TOXIC COMMENT CLASSIFICATION



A Project Report in partial fulfillment of the degree

# Bachelor of Technology

in

# Computer Science & Engineering

### By

19K41A05F8 K. Shanthan Reddy

19K41A05H7 L. Akhil Reddy

### Under the Guidance of

**D. RAMESH**

**Submitted to**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**S.R. ENGINEERING COLLEGE(A), ANANTHASAGAR, WARANGAL (Affiliated to JNTUH, Accredited by NBA) Dec-2021.**



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

**CERTIFICATE**

This is to certify that the Project Report entitled “ TOXIC COMMENTS CLASSIFICATION” is a record of bonafide work carried out by the student(s) K.SHANTHAN , L.AKHIL of Roll No(s) 19K41A05F8, 19K41A05H7 during the academic year 2021-22 in partial fulfillment of the award of the degree of ***Bachelor of Technology*** in **Computer Science & Engineering** by the Jawaharlal Nehru Technological University, Hyderabad

# Supervisor Head of the Department

**External Examiner ABSTRACT**

Over the past ten years, comment analysis have seen significant growth. Users mental plays a crucial in the activate participation in day to day social media activities. Negative comments plays a negative role in user’s mental health and performance. Nowadays users leave numerous comments on different social networks, news portals, and forums. Some of the comments are toxic or abusive. Due to numbers of comments, it is a lot of work to manually moderate all of the comments. So we have taken an natural language processing as core subject and took Lstm model to train the comments and universal sentence encoder to embed the sentences. The dataset we used have a large amount of toxic and non toxic comments , but by using this model we were able to predict if that comment is toxic or not and what is the toxicity of that comment and by using the lstm model for this project made us get a good accuracy. The dataset we took all was compatible with the model. So, we got any accuracy rate of 86%.

# Table of Contents

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Content** | **Page No** |
| 1 | Introduction | 1 |
| 2 | Literature Review | 3 |
| 3 | Design | 4 |
| 4 | Dataset | 6 |
| 5 | Pre-processing | 7 |
| 6 | Methodology | 10 |
| 7 | Results & conclusion | 16 |
| 8 | Future scope | 18 |
| 9 | References | 19 |

1. **INTRODUCTION**

In the early days of the Internet, people communicated only by e-mail that encountered spam. At the time, classifying emails as good or bad, whether or not spam was a challenge. Internet communications and data flows have changed significantly over time, especially since the development of social networks. With the advent of social media, classifying content as “good” or “bad” has become more important than ever to prevent social harm and prevent people from engaging in antisocial behavior.

Internet is an open discussing space for anyone to express their opinions. We can express anything whatever we need and there is no classification in the comments. However the use of internet has been more in the recent days and there are many number of platforms where we can express our feeling without any restrictions. Also the use of internet has been very useful in many ways which helps us in learning.

But where there is good there will always be bad, from all the good stuff we use we can get some bad things, such as usage of some inappropriate platforms where there are some false message and have negative impact on us. However harassment and abusive language can effect people’s freedom of sharing their opinions. This can disturb the environment of the internet. However most of the platforms have constant monitoring of their platform to minimize such inappropriate comments. But these are all temporary and need to be constantly monitored. Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments if its toxic.

Motivated by this problem, we want to build a platform where we can know the amount of toxicity present in a sentence. Where toxicity is defined as anything that is rude, disrespectful or any otherwise, the sentence that is likely to make someone leave a discussion.

Because toxic comments limit people's ability to express themselves and have different opinions, as a result of the negative, there are no positive conversations on social networks. As a result, detecting and restricting antisocial behavior in online discussion forums is a pressing requirement. These toxic comments might be offensive, menacing, or disgusting. Our goal is to identify these harmful remarks.

The dataset we took from Kaggle it says about the comments , toxic, severe toxic , obscene, threat, insult, identity hate in the columns and it is a very large data set containing more than 1500000+ rows. It is developed by jigsaw. The comments in the data are 90% non toxic and 10 toxic.(we got this while training the dataset).

Toxic comment classification we developed from using LSTM (long short term memory) model and embedded using universal sentence encoder(use). LSTM using one input layer and inputs (1,512) matrix and uses three LSTM layers for processing, we are using three layers for to get more accuracy and we are using a dense to the final output. Our main objective in this project is to give the toxicity of a given text. And to categorise the text into bad or good using the toxicity percentage of that text. So to get this used both lstm and use, because by combining both these get a good accurate success rate. We got a accuracy of 86% if we took 10000 rows in the dataset and got 97% accuracy if we took 35000 rows.

# Literature Survey

Bad words by text is a complex phenomenon, and different knowledge fields try to study and tackle this problem. This analysis of related work focuses on a computer science perspective of aggression identification, a recent emerging area. Currently, the scientific study of automatic identification of aggressive text, using information technology techniques, is increasing. In this study, several related literature are used to express different types of aggression. Some of those are hate (Tarasova et al.8 ), cyber bullying (Adamic9 ), abusive language (Nobata et al.3 ) to tackle this in past developers have developed toxic comment identifier using machine learning models like by using RNN, CNN but the problem they Have was not compatible with the model and the embedding was not so good. In the machine learning model they used seq2seq but did not good accuracy. Even they did develop in the natural language processing field they got but to a good extent thanks to the jigsaw , he developed the toxic comment classification project using long short term memory(LSTM). But the back draw was he didn’t used separate embedding models, he just included embedding layers into the lstm model.

