JADAVPUR UNIVERSITY

Hate Speech and Offensive Language Detection using Naive Bayes, SVM and BERT

BY

BHASKAR GARAI

EXAMINATION ROLL NO. – MCA226040

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 Naive Bayes, SVM and BERTI**" has been carried out by **Bhaskar Garai (**University Roll Number **- 001910503041,** Examination Roll No **- MCA226040,** University Registration Number **- 149901** 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 Naïve Bayes, SVM and BERT” is a bonafide record of work carried out by Bhaskar Garai 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 Naive Bayes, SVM and BERT**” 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) - BHASKAR GARAI

Roll No. - 001910503041

Examination Roll No. - MCA226040

University Registration No. - 149901

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.

Bhaskar Garai

Roll No.: 001910503041

Reg. No.: 149901

Examination Roll No.: MCA226040

CONTENTS

ABSTRACT

Nowadays, with the rise in the number of smartphones and laptops around the world, social media has become one of the most popular and an important sphere in everyone's lives. It has become one of the most used platforms for communication and dissemination of information throughout the world. But sometimes, along with this content, there also exists hateful and offensive content in these platforms, which is unwanted. Hate speech can be defined as any form of speech that expresses prejudice or bias against a particular group of people on the basis of race, religion, caste, creed etc. This also includes online bullying, cyber bullying, profanity etc. Offensive speech can also be defined as speech that makes the listener feel annoyed, upset, angry or experience any other negative emotions. In this paper, an attempt has been made to classify 2 datasets – one dataset has been used to classify text as containing hate-or-offensive-text or not containing hate-or-offensive-text, while the other dataset has been used to classify text as having hate text or offensive text or having neither hate-or-offensive-text. The ML models used are ensemble models using only BERT+SVM and NB+BERT+SVM. For the dataset having the 3 classifications, the accuracy, f1, precision and recall scores are respectively 87.65, 85.31, 85.23, 87.65 while for the dataset having the two classifications, the scores are respectively 70.53, 68.16,69.37 and 70.53 respectively.

INTRODUCTION

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 kind 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+SVM while the other model is NB+BERT+SVM.

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, either through NB+SVM or simply through SVM to get the output. After fine-tuning the parameters, the dataset with 2 classes has accuracy, f1, precision and recall score as 0.7053, 0.6816,0.6937 and 0.7053 while the dataset with 3 parameters has 0.8765, 0.8531, 0.8523, 0.8765

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 SVM(Support Vector Machine) to predict the class of the text.

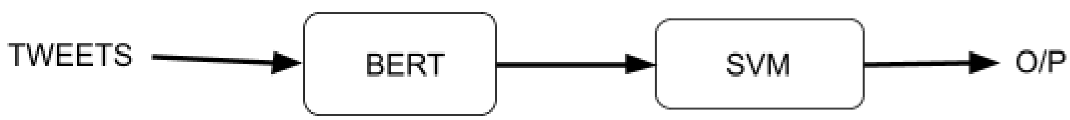


Fig 1a: BERT+SVM

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 SVM(Support Vector Machine).

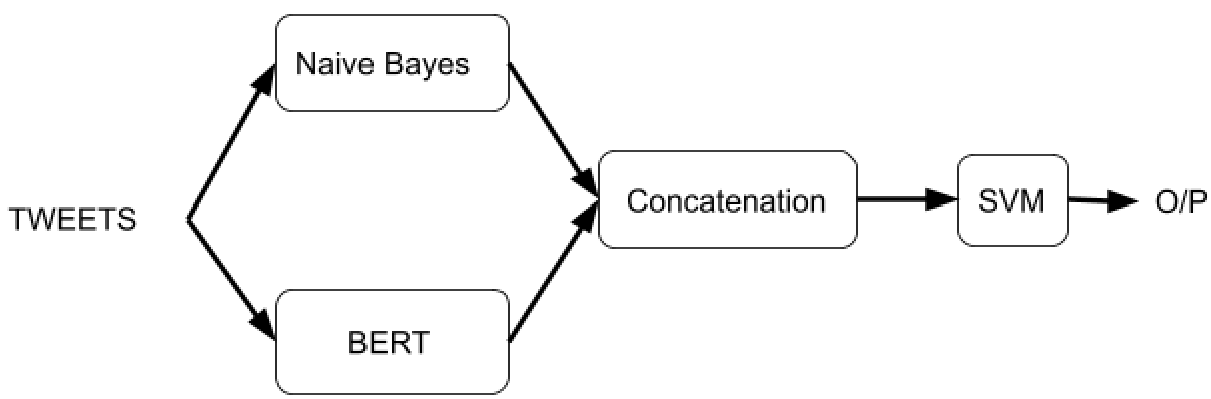


Fig1b: NB(without threshold value)+BERT+SVM

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 SVM(Support Vector Machine) we train the model using training data and test the model using the testing data.

