JADAVPUR UNIVERSITY

Hate Speech and Offensive Language Detection using Logistic Regression & Deep Learning Model

BY

SHROMONA SEN GUPTA

EXAMINATION ROLL NO. – MCA226011

UNDER THE SUPERVISION OF

PROF.(DR.) KAMAL SARKAR

PROJECT SUBMITTED IN PARTIAL FULFILLMENT FOR THE DEGREE OF MASTER OF COMPUTER APPLICATION

IN THE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING FACULTYOF ENGINEERING AND TECHNOLOGY

2022

Certificate of Recommendation

This is to certify that the thesis entitled “**Hate Speech and Offensive Language Detection using Logistic Regression and Deep Learning Model**“ has been carried out by **Shromona Sen Gupta (**University Roll Number - **001910503011,** Examination Roll No - **MCA226011,** University Registration Number **- 149874** of 2019-2020**)**, under the guidance and supervision of Prof. (Dr.) **Kamal Sarkar,** Department of Computer Science and Technology, Jadavpur University, Kolkata, is being presented for the partial fulfillment of the Degree of Master of Computer Applications during the academic year 2021-2022.The research results presented in the thesis have not been included in any other paper submitted for the award of any degree in any other university or institute.

Prof. (Dr.) **Kamal Sarkar** (Thesis Supervisor)
Department of Computer Science and Engineering
Jadavpur University, Kolkata-32

Countersigned

Prof. (Dr.) Anupam Sinha
Head of Department
Computer Science and Engineering
Jadavpur University, Kolkata-32

Prof. Chandan Mazumder
Dean
Faculty of Engineering and Technology
Jadavpur University, Kolkata-32

CERTIFICATE OF APPROVAL

This is to certify that the project entitled "**Hate speech and offensive language detection using Logistic Regression & Deep Learning Model**" is a bonafide record of work carried out by Shromona Sen Gupta in fulfillment of the requirements for the award of the degree of Master of Computer Application in the Department of Computer Science and Engineering, Jadavpur University. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed, or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

Signature of Examiner 1	Signature of Examiner 2
Date:	Date :

Date:

FACULTY OF ENGINEERING AND TECHNOLOGY JADAVPUR UNIVERSITY

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that this thesis entitled “**Hate Speech and Offensive Language Detection using Logistic Regression and Deep Learning Model**” contains original research work by the undersigned candidate, as part of his degree of Master of Computer Applications.

All information in this document has been obtained nad presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name (in Block Letters) - SHROMONA SEN GUPTA

Roll No. - 001910503011

Examination Roll No. - MCA226011

University Registration No. - 149874

ACKNOWLEDGMENT

I am delighted to convey my gratitude to my thesis supervisor Prof.(Dr.) Kamal Sarkar , Department of Computer Science & Engineering, my supervisor, for his overwhelming support and encouragement towards accomplishment throughout the duration of the project without which this work would not have been possible. His positive outlook and confidence inspired me and gave me confidence. I am grateful for his support, encouragement, and his valuable suggestions to complete this work. I feel deeply honored that I got this opportunity to work under him.

I would like to express my sincere thanks to all my teachers for providing a sound knowledge base and cooperation.

I would like to thank all the faculty members of the Department of Computer Science & Engineering of Jadavpur University for their continuous support. Last, but not the least, I would like to thank my batch mates for staying by my side when I need them.

Shromona Sen Gupta

Roll No.: 001910503011

Reg. No.: 149874 of 2019-20

Examination Roll No.: MCA226011

CONTENTS

ABSTRACT

What is hate speech?

The words which propagate hatred towards a person or a group of persons in any form on the basis of sex, religion, race etc. It may also lead to violence.

What is offensive speech?

The words which offend someone or their opinion on anything. It may consist of the use of vulgar language or emoticons or any form of verbal abuse.

Difference between hate and offensive speech

1. Hate speech often leads to violence whereas offensive speech may not lead to violence.
2. Offensive speech is more of a personal attack to make a person feel worse whereas hate speech concentrates more on spreading violence and hatred among communities and making use of it.

Today, in the world of technology, people in the name of free speech often try to demean others and their beliefs and opinions. To maintain peace in the world of social media so that toxic people cannot spread hate and offend, it is necessary to detect hate and offensive speech. This model, if implemented, can reduce violence among communities and create peace and harmony in our community.

INTRODUCTION

It is common today for people to express their opinions and emotions. To track down whether these tweets are hateful or offensive we have implemented some NLP models- NaiveBayes, BERT and Logistic Regression. We have also used deep- learning layers: Deep Layer and SoftMax layer. Now the question arises what are hate and offensive.

Hate and offensive speech consists of illegal words that cannot be used to arouse violence or hatred or any personal attack.

Nowadays, social media has become an important part in everyone's daily life, which is used for communication and dissemination of information from anywhere to anywhere on the globe. Although this can be used for this positive and uplifting work, because of no overseer or due to the high number of users, people use hate and offensive speech, which is quite harmful and they get away with it. Hate speech, as defined in Merriam-Webster Dictionary is as follows – 'speech that is intended to insult, offend, or intimidate a person because of some trait (as race, religion, sexual orientation, national origin, or disability).' Offensive speech can be defined as 'speech that is used to cause anger, upset, resentful or annoyed'. This is quite harmful as this leads to psychological changes, cyber bullying, flaming etc.

The main as well as the biggest obstacle in classification of such texts is context, i.e., the text which we need to classify either as having hate content or offensive content has context attached to it, which means that for different individuals having different backgrounds or different traits, certain texts or words have different meaning altogether or the context becomes different. So, there is no fixed way to classify the text as offensive or hateful. So, it becomes really hard to train any ML model that can classify any text as hateful or offensive.

To tackle such kinds of problems, there have been efforts made throughout the world like Task 12: OffensEval 2: Multilingual Offensive content identification in Social Media text or OSATC4 shared task on offensive content detection. In this paper, I have taken 2 datasets – one contains 2 classes (hateful-or-offensive and not-hateful-or-offensive) and another has 3 classes (hateful, offensive and neither).The models that are trained are ensemble models – one having BERT+LR while the other model is NB+BERT+LR using threshold value and without using threshold value. We have also used deep learning models- One with NB+BERT+Dense layer+Softmax layer with and without using threshold value and the other one BERT+Dense layer+ Softmax layer.

In this paper, the programming is done in the following method – First, the dataset is split in training and testing parts, where the model will be trained based on the training data and its accuracy will be tested based on the test data. After that, the data preprocessing is done. Here BERT was used primarily for text encoding. After the text has been encoded, then it is passed through NB+LR,simply through LR, NB+Dense layer+Softmax layer and Dense layer+Softmax layer to get the output. After fine-tuning the parameters, the dataset with 2 classes has accuracy, f1, precision and recall score as 0.7062, 0.6762, 0.7062 and 0.6888 while the dataset with 3 parameters has 0.8625, 0.8440, 0.8625 and 0.8446 respectively .

METHODOLOGY

We have used the following models to predict the class of the texts:

Model A: In this model, we use BERT to encode the text and use Logistic Regression to predict the class of the text.

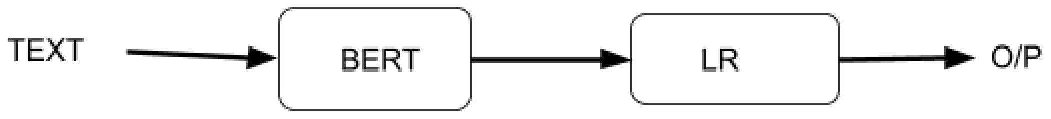


Fig 1a: BERT+LR

Model B: In this model we use the probabilities given by Naive Bayes and encode the text with BERT. Then we concatenate these two vectors and use this new vector to predict the class using Logistic Regression.

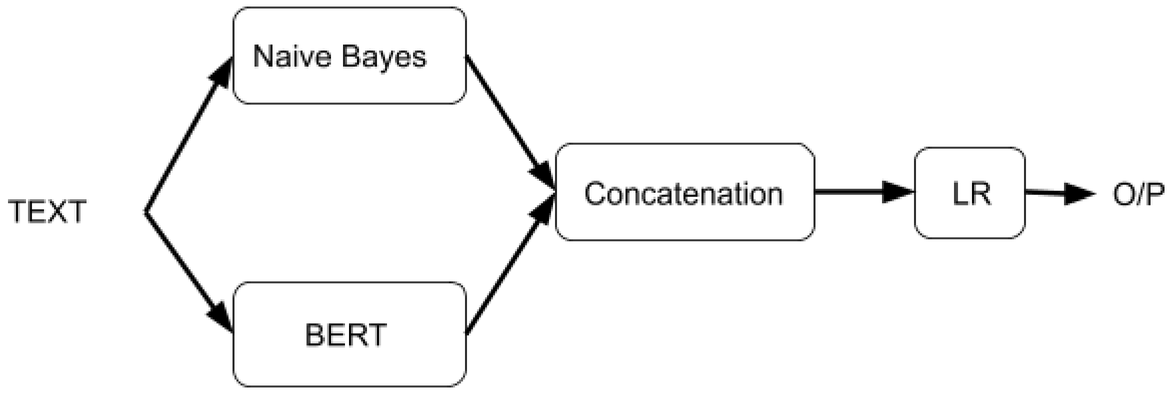


Fig 1b: NB(without threshold value)+BERT+LR

Model C: In this model we use the probability vector given by Naive Bayes and compare the highest probability or the class entropy with the threshold value. If the class entropy is greater than the threshold value, predict the class. Otherwise, encode the text using BERT and then concatenate the corresponding vector with the probability vector produced by Naive Bayes. Then, using Logistic Regression we train the model using training data and test the model using the testing data.