# DESIGN:

## REQUIREMENT SPECIFICATION (S/W & H/W)

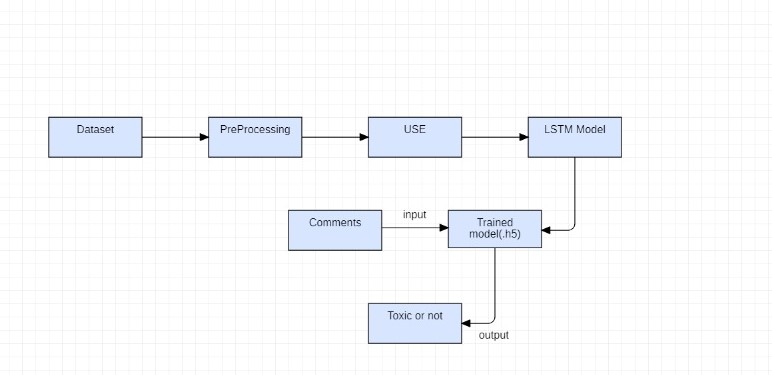
**Hardware Requirements**

* + - **System** : Pentium 4, Intel Core i3, i5, i7 and 2GHz Minimum
    - **RAM** : 4GB or above
    - **Hard Disk** : 10GB or above
    - **Input** : Keyboard and Mouse
    - **Output** : Monitor or PC

## Software Requirements

* + - **OS** : Windows 8 or Higher Versions
    - **Platform** : Google Collab
    - **Program Language** : Python

## Flow chart



**Figure 1 -** flow chart

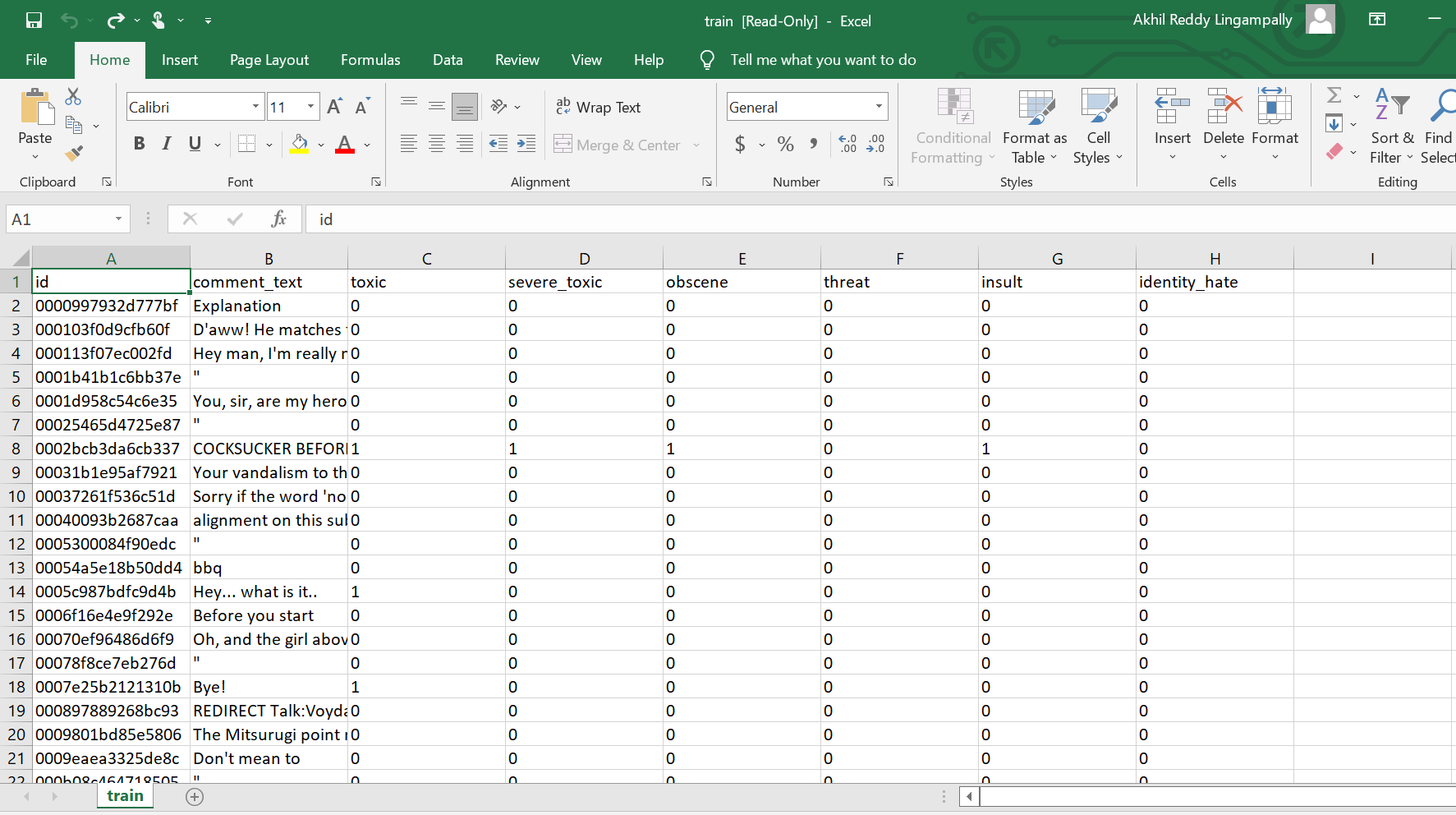
# DATASET:

We have collected our dataset from Kaggle. The data we collected has 8 columns and 150000+ rows . but for our model we need only 2 columns so out of 8 columns we consider only 2 columns. Also we do not need all the so we train the model only using the 2 columns. The model to be trained with over 150000+ rows has more execution time so we lower it to 10000 and we train the model. It is developed by jigsaw. The comments in the data are 90% non toxic and 10% toxic.(we got this while training the dataset).

Columns in dataset:

* id
* comment\_text
* toxic
* severe\_toxic
* obscene
* threat
* insult
* identify\_hate

From the above columns we will use only comment\_text, toxic.



**Figure 2 : dataset**

# DATA PREPROCESSING:

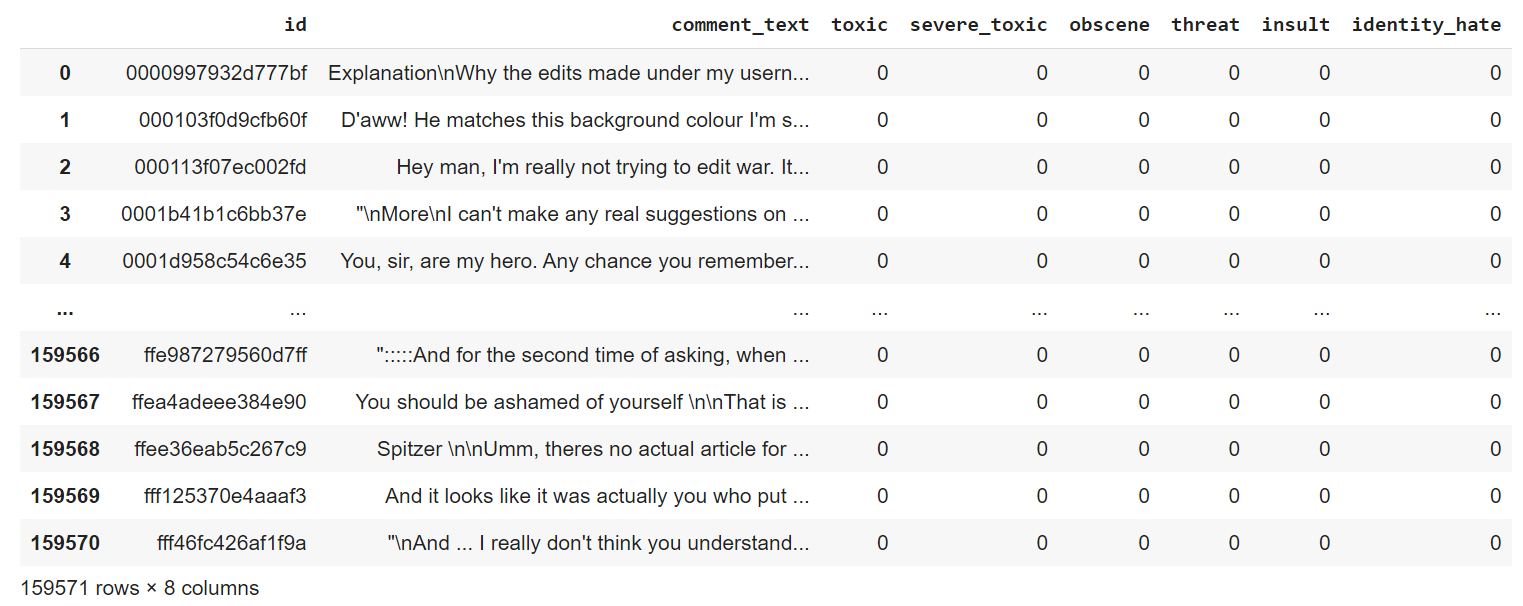
First we are going to install the tensor flow 2.6.0 version rather than new version, because we are getting some errors in our project for the latest version.th

%%capture

!pip install tensorflow\_text==2.6.0

Then we are going to import all the necessary libraries required for the project.

We will be doing this project in google colab. We are going to import the dataset from the google drive of [kundurushanthan777@gmail.com](mailto:kundurushanthan777@gmail.com).



