	4.285866479498154
Threat	99.70044682304429	99.87341058212333	0.17296375907903894
Insult	95.06363938309592	98.46400661774382	3.400367234647902
Identity hate	99.11951419744189	99.60268469834745	0.48317050090555824

Plot of results for change are below



2) Deep Learning (Recurrent Neural Network) based LSTM (Long Short Term Memory) model :-
The RNN type model used here is the LSTM model. This model is attractive because the individual cell states in the model have the ability to remove or add information to the cell state through gates layers. This is useful in practice because it allows the model to remember insights derived from words throughout the comment. The LSTM model consist of one densely connected layer with 60 units across the concatenated word vectors for each of the words in the comment.

Solver used for the model is sequential with below code fragment: -

```
from keras.models import Sequential from keras.layers import Dense, Activation model = Sequential([ Dense(32, input_shape=(784,)), Activation('relu'), Dense(10), Activation('softmax'), ])

optimizer = optimizers.Adam(lr=0.001)   ##Adam is much faster optimizer for large amount of data.

model.compile(loss='binary_crossentropy',        ## binary cross entropy returns true/false and much better in terms of loss function.
        optimizer=optimizer,
        metrics=['accuracy'])
```

Keras has an API related to LSTM which can be defined as below: -

```
keras.layers.LSTM(units, activation='tanh', recurrent_activation='hard_sigmoid', use_bias=True, kernel_initializer='glorot_uniform', recurrent_initializer='orthogonal', bias_initializer='zeros', unit_forget_bias=True, kernel_regularizer=None, recurrent_regularizer=None, bias_regularizer=None, activity_regularizer=None, kernel_constraint=None, recurrent_constraint=None, bias_constraint=None, dropout=0.0, recurrent_dropout=0.0, implementation=1, return_sequences=False, return_state=False, go_backwards=False, stateful=False, unroll=False)
```

Values came out from LSTM are below :-

For toxicity of 5th sentence:-

```
Congratulations from me as well, use the tools well.  · talk "[0 0 0 0 0 0]Epoch 1/2  2624/159571 [.....] - ETA: 39:52 - loss: 0.2939 - acc: 0.9090

159571/159571 [==============================] - 2367s 15ms/step - loss: 0.0744 - acc: 0.9747

Epoch 2/2159552/159571 [============================>.] - ETA: 0s - loss: 0.0576 - acc: 0.9796

159571/159571 [==============================] - 2365s 15ms/step - loss: 0.0576 - acc: 0.9796


<keras.callbacks.History at 0x7f6e8594c550>
```

Epochs gave more accuracy towards the model prediction.

Refinement

Scikit-learn provides a pipeline utility to help automate machine learning workflows. Pipelines are very common in Machine Learning systems, since there are lot of data to manipulate and many data

transformations to apply. So we will utilize pipeline to train every classifier. Pipeline mode applied for logistic regression with stopwords ("english"). Below is the accuracy results:-

Training the model using pipeline to accurately fit the test data using stopwords

[nltk_data] Downloading package stopwords to /content/nltk_data... [nltk_data] Package stopwords is already up-to-date!

Processing toxic Test accuracy is 0.9549174879887579

Processing severe_toxic Test accuracy is 0.9910556600011394

Processing obscene Test accuracy is 0.9761864068820145

Processing threat Test accuracy is 0.9973793653506523

Processing insult Test accuracy is 0.9687043050570653

Processing identity_hate Test accuracy is 0.991758293928863

For LSTM model refinement done with the different epoch values and different solvers like SGD classifier as currently I am using Adam classifier which is best fit for given data. There is no much visible difference so this is the best testing accuracy we can say.

In future, we can think of comparison with CNN (Convolutional Neural Network) or some mixed form of RNN.