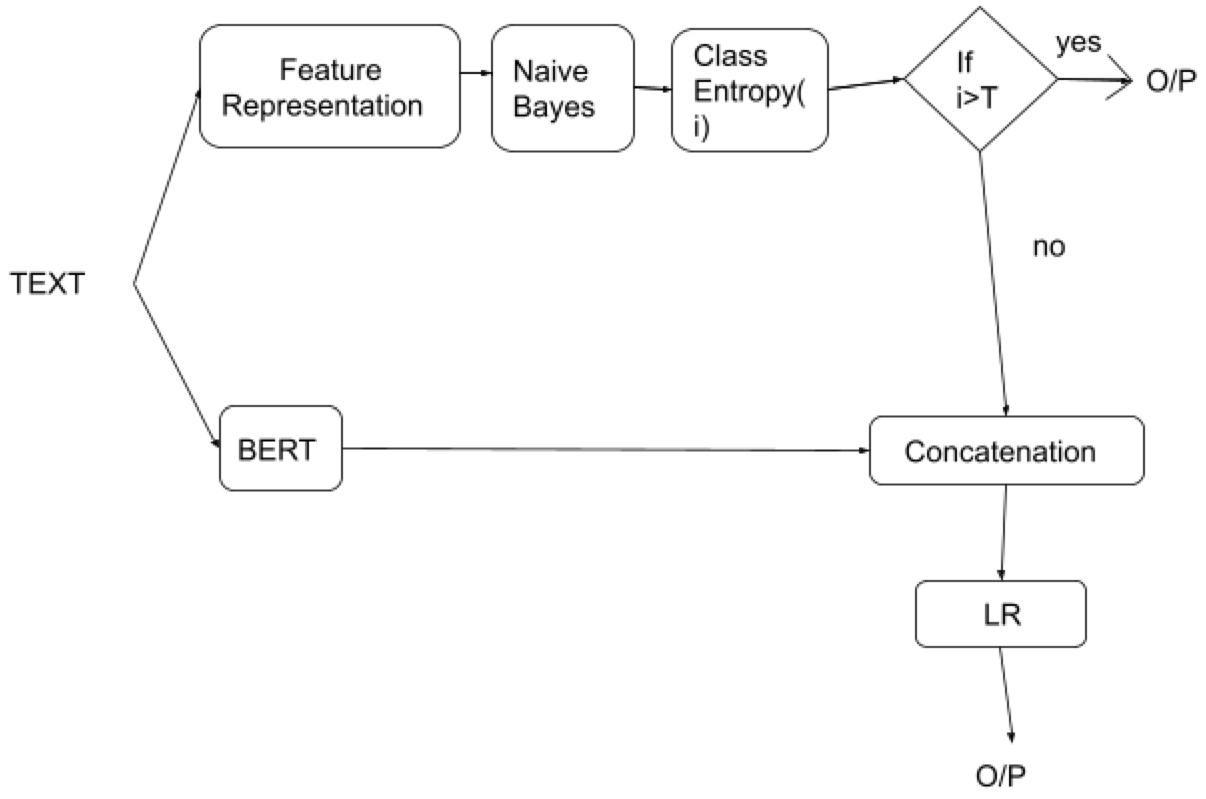


Fig1c: NB(with threshold value)+BERT+LR

Model D: In this deep model, we first encode the text and pass the feature vector to the dense+ softmax layer and predict the output class.

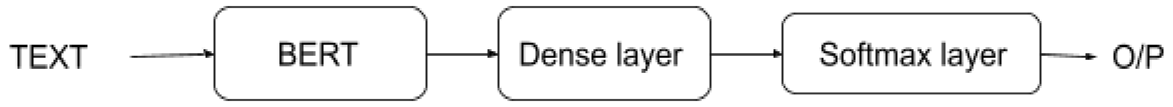


Fig 1d: BERT+Dense layer+Softmax layer

Model E: In this deep model, we concatenate the probability vector obtained from Naive Bayes and the feature vector from BERT and pass the resultant vector to the dense and softmax layer to predict the output.

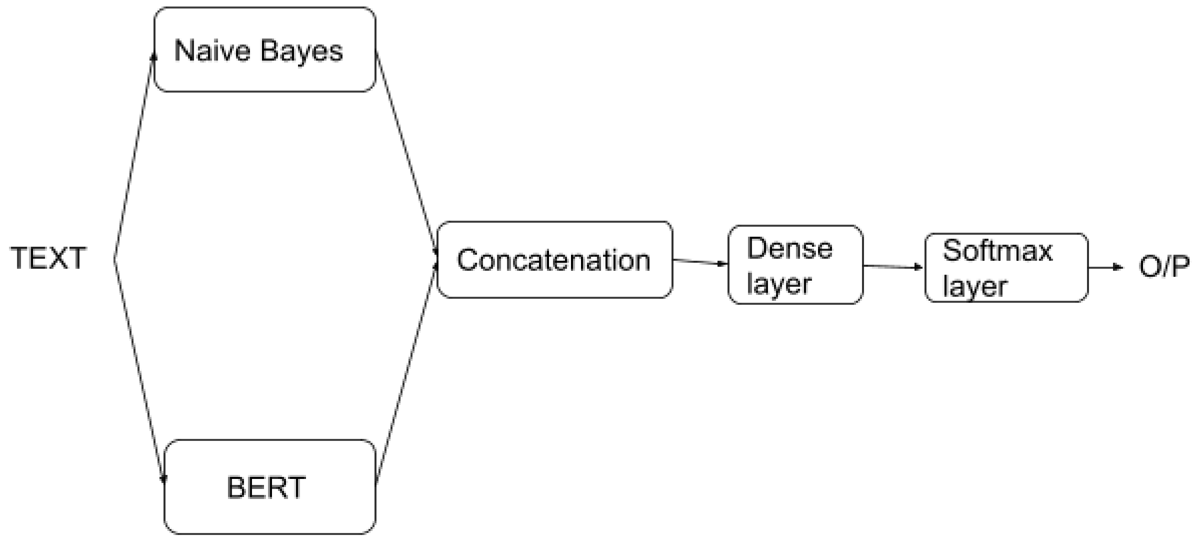


Fig 1e: NB(without threshold value on NB's output)+BERT+Dense layer+Softmax layer

Model F: In this deep model, we obtain the max-probability from the naive-bayes and compare it with the threshold value. If the class entropy is greater than the threshold value, we predict the class obtained by the naive-bayes. Else, with the help of BERT we obtain the feature vector and concatenate it with the probability vector. Then the resultant vector is passed to the dense and soft max layer to predict the class.

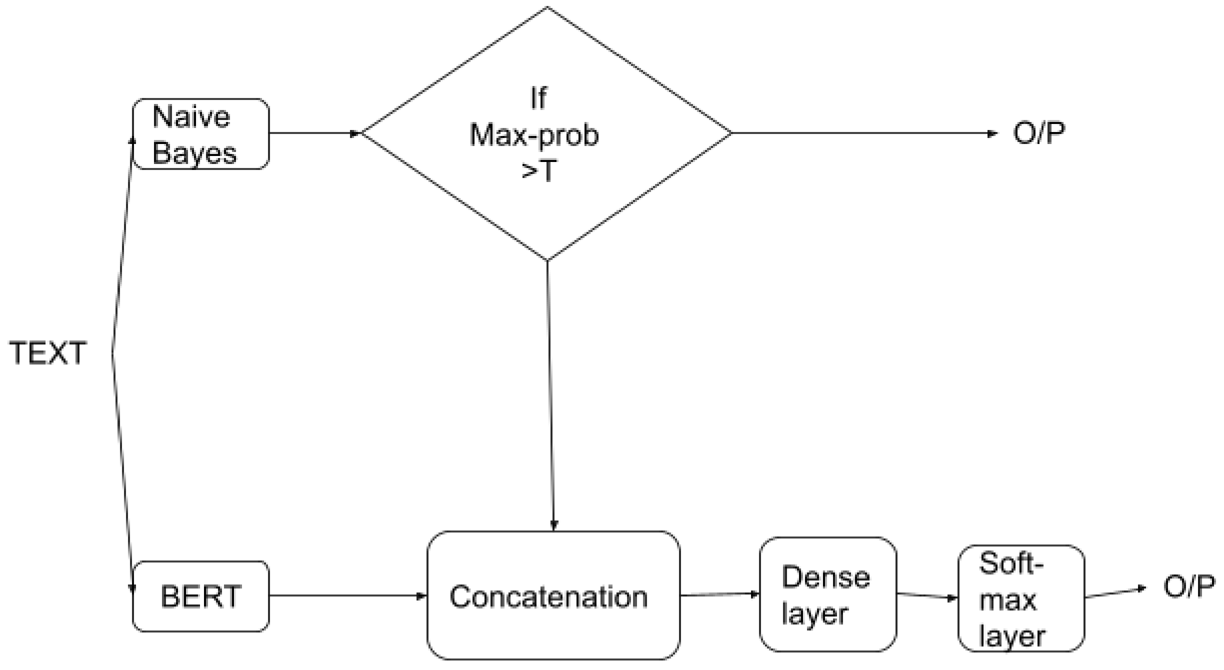


Fig 1f: NB(using threshold value on NB's output)+BERT+Dense layer+Softmax layer

NAÏVE BAYES CLASSIFIER

Suppose, Tweet is a set of text documents along with their target values. V is the set of all possible target values. This function learns the probability terms $P(w_k|v_j)$, describing the probability that a randomly drawn word from a document in class v_j will be English word w_k. It also learns the class prior probabilities $P(v_j)$.