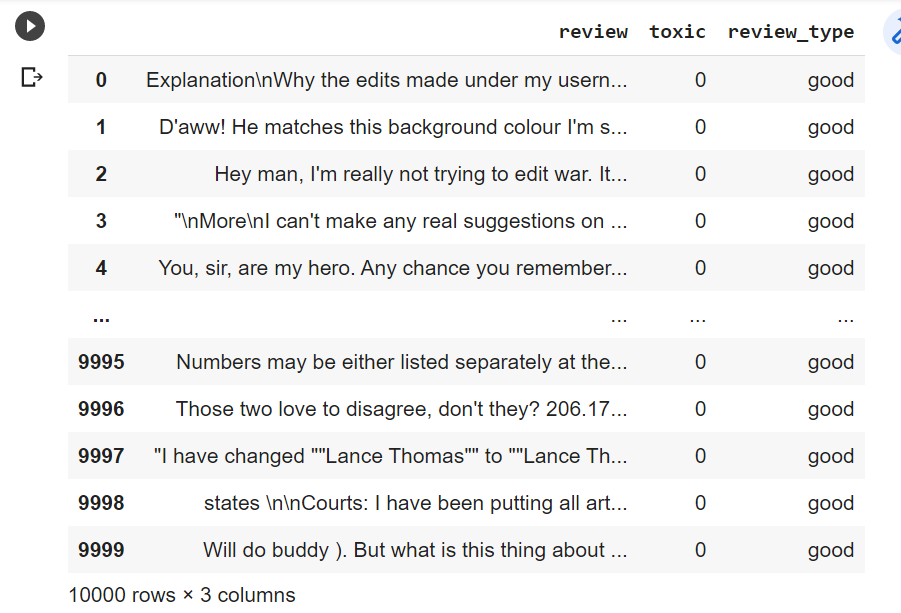
**Figure 3 : dataset**

Since the dataset is large it is going to take more amount of time train and test the dataset. Since the dataset contains 150000+ lines. So, in the next step we are going to reduce the size of the rows to 10000 only to easily test and train the data.



**Figure 4 : dataset**

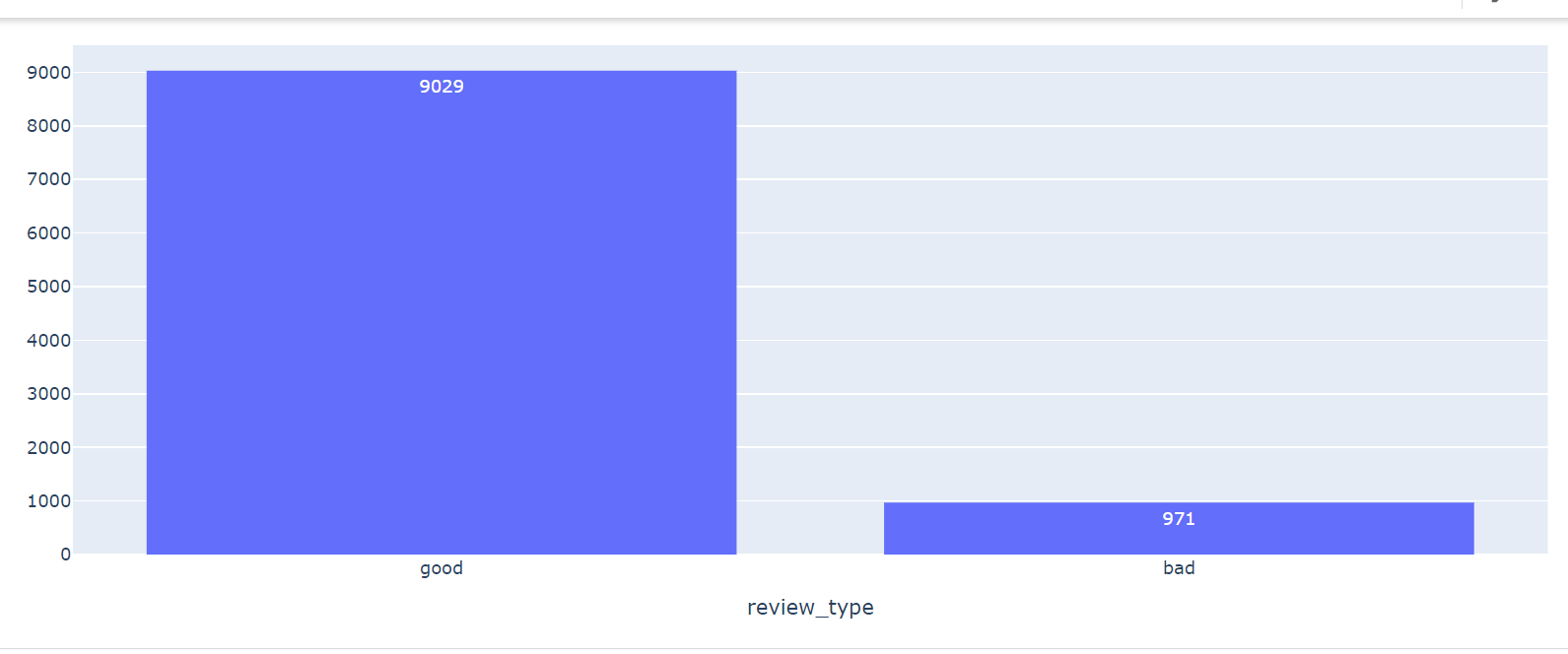
Now we have removed the 6 columns from the dataset because we need to train only 2 columns from it and added another column to justify good or bad comment .