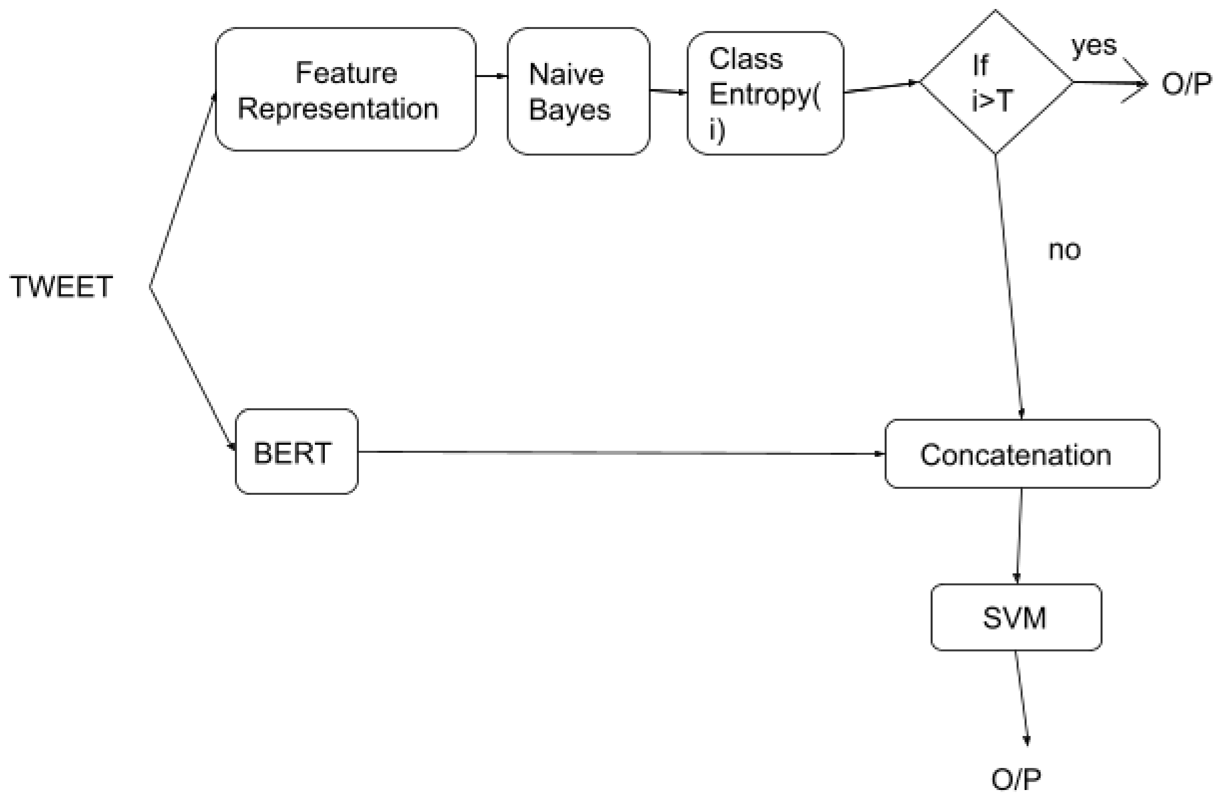


Fig1c: NB(with threshold value)+BERT+SVM

NAÏVE BAYES CLASSIFIER

Suppose, Tweet is a set of text documents along with their target values. V is the setoff 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 Tweet.

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

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

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	Bitches 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	really	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(porch | 3:2)=(0+1)/(10+20)=1/30=0.0333

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

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

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

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

P(rihanna | 3:2)=(1+1)/(10+20)=2/30=0.0667

P(is | 3:2)=(2+1)/(10+20)=3/30=0.1000

P(chris | 3:2)=(1+1)/(10+20)=2/30=0.0667

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

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

P(RT | 3:2)=(2+1)/(10+20)=3/30=0.1000

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

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

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

P(really | 3:2)=(1+1)/(10+20)=2/30=0.0667

P(trash | 3:2)=(2+1)/(10+20)=3/30=0.1000

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 <u>bank</u> 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 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.