1. Collect all words, punctuations, and other tokens that occur in Tweets.

```
Vocabulary <- c the set of all distinct words and other tokens occurring in any text document from Tweet
```

2. Calculate the required $P(v_j)$ and $P(w_k|v_j)$ probability terms

```
For each target value $v_j$ in V do
    $docs_j$ <- the subset of documents from Tweet for 1which the target value is $v_j$.
    $P(v_j)$ <- $|docs_j|/|Tweet|$
    $Text_j$ <- a single document created by concatenating all members of $docs_j$
    n <- total number of distinct word positions in $Text_j$
    for each word $w_k$ in Vocabulary
            $n_k$ <- number of times word $w_k$ occurs in $Text_j$
        $P(w_k|v_j)$ <- $(n_k+1)/(n+|Vocabulary|)$
```

Return the estimated target value for the document Doc. a_i denotes the word found in the i^{th} position within Doc.

```
    positions <- all word positions in Doc that contain tokens found in Vocabulary
    Return $V_{NB}$ where
```

$$V_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_{i \in positions} P(a_i|v_j)$$

Example:

Let us consider the dataset:

Doc	Tweet (Text)	Label
1	Youre a f*cking n*gger	1:0
2	Porch monkey life	1:0
3	RT its knives b*tch	2:1
4	B*tches know they b*tches b*tch	2:1
5	RT Rihanna really is trash	3:2
6	RT chris brown is trash	3:2

Unique words={youre, a, f*cking, n*gger, porch, monkey, life, RT, its, knives, b*tch, b*tches, know, they, Rihanna, really, is, trash, chris, brown}

Doc	youre	a	f*cking	n*gger	porch	monkey	lIfe	RT	Its	KnIVEs	b*tch	b*tches	know	they	rIhanna	reaIly	is	trash	chris	brown	CLAAS
1	1	1	1	1																	1:0
2					1	1	1														1:0
3								1	1	1	1										2:1
4											1	2	1	1							2:1
5								1							1	1	1	1			3:2
6								1									1	1	1	1	3:2

P(1:0)=2/6=⅓=0.3333

P(youre | 1:0)=(1+1)/(7+20)=2/27=0.0741

P(a | 1:0)=(1+1)/(7+20)=2/27=0.0741

P(f*cking | 1:0)=(1+1)/(7+20)=2/27=0.0741

P(porch | 1:0)=(1+1)/(7+20)=2/27=0.0741

P(life | 1:0)=(1+1)/(7+20)=2/27=0.0741

P(its | 1:0)=(0+1)/(7+20)=1/27=0.0370

P(b*tch | 1:0)=(0+1)/(7+20)=1/27=0.0370

P(know | 1:0)=(0+1)/(7+20)=1/27=0.0370

P(rihanna | 1:0)=(0+1)/(7+20)=1/27=0.0370

P(is | 1:0)=(0+1)/(7+20)=1/27=0.0370

P(chris | 1:0)=(0+1)/(7+20)=1/27=0.0370

P(n*gger | 1:0)=(1+1)/(7+20)=2/27=0.0741

P(monkey | 1:0)=(1+1)/(7+20)=2/27=0.0741

P(RT | 1:0)=(0+1)/(7+20)=1/27=0.0370

P(knives | 1: 0)=(0+1)/(7+20)=1/27=0.0370

P(b*tches | 1:0)=(0+1)/(7+20)=1/27=0.0370

P(they | 1:0)=(0+1)/(7+20)=1/27=0.0370

P(really | 1:0)=(0+1)/(7+20)=1/27=0.0370

P(trash | 1:0)=(0+1)/(7+20)=1/27=0.0370

P(brown | 1:0)=(0+1)/(7+20)=1/27=0.0370

P(2:1)=2/6=⅓=0.3333

P(youre | 2:1)=(0+1)/(9+20)=1/29=0.0345

P(f*cking | 2:1)=(0+1)/(9+20)=1/29=0.0345

P(porch | 2:1)=(0+1)/(9+20)=1/29=0.0345

P(life | 2:1)=(0+1)/(9+20)=1/29=0.0345

P(its | 2:1)=(1+1)/(9+20)=2/29=0.0690

P(b*tch | 2:1)=(2+1)/(9+20)=3/29=0.1034

P(know | 2:1)=(1+1)/(9+20)=2/29=0.0690

P(rihanna | 2:1)=(0+1)/(9+20)=1/29=0.0345

P(is | 2:1)=(0+1)/(9+20)=1/29=0.0345

P(chris | 2:1)=(0+1)/(9+20)=1/29=0.0345

P(a | 2:1)=(0+1)/(9+20)=1/29=0.0345

P(n*gger | 2:1)=(0+1)/(9+20)=1/29=0.0345

P(monkey | 2:1)=(0+1)/(9+20)=1/29=0.0345

P(RT | 2:1)=(1+1)/(9+20)=2/29=0.0690

P(knives | 2:1)=(1+1)/(9+20)=2/29=0.0690

P(b*tches | 2:1)=(2+1)/(9+20)=3/29=0.1034

P(they | 2:1)=(1+1)/(9+20)=2/29=0.0690

P(really | 2:1)=(0+1)/(9+20)=1/29=0.0345

P(trash | 2:1)=(0+1)/(9+20)=1/29=0.0345

P(brown | 2:1)=(0+1)/(9+20)=1/29=0.0345

P(3:2)=2/6=⅓=0.3333

P(youre | 3:2)=(0+1)/(10+20)=1/30=0.0333

P(a | 3:2)=(0+1)/(10+20)=1/30=0.0333

P(f*cking | 3:2)=(0+1)/(10+20)=1/30=0.0333	P(n*gger | 3:2)=(0+1)/(10+20)=1/30=0.0333
P(porch | 3:2)=(0+1)/(10+20)=1/30=0.0333	P(monkey| 3:2)=(0+1)/(10+20)=1/30=0.0333
P(life | 3:2)=(0+1)/(10+20)=1/30=0.0333	P(RT | 3:2)=(2+1)/(10+20)=3/30=0.1000
P(its | 3:2)=(0+1)/(10+20)=1/30=0.0333	P(knives | 3:2)=(0+1)/(10+20)=1/30=0.0333
P(b*tch | 3:2)=(0+1)/(10+20)=1/30=0.0333	P(b*tches | 3:2)=(0+1)/(10+20)=1/30=0.0333
P(know | 3:2)=(0+1)/(10+20)=1/30=0.0333	P(they | 3:2)=(0+1)/(10+20)=1/30=0.0333
P(rihanna | 3:2)=(1+1)/(10+20)=2/30=0.0667	P(really | 3:2)=(1+1)/(10+20)=2/30=0.0667
P(is | 3:2)=(2+1)/(10+20)=3/30=0.1000	P(trash | 3:2)=(2+1)/(10+20)=3/30=0.1000
P(chris | 3:2)=(1+1)/(10+20)=2/30=0.0667	P(brown | 3:2)=(1+1)/(10+20)=2/30=0.0667

Let's classify the new document:

RT Horrible RT Invader Zim was trash

If V_j =1:0, then

P(1:0) P(RT | 1:0) P(Horrible | 1:0) P(Invader | 1:0) P(Zim | 1:0) P(was | 1:0) P(trash | 1:0)

=0.3333 * 0.0370 * 0.0370 * 0.0370 * 0.0370 * 0.0370 * 0.0370

=8.5516e-12

If V_j=2:1, then

P(2:1) P(RT | 2:1) P(Horrible | 2:1) P(Invader | 2:1) P(Zim | 2:1) P(was | 2:1) P(trash | 2:1)

=0.3333 * 0.0690 * 0.0345 * 0.0345 * 0.0345 * 0.0345 * 0.0345

=1.1240e-9

If V_j=3:2, then

P(3:2) P(RT | 3;2) P(Horrible | 3:2) P(Invader | 3:2) P(Zim | 3:2) P(was | 3:2) P(trash | 3:2)

=0.3333 * 0.1000 * 0.0333 * 0.3333 * 0.3333 * 0.3333 * 0.1000

=4.1132e-5

So, the new document belongs to the 3:2 class.