IV. Results

Model Evaluation and Validation

Final model supporting all qualities is Recurrent Neural Network based LSTM(Long Short Term Memory model). This model performed well with previous inputs and it automatically learns and predicts toxicity in precise manner. For basic baseline model we can consider Logistic Regression as a part for supervised learning model, since we need to check with already existing data as well.
For splitting the model I have used
from sklearn.model_selection import train_test_split
This separates out train and test data accordingly. Few Parameters it used are for model_selection:-
train_pipe, test_pipe = train_test_split(toxictrain, random_state=42, test_size=0.33,shuffle=True)

Parameters used for model are:-
Logistic Regression:-
logregress = LogisticRegression(C=12.0,solver='sag')
RNN with LSTM :-

```
model = Sequential()
print("Emedding layer - layer will output word vectors for each one of the words in senctence")
model.add(Embedding(vocabsize+1,
                    50, weights=[embeddingmatrix],
                    input_length=371,
                    trainable=False))

model.add(Bidirectional(LSTM(units=50,return_sequences=False,dropout=0.1, recurrent_dropout=0.1)))
model.add(Dense(50, activation="relu"))
```

```
model.add(Dropout(0.1))
model.add(Dense(6, activation='sigmoid'))

optimizer = optimizers.Adam(lr=0.001)
model.compile(loss='binary_crossentropy',
        optimizer=optimizer,
        metrics=['accuracy'])
```

Let's evaluate and validate supervised learning using Logistic Regression first for toxic comments only. From Table below for Logistic Regression let's analyse this:-

Catergory	Baseline Percent	Logistic Regression Percent	Increase in percent
Toxic	90.41555169799024	97.91127460503475	7.4957229070445095
Severe_toxic	99.00044494300343	99.41154721095938	0.41110226795595395
Obscene	94.7051782592075	98.99104473870565	4.285866479498154
Threat	99.70044682304429	99.87341058212333	0.17296375907903894
Insult	95.06363938309592	98.46400661774382	3.400367234647902
Identity hate	99.11951419744189	99.60268469834745	0.48317050090555824

Baseline Average : - 96.335 (approx.)
Logistic Regression fitted mode Average : - 99.038 (approx.)
As we can clearly see for average accuracy score percentage increase observed by 2.7%. That means our model trained and test well with given data.
So, I can clearly say there is significant change in accuracy while fitting of data. Data is properly fitted by Logistic Regression as a benchmark model.

Now let's visualize the change and results seen by Neural Network model. For Deep Learning model, we generally try to make our model behave in such a way that it can work properly for any random input that will come. It's like we are moving towards making our model intelligent.

As discussed, the model which I chose for Deep Learning model is Recurrent Neural Network based Long Short Term Memory Model. This model generally learns from previous inputs and behaves accordingly in future inputs. Let's see how it behaved when we fitted the data and trained it. Below are the accuracy results after training the data:-

```
Epoch 1/2  2624/159571 [.....] - ETA: 39:52 - loss: 0.2939 - acc: 0.9090
159571/159571 [==============================] - 2367s 15ms/step - loss: 0.0744 - acc: 0.9747
Epoch 2/2159552/159571 [==============================>.] - ETA: 0s - loss: 0.0576 - acc: 0.9796
159571/159571 [==============================] - 2365s 15ms/step - loss: 0.0576 - acc: 0.9796
```

As we can clearly visualize for epoch 2 from epoch1 there is a change in accuracy of approximately 7%. Finally, while training the data for epoch accuracy rate comes out to be 97.96%. This result is pretty much fine and giving accurate results in terms of predicting and classifying the toxicity level in comments. If we increase epoch results may get refine but time limit or memory issues might come in between.
So, in total what we can classify here is average accuracy scores for our models as below:-

Baseline	Logistic Regression	RNN based LSTM

96.335	99.038	97.96

Yes, final model is reasonable and aligned with solutions expectations. I will justify the behavior in next section. Final parameters of model are appropriate and tested across various inputs even unseen as well. Let's see below how toxicity levels behaved for some random inputs:-