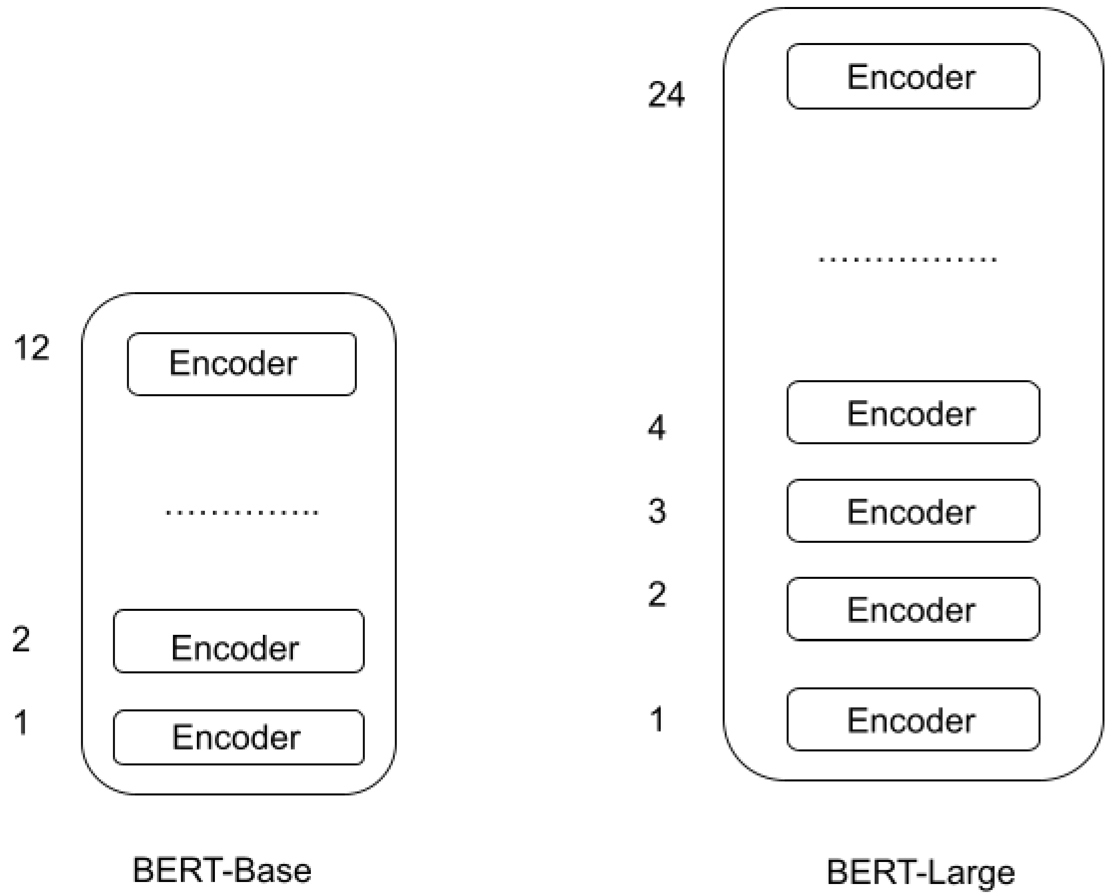


Fig2a: BERT architecture

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.