BERT

Here’s how the research team of GoogleAI behind BERT describes the NLP framework:

“BERT stands for Bidirectional Encoder Representations from TRansformers. It is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks.”

First, BERT is based on the Transformer Architecture.

Second, BERT is pre-trained on a large corpus of unlabelled text including the entire Wikipedia and Book Corpus.

Third, BERT is a “deeply bidirectional” model. It means that BERT learns information from both the left and the right side of a token’s context during the training phase.

Example:

—----context—------

We went to the river <u>bank</u>.

I need to go to the bank to make a deposit.

—----context—----

If we try to predict the nature of the word “bank” by only taking either the left or the right context, then we will be making an error in at least one of the two given examples. One way to deal with this is to consider both the left and the right context before making a prediction and that is what BERT does.

Also, we can fine-tune it by adding just a couple of additional output layers to create state-of-art models for a variety of NLP tasks.

Hence, BERT is a two step process-

Train a language model on a large unlabelled text corpus.

Fine-tune this large model to specific NLP tasks to utilize the large repository of knowledge this model has gained.

Architecture: The BERT architecture builds on top of the Transformer. We currently have two variants available:

BERT Base: 12 layers (transformer blocks), 12 attention heads and 110 million parameters.

BERT Large: 24 layers (transformer blocks), 16 attention heads and 340 million parameters.

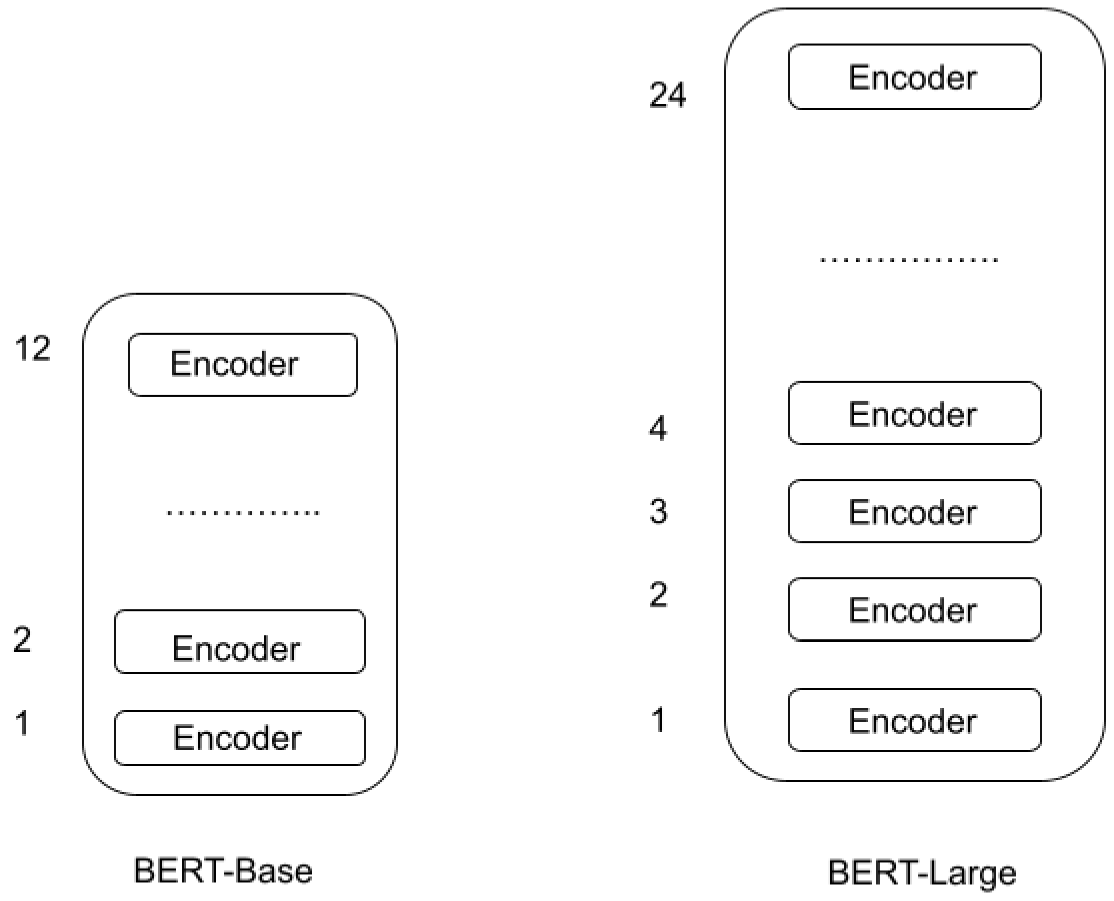


Fig2a: Architecture of BERT

Text Processing: The developers behind BERT have added a specific set of rules to represent the input text for the model. Many of these are creative design choices that make the model even better.

For starters, every input embedding is a combination of 3 embeddings:

1. Position Embeddings: BERT learns and uses positional embeddings to express the position of words in a sentence.
2. Segment Embeddings: BERT can also take sentence pairs as inputs for tasks. That is why it learns a unique embedding for the first and the second sentences to help the model distinguish between them.
3. Token Embeddings: These are the embeddings learned for the specific toen from the WordPiece token vocabulary.

For a given token, its input representation is constructed by summing the corresponding token, segment and position embeddings.

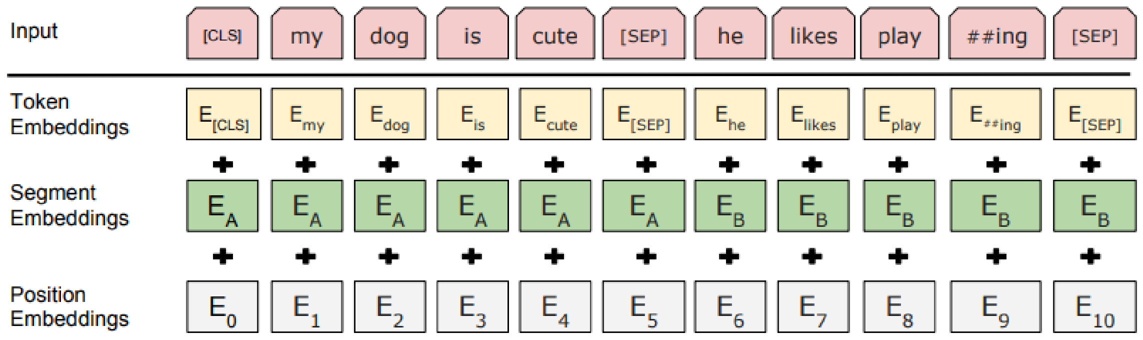


Fig 2b: Embeddings in BERT

Pre-training tasks: BERT is pre-trained on 2 NLP tasks:

1. **Masked Language Modeling (Bi-directionality)**: BERT is a deeply bidirectional model. The network effectively captures information from both the right and left context of a token from the first layer itself and all the way through to the last layer.

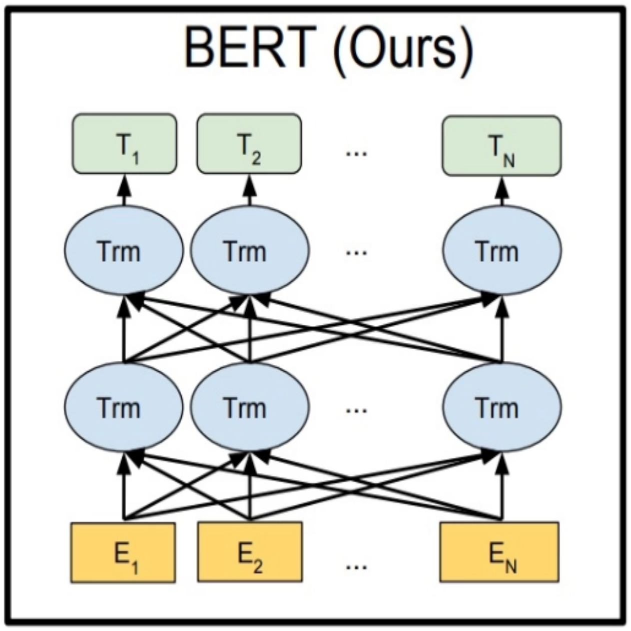


Fig2b: Bidirectionality in BERT

Let's take an example: “I love to read data science blogs on Analytics Vidhya”.

We want to train a bidirectional language model. Instead of trying to predict the next word in the sequence, we can build a model to predict a missing word from within the sequence itself.