Predicting the model
Toxicity levels for 'Why the fuck are you here?':
Toxic: 99%
Severe Toxic: 32%
Obscene: 97%
Threat: 4%
Insult: 81%
Identity Hate: 8%

Toxicity levels for 'i will fight against you':
Toxic: 31%
Severe Toxic: 0%
Obscene: 8%
Threat: 2%
Insult: 12%
Identity Hate: 1%

Toxicity levels for 'have a nice day':
Toxic: 5%
Severe Toxic: 0%
Obscene: 2%
Threat: 0%
Insult: 2%
Identity Hate: 0%

From the above results we can clearly say results are trustworthy and can classify more appropriately.
For Cross-Validation, I have used "from sklearn.model_selection import train_test_split" which gave me output as (106912,) (52659,) while printing shape of data after preprocessing.
Similarly, I tried cross-validating using kFold "*class* sklearn.model_selection.**KFold**(*n_splits=3, shuffle=False, random_state=None*)"
Below are the results from kFold validation:-

```
for train_index, test_index in kf.split(toxictrain):

  print("TRAIN:", train_index, "TEST:", test_index)

  X_train, X_test = X[train_index], X[test_index]
```

TRAIN: [79786 79787 79788 ... 159568 159569 159570] TEST: [0 1 2 ... 79783 79784 79785]
TRAIN: [0 1 2 ... 79783 79784 79785] TEST: [79786 79787 79788 ... 159568 159569 159570]

Justification

Final results are much more strong and significant than the benchmark result reported.
We can clearly see with the accuracy score table :-

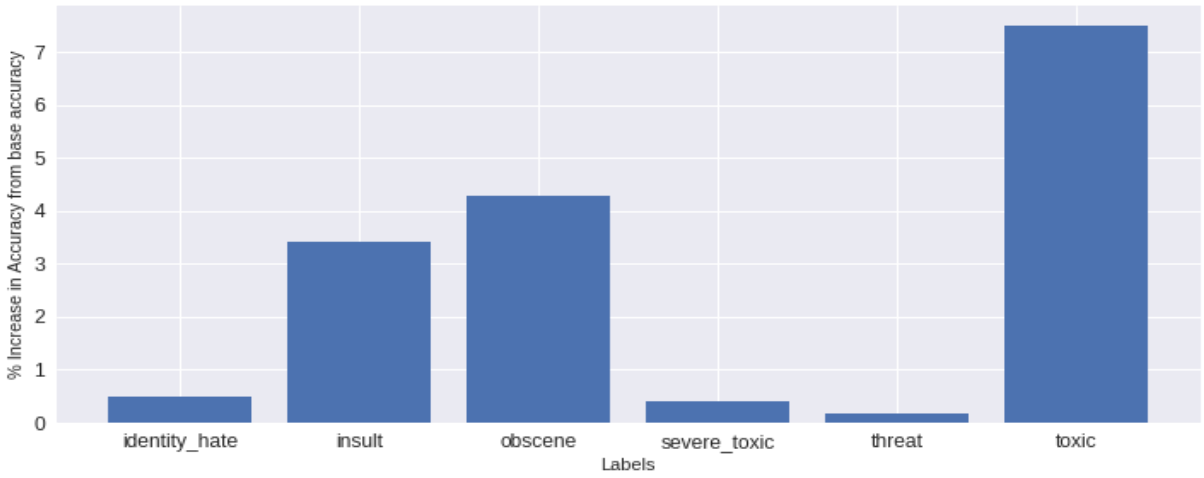
Baseline	Logistic Regression	RNN based LSTM
96.335	99.038	97.96

If we could have increase the epoch values then it accuracy will increase exponentially and surpass the logistic regress accuracy as well but I felt it would be fast and memory efficient analysis to provide deep learning model with epoch 2. Our main goal was to just train and classify the model in such a way that it can learn any random sentence quickly and classify it which can be used to remove from abusive comments later on.