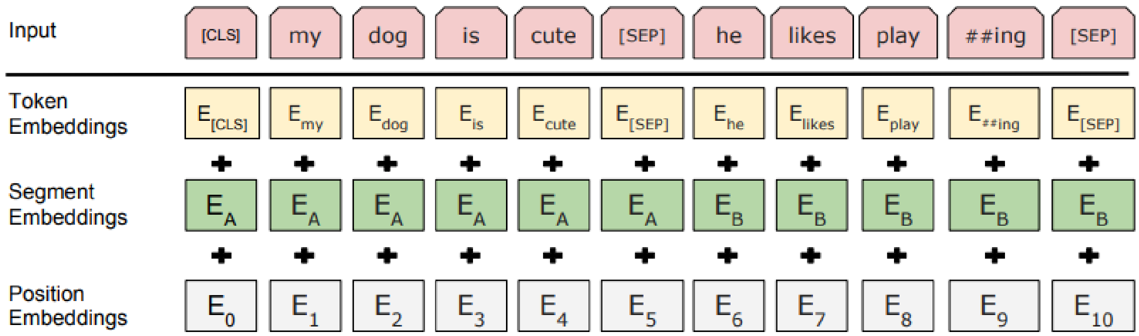
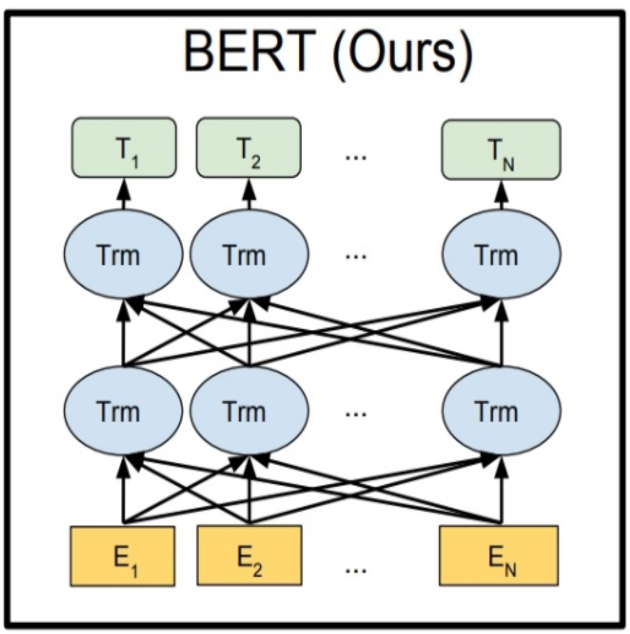


Fig 2b: Embeddings in BERT

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.



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.

SVM

Support Vector Machine Algorithm (SVM) is a supervised machine learning algorithm used for both classification and regression. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data point. The dimension of the hyperplane depends upon the no. of features.

Let us consider two independent variables x1, x2 and one dependent variable which is either a blue circle or a red circle.

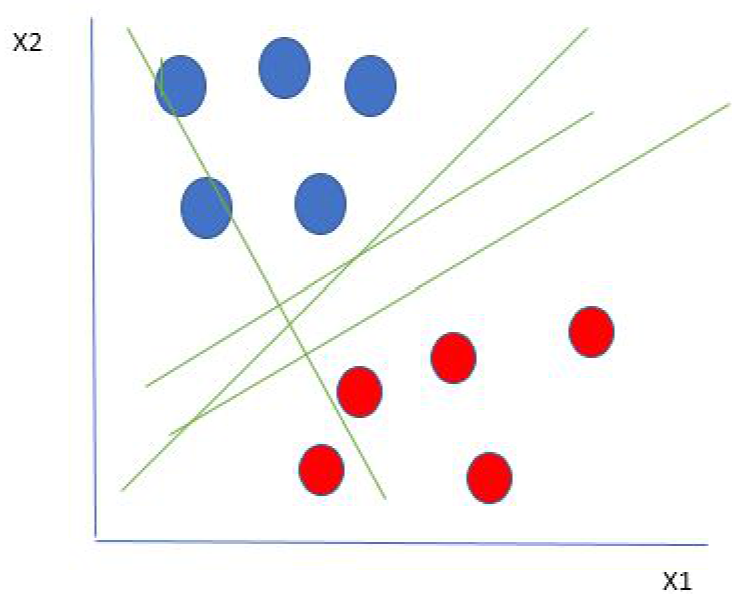


Fig3a: Case1- Linearly separable data points

It is very clear that there are multiple lines that segregates our dat points or does a classification between the red and blue circles.

Now, to choose the best hyperplane.

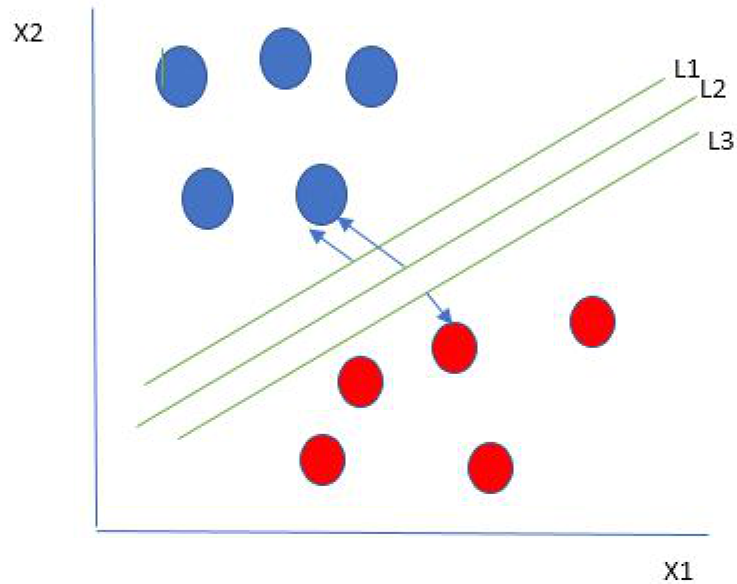


Fig3b: Selection of hyperplane

We choose the hyperplane whose distance from it to the nearest data point on each side is maximized and call it maximum-margin hyperplane. Here, we choose L2.