Let's replace “Analytics” with “[MASK]”. This is a token to denote that the token is missing. We will train the model in such a way that it should be able to predict “Analytics” as the missing token: “I love to read data science blogs on [MASK] Vidhya.”

1. To prevent the model from focusing too much on a particular position or tokens that are masked, the researchers randomly masked 15% of the words.
2. The masked words were not always replaced by the masked tokens [MASK] because the [MASK] token would never appear during fine-tuning.
3. So the researchers used the below techniques:
    a. 80% of the time the words were replaced with the masked token [MASK].
    b. 10% of the time the words were replaced with random words.
    c. 10% of the time the words were left unchanged.

2. **Next Sentence Prediction**: Let us take an example.

Given two sentences A and B, where B is the actual next sentence that comes after A in the corpus, or just a random sentence?

Consider that we have a text dataset of 100,000 sentences. So, there will be 50,000 training examples or pairs of sentences as the training data.

1. For 50% of the pairs, the second sentence would actually be the next sentence.
2. For the remaining 50% of the pairs, the second sentence would be a random sentence from the corpus.
3. The labels for the first case would be ‘IsNext’ and ‘NotNext’ for the second case.

LOGISTIC REGRESSION

Logistic Regression is one of the most popular ML algorithms which comes under the Supervised Learning Technique. It is used for predicting the categorical variable using a given set of independent variables. Hence, the output is a discrete value, either Yes or No, 0 or 1, True or False etc. But instead of giving the exact value it gives the probabilistic values which lie between 0 and 1.
We fit an “S” shaped logistic function which predicts two maximum values (0 or 1). The curve from the logistic function indicates the likelihood of something.

Logistic function (Sigmoid function): This is a mathematical function used to map the predicted values to probabilities. It maps any real value into another value within a range of 0 and 1. The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the “S” form. The S-form curve is called the Sigmoid function.

In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold value tends to 0.

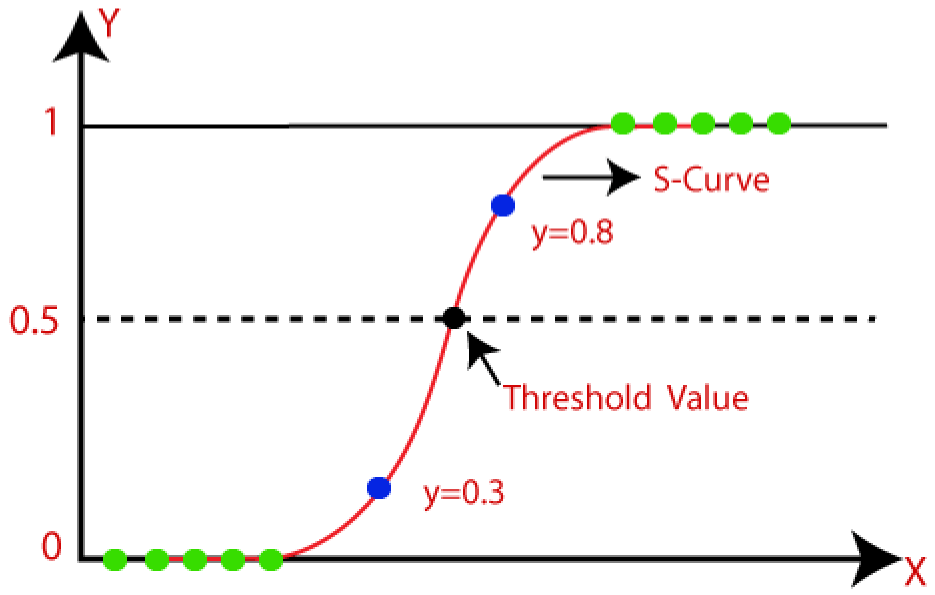


Fig3: Logistic function curve

Assumptions for Logistic Regression:

1. The dependent variable must be categorical in nature.
2. The independent variable should not have multicollinearity.

Logistic regression equation:

We know the eq of the straight line can be written as:

$y=b_0+b_1x_1+b_2x_2+b_3x_3+.....+b_nx_n$

In Logistic Regression y can be between 0 and 1 only. So divide the above eq by (1-y):

$y/(1-y)$; 0 for y=0, and infinity for y=1

But we need range between -[infinity] to +[infinity], then take logarithm of the equation:

$\log(y/(1-y))=b_0+b_1x_1+b_2x_2+b_3x_3+.....+b_nx_n$

The above eq is the final eq for Logistic Regression.

Types of Logistic Regression:

1. Binomial: In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
2. Multinomial: In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as “cat”, “dogs” or “sheep”.
3. Ordinal: In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as “low”, “Medium” or “High”.

DENSE LAYER

In any neural network, a dense layer is a layer that is deeply connected with its preceding layer which means the neurons of the layer are connected to every neuron of its preceding layer. This layer is the most commonly used layer in artificial neural network networks.

The dense layer's neuron in a model receives output from every neuron of its preceding layer, where neurons of the dense layer perform matrix vector multiplication.

$$A\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}.$$

Fig4: Matrix Multiplication

Values under the matrix are the trained parameters of the preceding layers and also can be updated by the backpropagation. Backpropagation is the most commonly used algorithm for training the feedforward neural networks. Generally, backpropagation in a neural network computes the gradient of the loss function with respect to the weights of the network for single input or output. From the above intuition, we can say that the output coming from the dense layer will be an N-dimensional vector, i.e., it is reducing the dimension of the vectors. Hence, a dense layer is used for changing the dimension of the vectors by using every neuron.

Dense Layer from Keras:
Keras provide dense layers through the following syntax-

```
        tf.keras.layers.Dense(unit, activation=None, use_bias=True,
kernel_initializer=”glorot_uniform”, bias_initializer=”zeroes”, kernel_regularizer=None,
bias_regularizer=None, activity_regularizer=None, kernel_constraint=None,
bias_constraint=None, **kwargs)
```

Keras Dense Layer Hyperparameters:

1. **Units**: Units are one of the most basic and necessary parameters of the Keras dense layer which defines the size of the output from the dense layer. It must be a positive integer since it represents the dimensionality of the output vector.

2. **Activation**: In neural networks, the activation function is a function that is used for the transformation of the input values of neurons. Basically, it introduces the nonlinearity into the networks of neural networks so that the networks can learn the relationship between the input and output values.

   If in this Keras layer no activation is defined it will consider the linear activation function. The following options are available as activation function Keras.

   a. Relu function (activation=activations.relu)- rectified linear unit activation function.
   b. Sigmoid function (activation=activations.sigmoid)- Sigmoid activation function, sigmoid(x)=1/(1+exp(-x))
   c. Softmax function (activation=activations.softmax)- softmax converts a vector of value to a probability distribution.
   d. Softplus function (activation=activations.softplus)- Softplus activation function, softplus(x)=log(exp(x)+1)
   e. Softsign function (activation=activations.softplus)- Softsign activation function, softsign(x)=x/(abs(x)+1).
   f. Tanh function (activation=activation.tanh)- Hyperbolic tangent activation function
   g. Selu function (activation=activations.selu)- Scaled Exponential Linear Unit (SELU)
   h. Elu function (activation=activations.elu)- Exponential Linear Unit
   i. Exponential function (activation=activations.exponential)- Exponential Activation function

3. **Use_bias**: Use_Bias parameter is used for deciding whether we want a dense layer to use a bias vector or not. It is a boolean parameter if not defined then use_bias is set to true.

4. **Kernel_initiaizer**: This parameter is used for initializing the kernel weights matrix. The weight matrix is a matrix of weights that are multiplied with the input to extract relevant feature kernels.

5. **Bias_initializer**: This parameter is used for initializing the bias vector. A bias vector can be defined as the additional sets of weight that require no input and correspond to the output layer. By default, it is set as zeroes.

6. **Kernel_regularizer**: This parameter is used for regularization of the kernel weight matrix if we have initialized any matrix in the kernel_initializer.

7. **Bias_regularizer**: This parameter is used for regularization of the bias vector if we have initialized any vector in the bias_initializer. By default, it is set to none.

8. **Activity_regularizer**: This parameter is used for the regularization of the activation function which we have defined in the activation parameter. It is applied to the output of the layer. By default, it is set to none.

9. **Kernel_constraint**: This parameter is used to apply the constraint function to the kernel weight matrix. By default, it is set as none.

10. **Bias_constraint**: This parameter is used to apply the constraint function to the bias vector. By default, it is set to none.

Basic Operations with Dense Layer:

Using the attributes a dense layer operation can be represented as:

Output=activation(dot(input, kernel)+bias)

Where if the input matrix for the dense layer has arank of more than 2, then dot product between the kernel and input along the last axis of the input and zeroth axis of the kernel using tf.tensordot calculated by the dense layer if the use-bias is False.

SOFTMAX LAYER

To understand the softmax layer, let us take an example.

Suppose, we have a dataset with five features from FeatureX1 to FeatureX5 and the target variable has three classes.

Now we create a simple neural network.