**Figure 5 : dataset**

**GRAPH:**

In this graph we are going to say about the no of toxic and non toxic comments in the dataset.

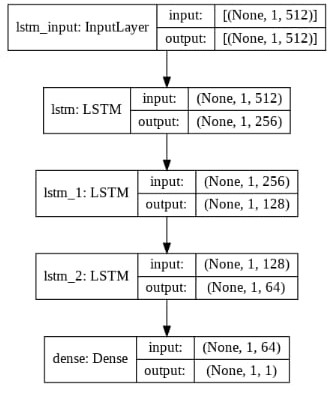


**Figure 6 : Graph**

Then we are going to train and test the dataset using the ratio of 0.75 : 0.25.

Using the universal sentence encoder with the ‘lstm’ model we are going to train the dataset.

By using ‘USE’ we are going to convert the text into vectors in (1,512) matrix format, then we are going to use three lstm layers to embed the given into vectors and convert them into smaller versions and at last we are going to use dense layer.



**Figure 7 : working**

# METHODOLOGY:

This section talks about the models used for the project. We used two models like LSTM(Long short term memory), universal sentence encoder.

### LSTM (Long Short Term Memory):

Long short-term memory (LSTM) is an artificial neural network used in the fields of artificial intelligence and deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. Such a recurrent neural network (RNN) can process not only single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition, machine translation, robot control, video games, and healthcare. LSTM has become the most cited neural network of the 20th century.

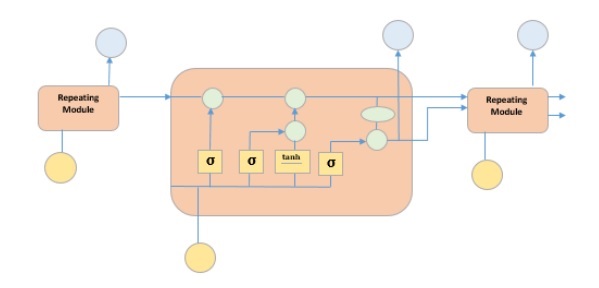
The name of LSTM refers to the analogy that a standard RNN has both "long-term memory" and "short-term memory". The connection weights and biases in the network change once per episode of training, analogous to how physiological changes in synaptic strengths store long-term memories; the activation patterns in the network change once per time-step, analogous to how the moment-to-moment change in electric firing patterns in the brain store short-term memories. The LSTM architecture aims to provide a short-term memory for RNN that can last thousands of timesteps, thus "long short-term memory".

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous application

At a high-level LSTM works very much like an RNN cell. Here is the internal functioning of the LSTM network. The LSTM consists of three parts, as shown in the image below and each part performs an individual function. The first part chooses whether the information coming from the previous timestamp is to be remembered or is irrelevant and can be forgotten. In the second part, the cell tries to learn new information from the input to this cell. At last, in the third part, the cell passes the updated information from the current timestamp to the next timestamp.

It is special kind of recurrent neural network that is capable of learning long term dependencies in data. This is achieved because the recurring module of the model has a combination of four layers interacting with each other.



The picture above depicts four neural network layers in yellow boxes, point wise operators in green circles, input in yellow circles and cell state in blue circles. An LSTM module has a cell state and three gates which provides them with the power to selectively learn, unlearn or retain information from each of the units. The cell state in LSTM helps the information to flow through the units without being altered by allowing only a few linear interactions. Each unit has an input, output and a forget gate which can add or remove the information to the cell state. The forget gate decides which information from the previous cell state should be forgotten for which it uses a sigmoid function. The input gate controls the information flow to the current cell state using a point-wise multiplication operation of ‘sigmoid’ and ‘tanh’ respectively. Finally, the output gate decides which information should be passed on to the next hidden state.

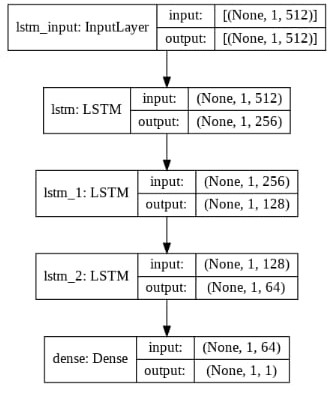
The LSTM is made up of four neural networks and numerous memory blocks known as cells in a chain structure. A conventional LSTM unit consists of a cell, an input gate, an output gate, and a forget gate. The flow of information into and out of the cell is controlled by three gates, and the cell remembers values over arbitrary time intervals. The LSTM algorithm is well adapted to categorize, analyze, and predict time series of uncertain duration.