Consider the scenario where we have one blue ball in the boundary of the red ball. This blue ball is an outlier.

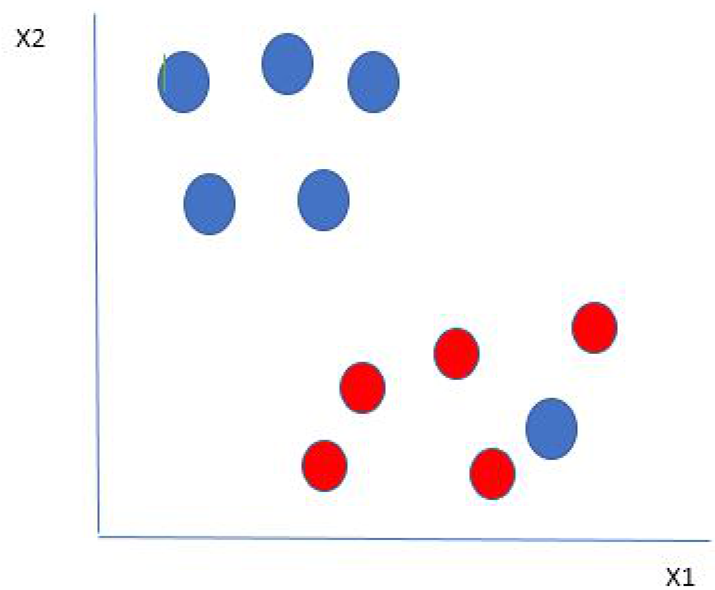


Fig 3b: Case2- In case of outlier

The SVM finds maximum margin as done previously and adds a penalty each time a point crosses the margin called soft margin. The SVM tries to minimize (1/margins^(Σpenalty)).

If the data is linearly separable, SVM solves this by creating a new variable using a kernel. We call a point x_i on the line and we create a new variable y_i as a function of distance from origin o.

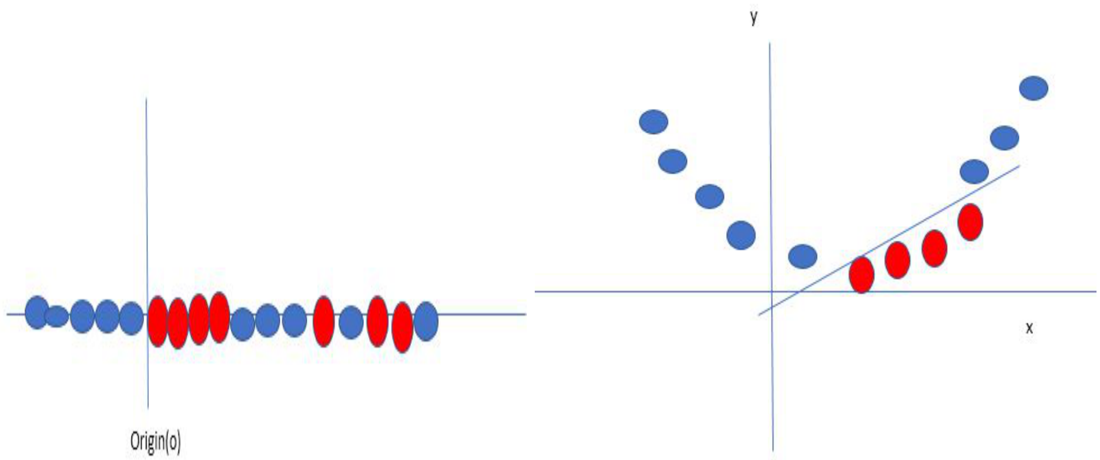


Fig3c: Solution of outlier using kernel

In this case, the new variable y is created as a function of distance from the origin. A nonlinear function that creates a new variable is referred to as a kernel.