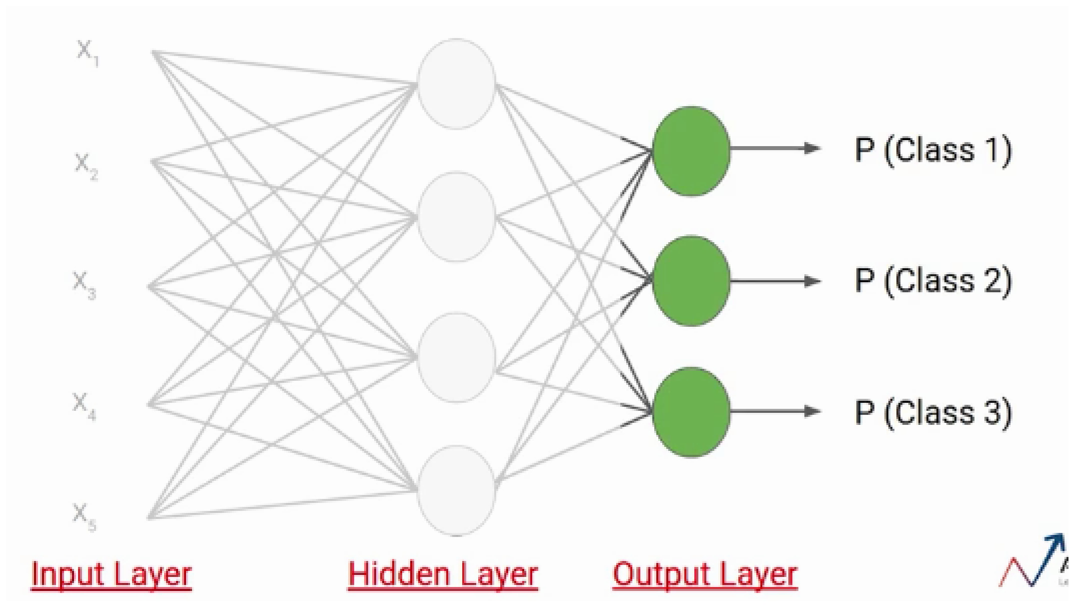


Fig5a: Neural network for five features and three classes

Here we have an input layer with 5 neurons as we have 5 features and we have one hidden layer which has 4 neurons. Each of these neurons uses inputs, weights and biases here to calculate a value which is represented as Zij here.

For example, the first neuron of the first layer is represented as Z11. Similarly, the second neuron of the first layer is represented as Z12 and so on.

Over these values, we apply the activation function. Let say a tanh activation function and send the values or result to the output layer.

The no. of neurons in the output layer depends on the no. of classes in the dataset. Since we have three classes in the dataset we will have three neurons in the output layer. Each of these neurons will give you the probability that the data points belong to class 1. Similarly, the second will give the probability that the data point belongs to class 2 and so on.

The Softmax activation function calculates the relative probabilities, i.e., it uses the value of Z21, Z22, Z23 to determine the final probability value.

The SoftMax function returns the probability of each class. The eq for the SoftMax activation function:

$$softmax(z_i) = \frac{exp(z_i)}{\sum_j exp(z_j)}$$

Here, the Z represents the values from the neurons of the output layer. The exponential acts as the nonlinear function. Then, these values are divided by the sum of exponential values in order to normalize and then convert them into probabilities.

Let us take an example.

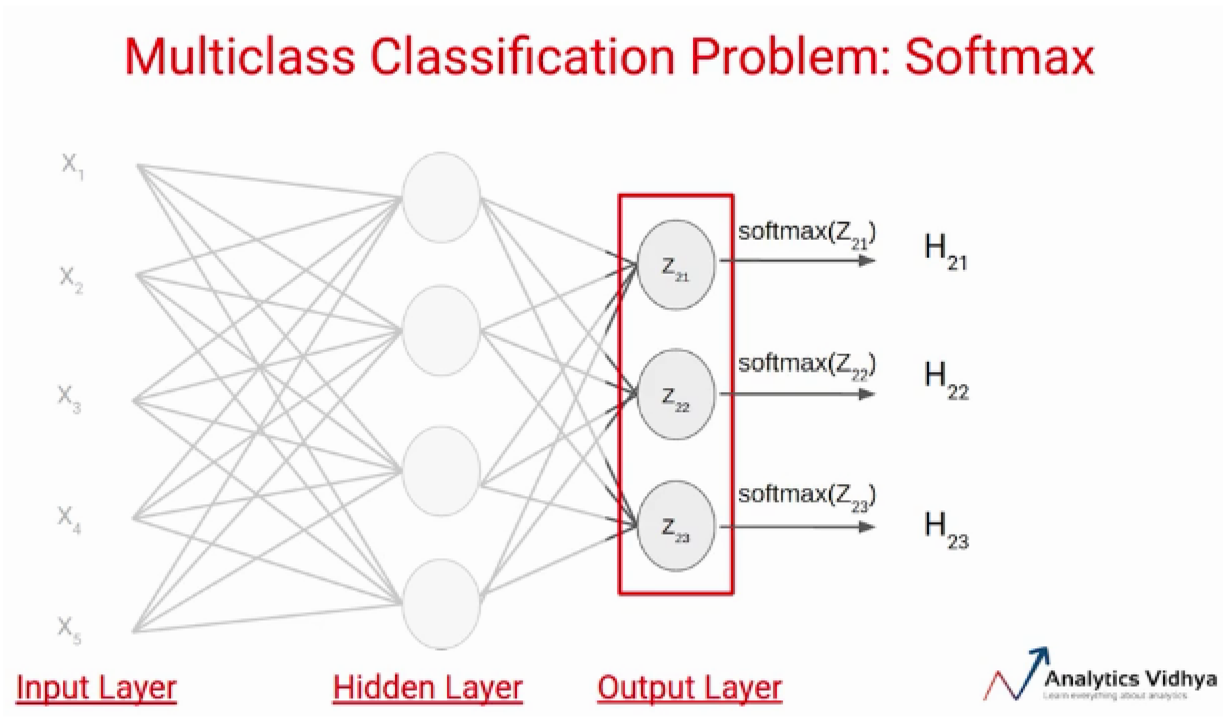


Fig5b: Multiclass classification problem: Softmax

Suppose the value of Z21, Z22, Z23 comes out to be 2.33, -1.46 and 0.56 respectively. Now the SoftMax activation function is applied to each of these neurons and the following values are generated.

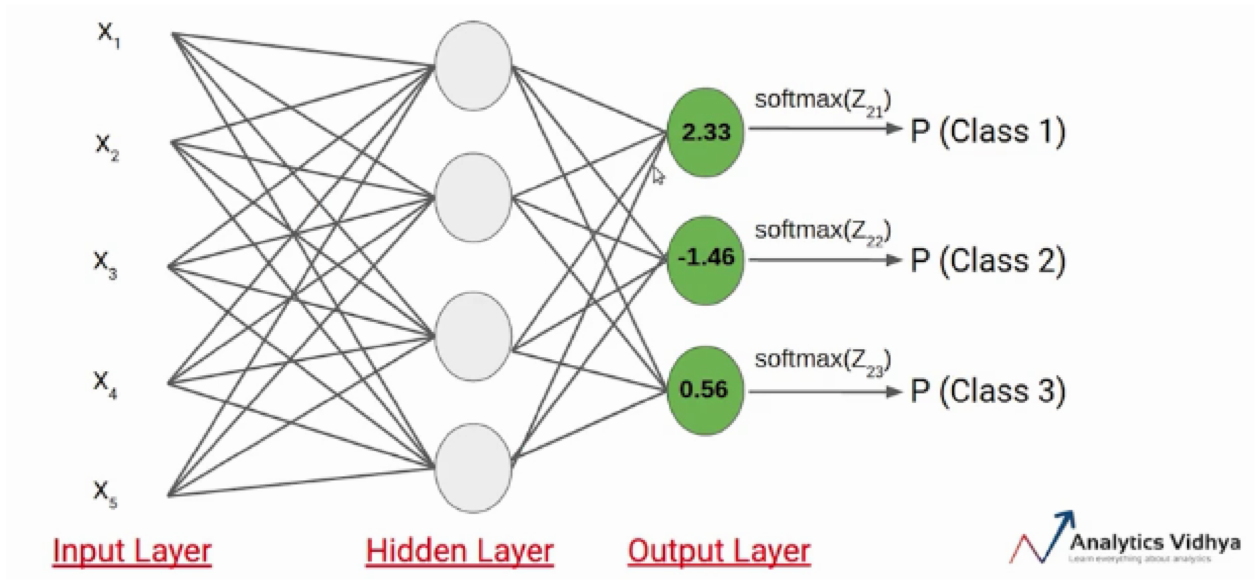
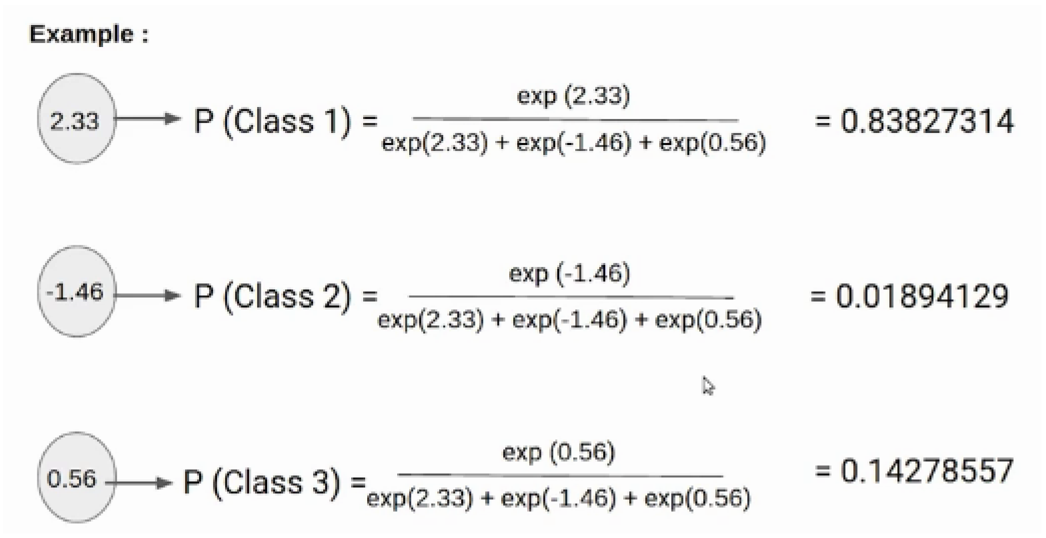