The cells store information, whereas the gates manipulate memory. There are three entrances:

* **Input Gate:** It determines which of the input values should be used to change the memory. The **sigmoid** function determines whether to allow 0 or 1 values through. And the**tan(h)** function assigns weight to the data provided, determining their importance on a scale of -1 to 1.
* **Forget Gate:**It finds the details that should be removed from the block. It is decided by a**sigmoid**function. For each number in the cell state Ct-1, it looks at the preceding state (ht-1) and the content input (Xt) and produces a number between 0 (omit this) and 1 (keep this).
* **Output Gate:**The block’s input and memory are used to determine the output.The **sigmoid** function determines whether to allow 0 or 1 values through. And the **tan(h)** function determines which values are allowed to pass through 0, 1. And
* the **tan(h)** function assigns weight to the values provided, determining their relevance on a scale of -1 to 1 and multiplying it with the sigmoid output.

The recurrent neural network uses long short-term memory blocks to provide context for how the software accepts inputs and creates outputs. Because the program uses a structure based on short-term memory processes to build longer-term memory, the unit is dubbed a long short-term memory block. In natural language processing, these systems are extensively used.

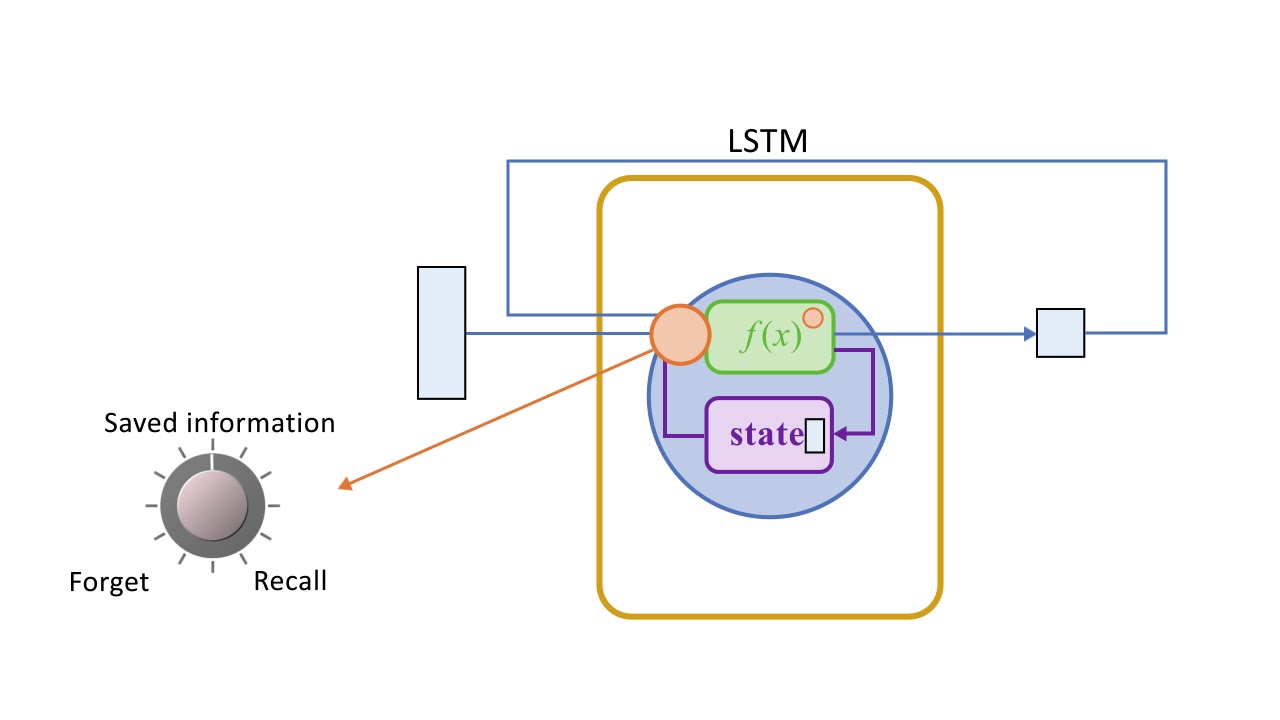
**Model Architecture :**



**LSTM life cycle:**

The LSTM cycle is divided into four steps:

* Using the forget gate, information to be forgotten is identified from a prior time step.
* Using input gate and tanh, new information is sought for updating cell state.
* The information from the two gates above is used to update the cell state.
* The output gate and the squashing operation provide useful information.