V. Conclusion

Free-Form Visualization

All relevant and important quality about the problem has been discussed properly. Datasets were clearly understood and refined in proper manner. Classification of toxic comments were done properly to all stages from Data Preprocessing to Model Evaluation and Results. All results are quantifiable and justifiable. All plots discussed above are clearly state and defined with axes, title and datum. Most important part is visualization has been done with proper quality and reference.
A good example can be using Logistic Regression training we got this result in form of graphs everything is mentioned properly as this is for change in accuracy:-



Reflection

All parts of project are thoroughly summarized. In a nutshell I can say these are the steps done end to end to solve the problem:-
1) Data Pre-processing step – Uploading, reading and filtering out data.
2) Data Optimization – Optimized and embedded data with various classifiers.
3) Model Training – Trained data with Supervised and Unsupervised learning especially feature is applying a neural network.
4) Model Test – Test data and also trained it for Random input.

Interesting aspect of project is using LSTM (RNN based model), the concept is completely different and is in very much use nowadays for Natural Language Processing problems. Keras is one of the fast and optimized libraries and it was good to work on different classifiers and solvers over it. Memory optimization and time limit while running on colab was a major issue it could not be a challenge if we have good GPU or CPUs.
It can be used in general setting to solve these types of problems as our model predicts every random sentence properly and classifies it accurately.

Challenges

While working towards the project challenges which I faced are many especially working towards deep learning model. Let me elaborate on this, for data preprocessing part it is still not perfectly evident to detect all words clearly because words can be formed using special characters, images or a word that can hurt sentiment. It is just a prediction which I made sometimes positive or negative emoticons and images are also there which are ambiguous in nature. For deep learning model, especially when I was working towards Neural Networks it particular uses time and memory consuming libraries which I think will be difficult for normal processor to work upon(it might get hang for sometime). Although, Keras libraries are much more optimized but refining data and making prediction with model takes time. It's always mandatory to have all libraries installed and should have fast CPU to process the data in case it's huge.

Improvement

For Future improvement, I thought of clubbing few algorithms related to Neural Networks. We can enhance our current RNN based model to more generalized model. It can be improved in much better fashion if we could try with Convolutional Neural Network as well. Few points I thought about which might help in improving the project. These are: -
1) Data Preprocessing – We can preprocess data more if we have good amount of regular expression or list of words those are not toxic (non-hurting words).
2) Data Embedding or Tokenizing – It behaved fine but we can create much better and smart vocabulary especially classifying images and complicated slang words.
3) Model Results – Here I thought we can use CNN (Convolutional Neural Network) for more accuracy with word embedding. May be a more refine algorithm which would be a mix of both CNN and RNN.

References:-

https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714
http://deeplearning.net/tutorial/lstm.html
https://isaacchanghau.github.io/post/lstm-gru-formula/
https://github.com/nikitabu/Kaggle_Jigsaw/blob/master/BASELINE%20-%20Bag%20of%20Words%20-%20Logistic%20Regression%20Pipeline%20CV.ipynb
http://dsbyprateekg.blogspot.sg/2017/12/can-you-build-model-to-predict-toxic.html
http://colah.github.io/posts/2015-08-Understanding-LSTMs/
https://medium.com/@nehabhangale/toxic-comment-classification-models-comparison-and-selection-6c02add9d39f
https://www.kaggle.com/sbongo/for-beginners-tackling-toxic-using-keras
https://towardsdatascience.com/multi-label-text-classification-with-scikit-learn-30714b7819c5
https://keras.io/optimizers/
https://web.stanford.edu/class/cs224n/reports/2762092.pdf
https://github.com/mborysiak/Toxic-Comments/blob/master/2.%20Recurrent%20Neural%20Network%20Models.ipynb

https://nlp.stanford.edu/projects/glove/