Fig 5c: Output by Softmax

These are the probability values that a data point belonging to the respective classes. The sum of the probabilities is equal to 1.



Example :

$$2.33 \longrightarrow P(\text{Class }1) = \frac{\exp(2.33)}{\exp(2.33) + \exp(-1.46) + \exp(0.56)} = 0.83827314$$

$$-1.46 \longrightarrow P(\text{Class }2) = \frac{\exp(-1.46)}{\exp(2.33) + \exp(-1.46) + \exp(0.56)} = 0.01894129$$

$$0.56 \longrightarrow P(\text{Class }3) = \frac{\exp(0.56)}{\exp(2.33) + \exp(-1.46) + \exp(0.56)} = 0.14278557$$

Hence, the input belongs to class 1.

RESULTS

We have used 2 datasets for hate and offensive language detection. They are as follows -

Description of Datasets

1. **Dataset 1** - The first dataset's name is Hate Speech and Offensive Language Dataset compiled by ANDRII SAMOSHYN on Kaggle and this is the link. It contains 24783 distinct rows and 7 columns. The columns are 'Unnamed:0', 'count', 'hate_speech', 'offensive_language', 'neither', 'class', and 'tweet'. The dataset contains 3 classes, hate speech - which is labeled as 0, offensive - which is labeled as 1 and neither - which is labeled as 2.
2. **Dataset 2** - The second dataset's name is HASOC 2019.This was made available fromposts and text taken from Twitter and Facebook. This dataset had three sub-tasks. I mainly focused on the first subtask - deciding whether the text has offensive or hateful content or not. The link of the dataset is given here. The dataset has been divided into training and test datasets. The training dataset contains 5853 entries whereas the test dataset contains 1154 entries.

Table 1 - Results for our proposed models for Dataset 1

Models	Accuracy Score	F1 Score	Precision Score	Recall Score
Model A	0.8418079096	0.8368240766	0.8328441034	0.8418079096
Model B	0.8466505246	0.842003635	0.8381640739	0.8466505246
Model C	0.862590799	0.8440688084	0.862590799	0.8446120962
Model D	0.8575464084	0.8529170264	0.849080955	0.8575464084
Model E	0.8547215496	0.8553861278	0.85606357	0.8547215496
Model F	0.849677159	0.8223700372	0.849677159	0.8307185236

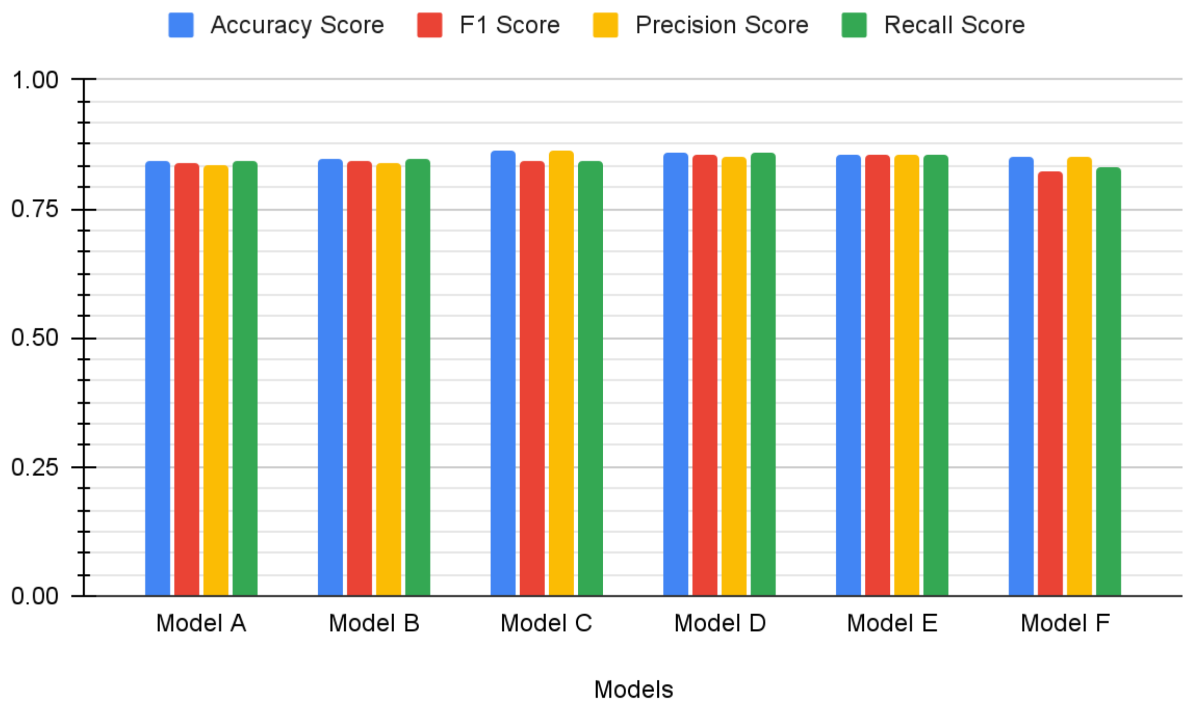


Fig 6a: Comparison of results obtained by our models. Model A: BERT+LR, Model B: NB+BERT+LR(without using threshold on NB's output), Model C: NB+BERT+LR(using threshold on NB's output), Model D: BERT+Dense layer+Softmax, Model E: NB+BERT+Dense layer+Softmax layer (without using threshold value on NB's output), Model F: NB+BERT+Dense layer+Softmax layer (using threshold value on NB's output)