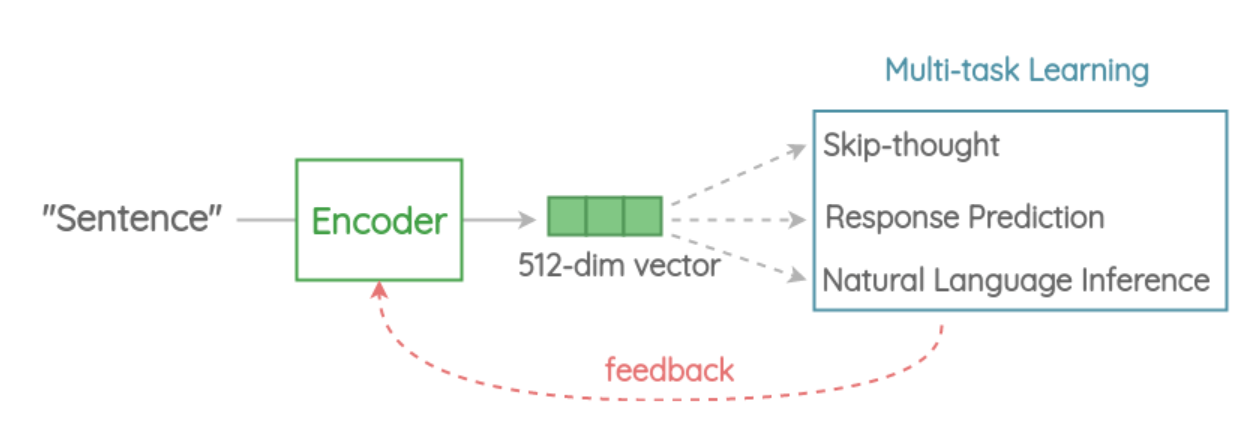


A dense layer receives the output of an LSTM cell. After the dense layer, the output stage is given the softmax activation function.

**Universal Sentence Encoder:**

The universal sentence encoder makes looking up embeddings at the sentence level as simple as it has previously been to look up embeddings at the word level. Then, using less supervised training data, the sentence embeddings can be easily employed to compute sentencelevel meaning similarity and improve performance on subsequent classification tasks. The universal sentence encoder model converts textual information into numerically represented, high-dimensional vectors called embeddings. It aims to transfer learning especially to other NLP tasks like text categorization, semantic similarity, and clustering. The freely accessible universal sentence encoder is listed in Tensor flow-hub. To learn for a wide range of jobs, it is trained on a number of data sources.

On a high level, the idea is to design an encoder that summarizes any given sentence to a 512-dimensional sentence embedding. We use this same embedding to solve multiple tasks and based on the mistakes it makes on those, we update the sentence embedding. Since the same embedding has to work on multiple generic tasks, it will capture only the most informative features and discard noise. The intuition is that this will result in an generic embedding that transfers universally to wide variety of NLP tasks such as relatedness, clustering, paraphrase detection and text classification.



1. **Tokenization** :

First, the sentences are converted to lowercase and tokenized into tokens using the Penn Tree bank (PTB) tokenizer.

1. Encoder This is the component that encodes a sentence into fixed-length 512-dimension embedding. In the paper, there are two architectures proposed based on trade-offs in accuracy vs inference speed.

Variant 1:

Transformer Encoder In this variant, we use the encoder part of the original transformer architecture. The architecture consists of 6 stacked transformer layers. Each layer has a self-attention module followed by a feed-forward network. The self-attention process takes word order and surrounding context into account when generating each word representation. The output context-aware word embeddings are added element-wise and divided by the square root of the length of the sentence to account for the sentence-length difference. We get a 512-dimensional vector as output sentence embedding. This encoder has better accuracy on downstream tasks but higher memory and compute resource usage due to complex architecture. Also, the compute time scales dramatically with the length of sentence as self-attention has O(n2) time complexity with the length of the sentence. But for short sentences, it is only moderately slower.

Variant 2:

Deep Averaging Network (DAN) In this simpler variant, the encoder is based on the architecture. First, the embeddings for word and bi-grams present in a sentence are averaged together. Then, they are passed through 4-layer feed-forward deep DNN to get 512-dimensional sentence embedding as output. The embeddings for word and bi-grams are learned during training. It has slightly reduced accuracy compared to the transformer variant, but the inference time is very efficient. Since we are only doing feedforward operations, the compute time is of linear complexity in terms of length of the input sequence.