Advantages:

1. Effective in high dimensional cases.
2. It is memory efficient as it uses a subset of training points in the decision function called support vectors.
3. Different kernel functions can be specified for the decision functions and it is possible to specify custom kernels.

In our work we have used the parameters discussed below:

1. C: float, default=1.0

   Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

2. kernel: {'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable, default='rbf'

Specifies the kernel type to be used in the algorithm. If none is given, ‘rbf’ will be used. If a callable is given it is used to pre-compute the kernel matrix from the data matrices; that matrix should be an array of shape (n_samples, n_samples).

3. degree: int, default=3

   Degree of the polynomial kernel function (‘poly’). Ignored by all other kernels.

4. gamma: {‘scale’, ‘auto’} or float, default=’scale’

   Kernel coefficient for ‘rbf’, ‘poly’, and ‘sigmoid’

a. If gamma=’scale’ (default) is passed then it uses 1/(n_features*X.var()) as value of gamma
b. If ‘auto’, uses 1/n_features

RESULT

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.

Result obtained from dataset1:

Table 1 - Results for our proposed models for Dataset 1

Models	Accuracy Score	F1 Score	Precision Score	Recall Score
BERT+SVM	0.8706618240516546	0.8465132970665215	0.8487788236117944	0.8706618240516546
NB+BERT+SVM	0.8765133171912833	0.8531834814212823	0.852399276445212	0.8765133171912833
NB+BERT+SVM with threshold probability	0.8684422921711057	0.8408517850821733	0.8480613543507771	0.8684422921711057

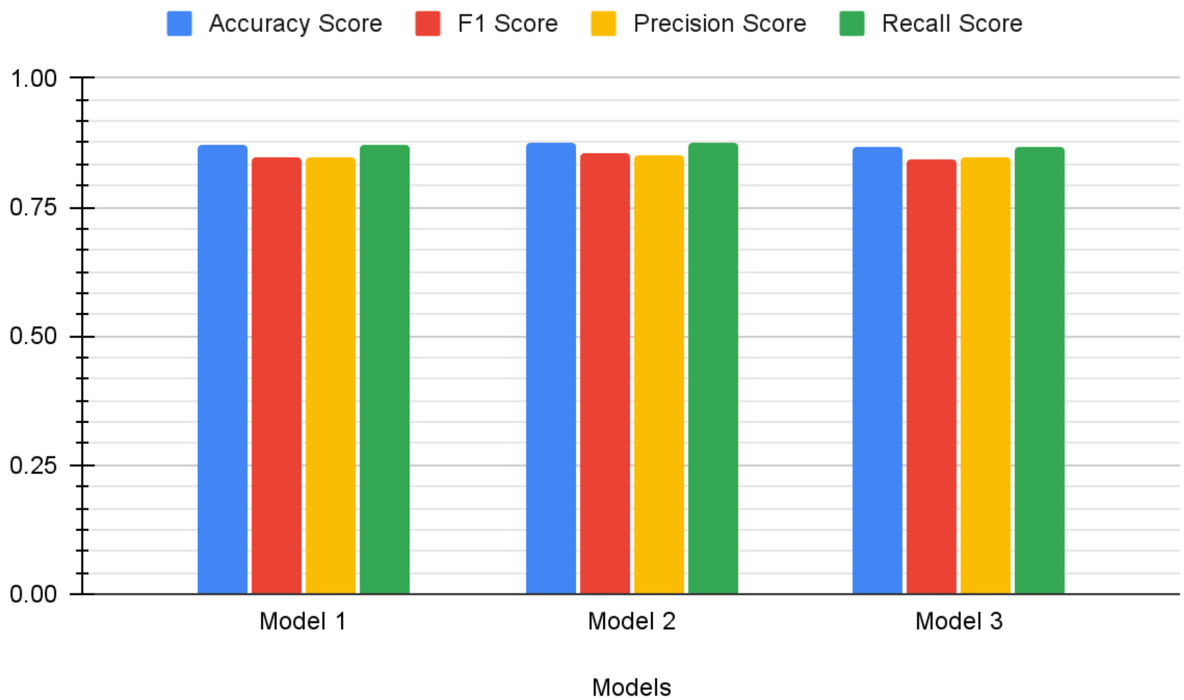


Fig 4a: Comparison of results obtained by our models. Model1: BERT+SVM, Model2: NB+BERT+SVM(without using threshold on NB's output), Model3: NB+BERT+SVM(using threshold on NB's output)