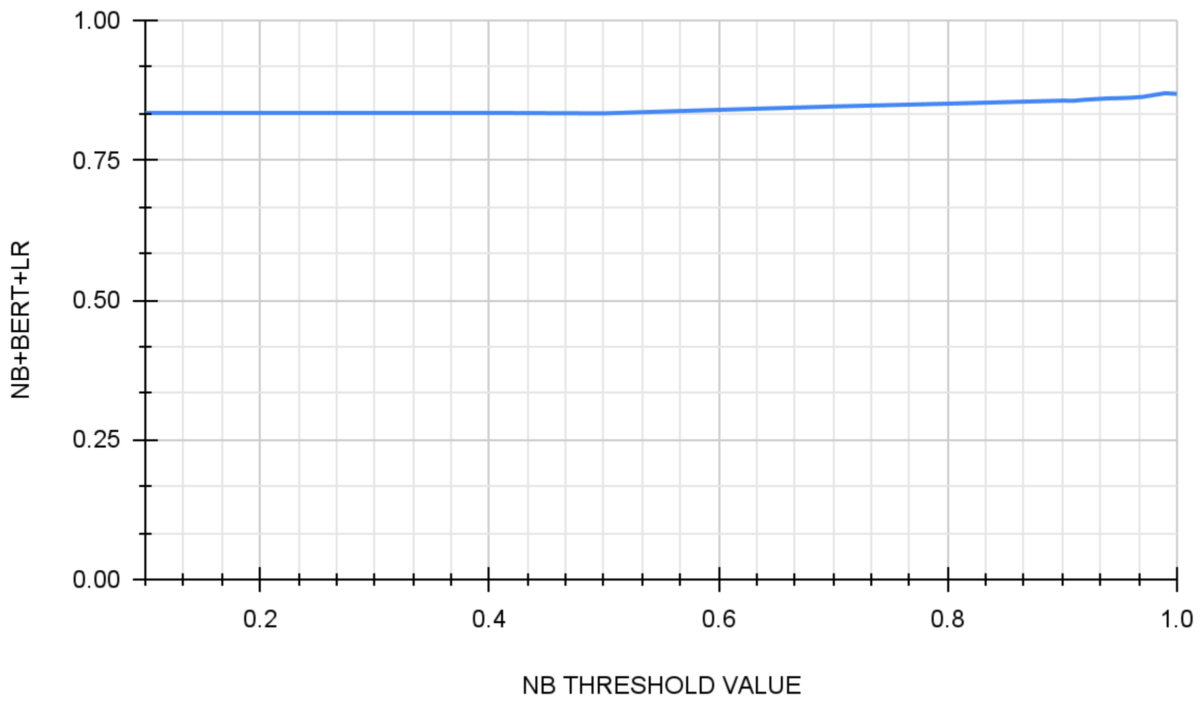


Fig 4b: Effect on model accuracy when threshold on NB’s output is varied

<u>Result obtained from dataset2:</u>

Table 2 - Results for our proposed models for Dataset 2

Models	Accuracy Score	F1 Score	Precision Score	Recall Score
Model A	0.6594454073	0.6549326588	0.6520852999	0.6594454073
Model B	0.6585788562	0.6539291676	0.6510244204	0.6585788562
Model C	0.6724436742	0.6576466023	0.6724436742	0.6561874437
Model D	0.6663778163	0.6631362031	0.6609887647	0.6663778163
Model E	0.6672443674	0.6680854409	0.6690257641	0.6672443674
Model F	0.7062391681	0.6762404805	0.7062391681	0.6888290328

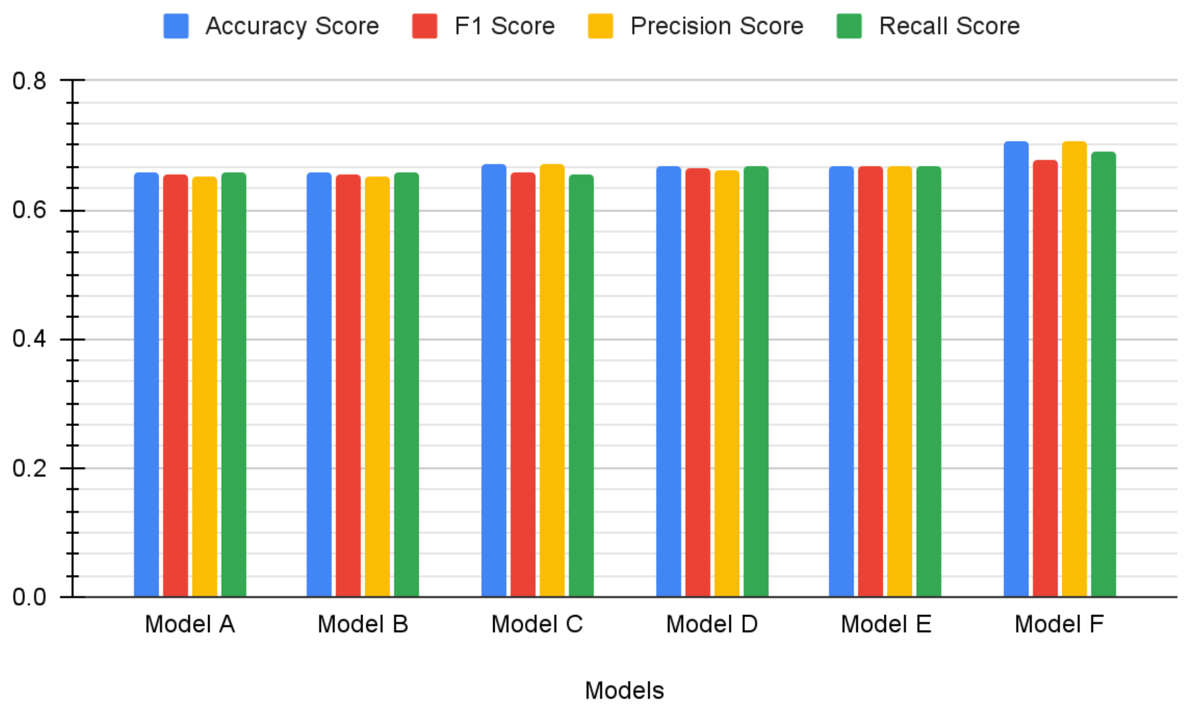


Fig 6c: Comparison of results obtained by our models. Model A: BERT+LR, Model B: NB+BERT+LR(without using threshold on NB's output), Model C: NB+BERT+LR(using threshold on NB's output), Model D: BERT+Dense layer+Softmax, Model E: NB+BERT+Dense layer+Softmax layer (without using threshold value on NB's output), Model F: NB+BERT+Dense layer+Softmax layer (using threshold value on NB's output)

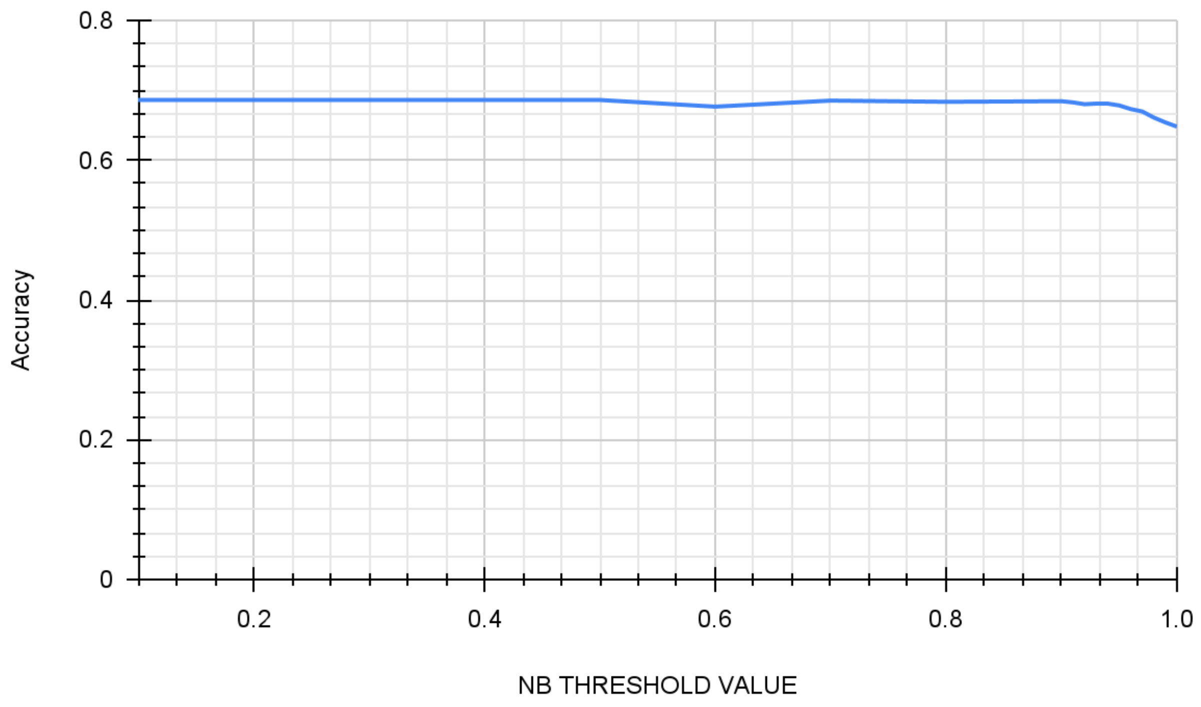


Fig 6d: Effect on model accuracy when threshold on NB's output is varied

REFERENCES

1. **Naive Bayes:**
    a. **https://deepakdvallur.weebly.com/uploads/8/9/7/5/89758787/ml-lab6-doc.pdf**

2. **BERT:**
    a. **https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/**
    b. **https://huggingface.co/docs/transformers/model_doc/bert**

3. **Logistic Regression:**

    a. **https://kavita-ganesan.com/news-classifier-with-logistic-regression-in-python/#.YrSvNXZBxEY**
    b. https://kavita-ganesan.com/news-classifier-with-logistic-regression-in-python/#.YrSvNXZBxEY
    c. https://kavita-ganesan.com/news-classifier-with-logistic-regression-in-python/#.YrSvNXZBxEY

4. **Dense layer**

    a. **https://keras.io/api/layers/core_layers/dense/**
    b. **https://www.tutorialspoint.com/keras/keras_dense_layer.htm**

5. **Softmax layer**

    a. **https://keras.io/api/layers/activation_layers/softmax/**
    b. **https://www.analyticsvidhya.com/blog/2021/04/introduction-to-softmax-for-neural-network/**

6. Overview of the HASOC track at FIRE 2019: Hate speech and OffensiveContent Identification in Indo-European Languages; Author: Sandip Modha, Thomas Mandi, Prasenjit Majumder and Daksh Patel; Year: 2019