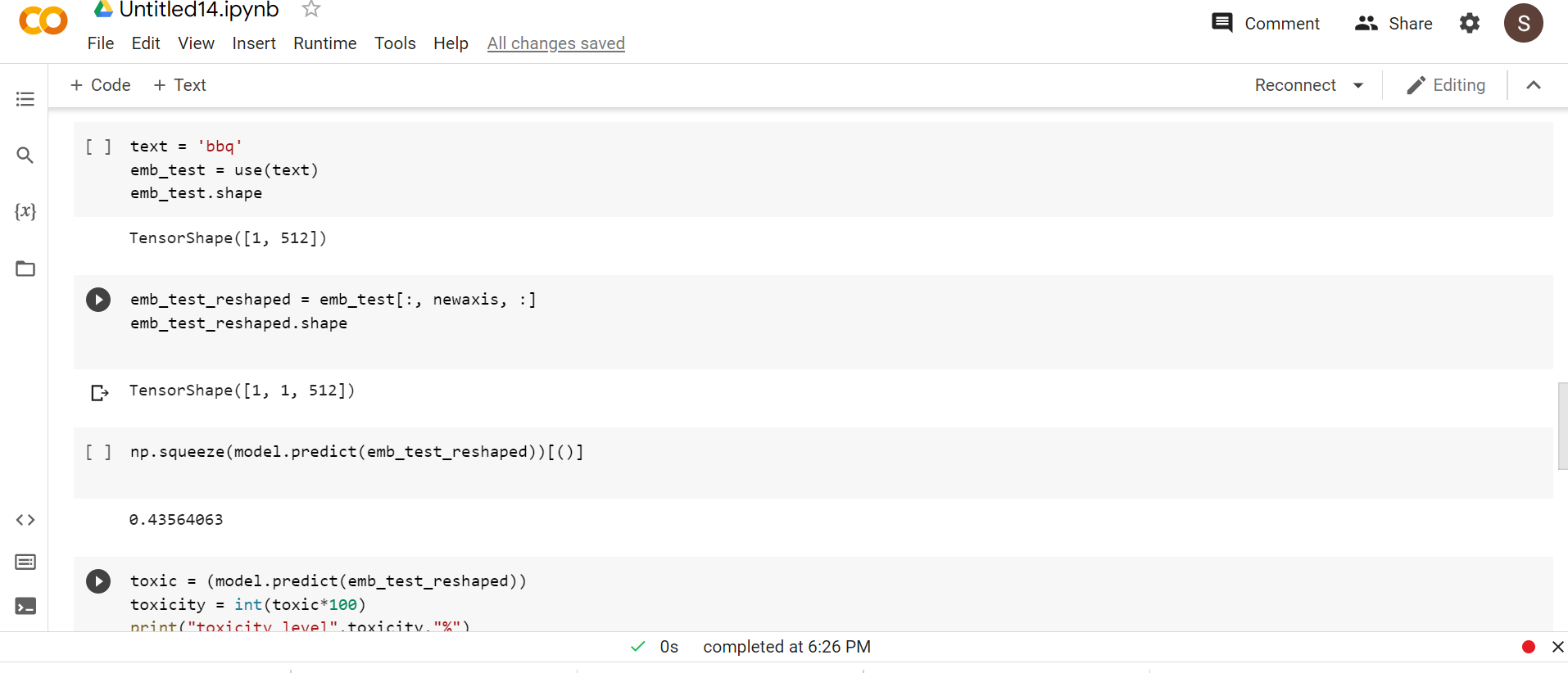
3. Multi-task Learning To learn the sentence embeddings, the encoder is shared and trained across a range of unsupervised tasks along with supervised training on the SNLI corpus.

# RESULTS AND CONCLUSION:

# 

# From the above project we have done we got an accuracy rate of 86% and we are able to the toxicity percentage of the given text. We are also showing the if the comment is good or bad. So by using lstm model only we are getting this of accuracy rate.

# 





# FUTURE SCOPE:

We are going to include two more columns in from the data set and going to train the model. And to develop and fully functional website for the comment classification. So that we can act more swiftly to take action on the comment user.

# REFERENCES:

# H.Almerekhi, H.kwak, j.salmin are these comment triggering? Predicting triggers of toxicity in online discussions,Apr.2020,pp.3033-3040.

1. M. Anand, R. Eswari, Classiﬁcation of Abusive Comments in Social Media using Deep Learning, 2019 3rd International Conference on Computing Methodologiesand Communication (ICCMC), Erode, India, Mar. 2019, pp. 974–977.
2. ] A. Bleiweiss, LSTM neural networks for transfer learning in online moderation of abuse context, ICAART 2019 - Proceedings of the 11th International Conference on Agents and Artiﬁcial Intelligence, Prague, Czech Republic,2019, pp. 112–122.
3. ⇒208
4. E. Brassard-Gourdeau, R. Khoury, Using Sentiment Information for Preemptive Detection of Toxic Comments in Online Conversations, ArXiv200610145 Cs, Jun.2020.
5. S. Carta, A. Corriga, R. Mulas, D. R. Recupero, R. Saia, A supervised multi-

class multi-label word embeddings approach for toxic comment classiﬁcation, IC3K 2019 - Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management ,vinena,2020.

1. A. G. D’Sa, I. Illina, D. Fohr, Towards non-toxic landscapes: Automatic toxic

comment detection using DNN, ArXiv191108395 Cs Stat, Nov. 2019, Accessed:

Jul. 03, 2020.

1. Toxic comment classification identify and classify toxic online by jigsaw. In 2017.
2. S. Deshmukh, R. Rade, Tackling Toxic Online Communication with Recurrent

Capsule Networks, 2018 Conference on Information and Communication Technology (CICT), Jabalpur, India, 2018.

1. S. Carta, A. Corriga, R. Mulas, D. R. Recupero, R. Saia, A supervised multi-

class multi-label word embeddings approach for toxic comment classiﬁcation, IC3K 2019 - Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management ,vinena,2020.