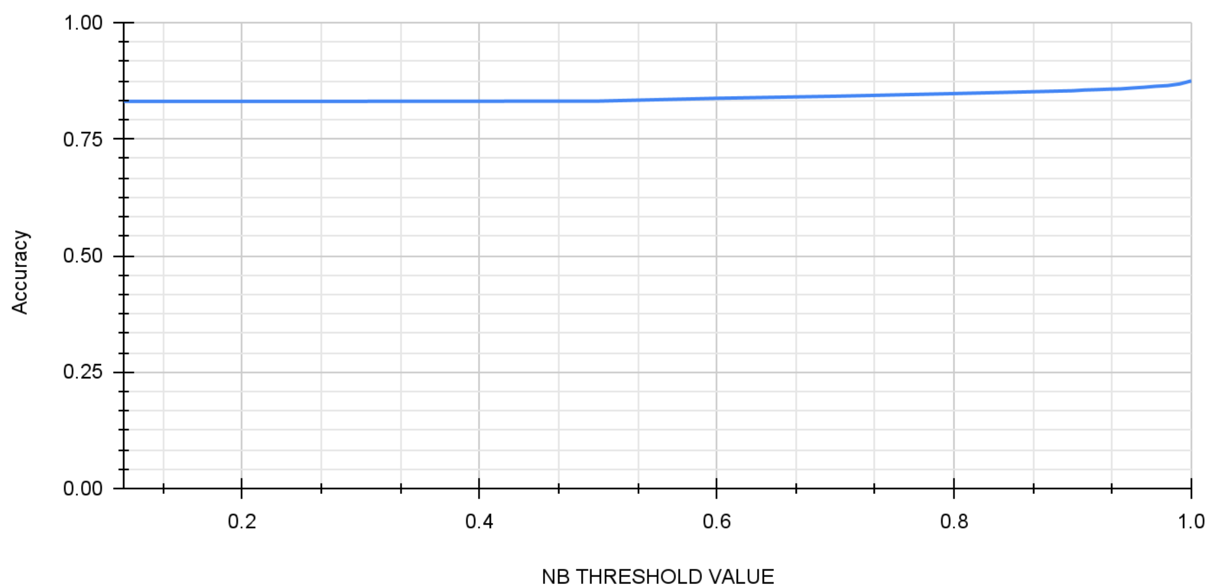


Fig 4b: Effect on model accuracy when threshold on NB's output is varied

Result obtained from dataset2:

Table 2 - Results for our proposed models for Dataset 2

Models	Accuracy Score	F1 Score	Precision Score	Recall Score
BERT+SVM	0.7027729636048526	0.6819101984194148	0.6899885660853452	0.7027729636048526
NB+BERT+SVM	0.7027729636048526	0.6815166919200326	0.6900302503966603	0.7027729636048526
NB+BERT+SVM with threshold probability	0.7053726169844021	0.6816740435199768	0.693770400170391	0.7053726169844021

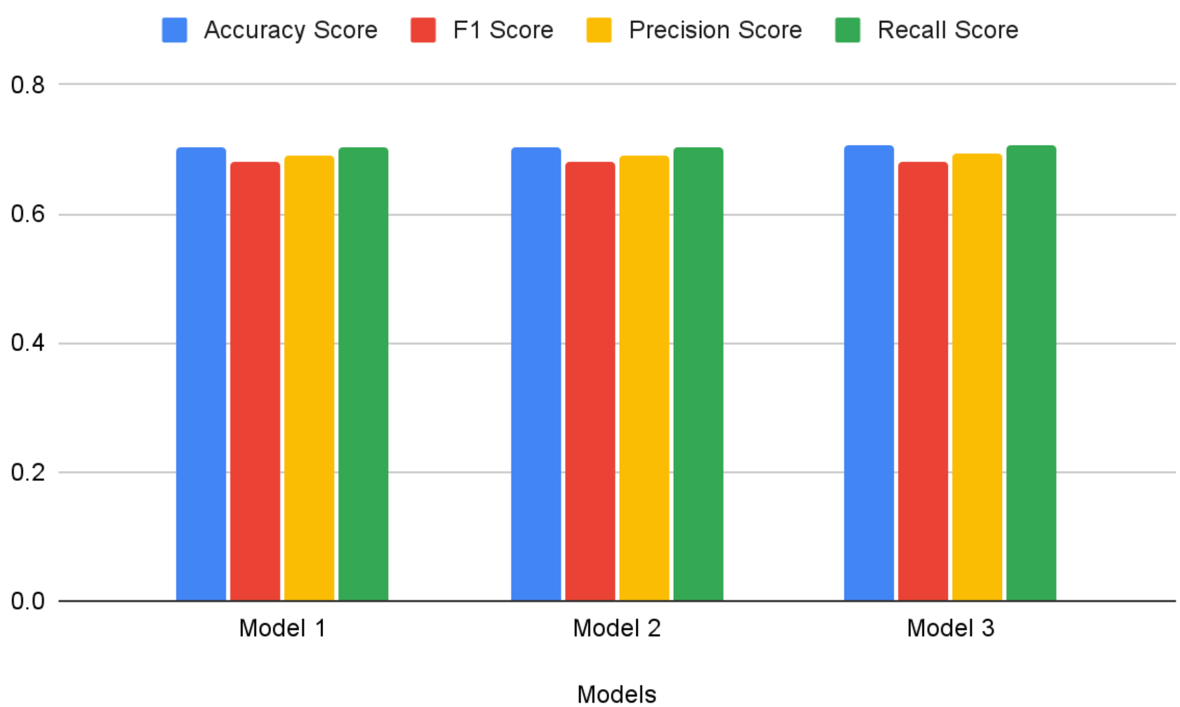


Fig 4c: Comparison of results obtained by our models. Model1: BERT+SVM, Model2: NB+BERT+SVM(without using threshold on NB's output), Model3: NB+BERT+SVM(using threshold on NB's output)

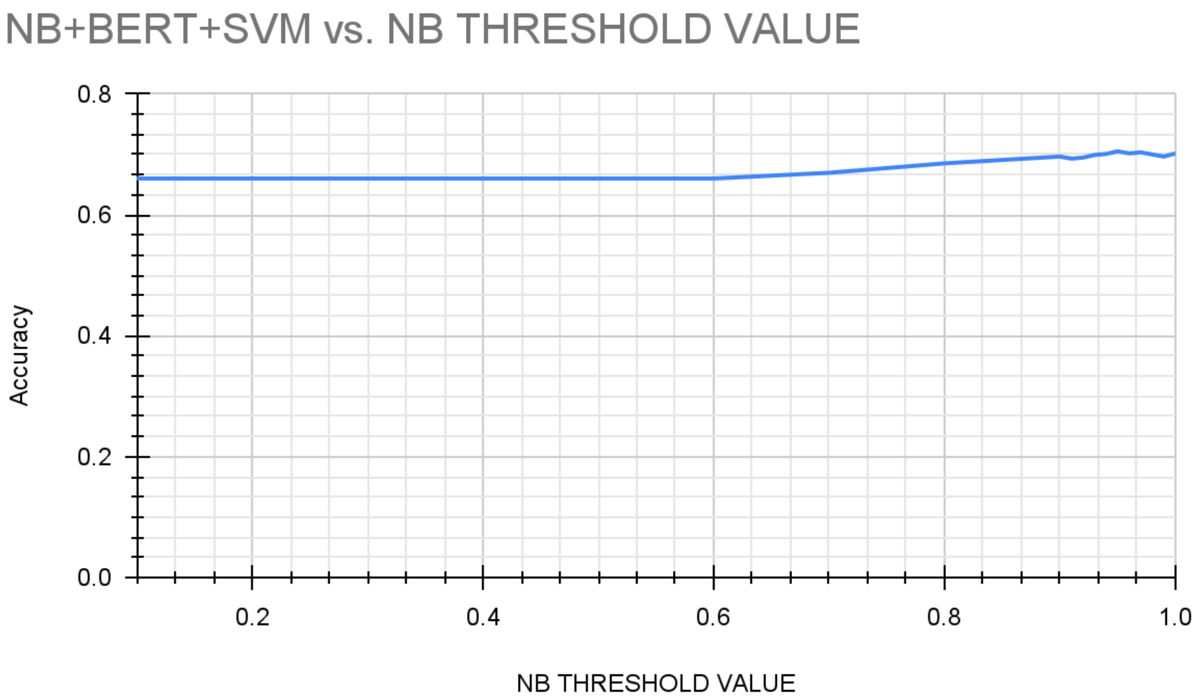


Fig 4d: Effect on model accuracy when threshold on NB's output is varied.

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. **SVM:**
   a. https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/
   b. https://www.geeksforgeeks.org/introduction-to-support-vector-machines-svm

4. 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: 201