

Multi label pernicious comment classification using Machine Learning Algorithm

Jotaniya Hitesh Ghanshyambhai¹, Dr. Kinjal Adhvaryu²

¹Research Scholar, Shankersinh Vaghela Bapu Institute of Technology, Gandhinagar
Gujarat Technological University

²Principal, Shankersinh Vaghela Bapu Institute of Technology, Gandhinagar
Gujarat Technological University

Abstract: Millions of comments are sent day to day on internet, and there are many hidden pernicious comments in these contents. Manually identifying and detecting pernicious content in these comments are nearly impossible task. There is a need to use Machine Learning techniques to identify or to detect pernicious content present in their comments. Not only that there is a need to categories them in different levels like toxic, severe toxic, obscene, threat, insult, and identity-hate.

Thus, we address the task of automatically detecting pernicious comments in user generated texts. Predicting the comments contain any pernicious content or not. Also listing them into different labels.

The Challenge of Automatically Detecting of pernicious comments on huge set of Commented Data; and also classifying them into six different pernicious labels; like (toxic, severe toxic, obscene, threat, insult, and identity-hate) using various Machine Learning models or algorithms will be definitely beneficial to our social community.

Various Machine Learning algorithms BERT (Bidirectional Encoder Representations from Transformers), CNN, Bi-gram and K-fold cross validation is being applied; then the result is compared with its efficiency and accuracy; reports will be generated.

Keywords: BERT (Bidirectional Encoder Representations from Transformers), CNN - convolutional neural network, KNN- k nearest neighbour, k-fold cross validation, NLP – Natural Language Processing.

2. INTRODUCTION

Internet is the biggest platform to represent individuals' skills. And Most of the website provider facilitate of commenting on their product or service. However, there is chances that someone use abominable language in his/her comments; cyber-bullying, homosexual attacks, offensive words, abusive words and hateful language in comments.

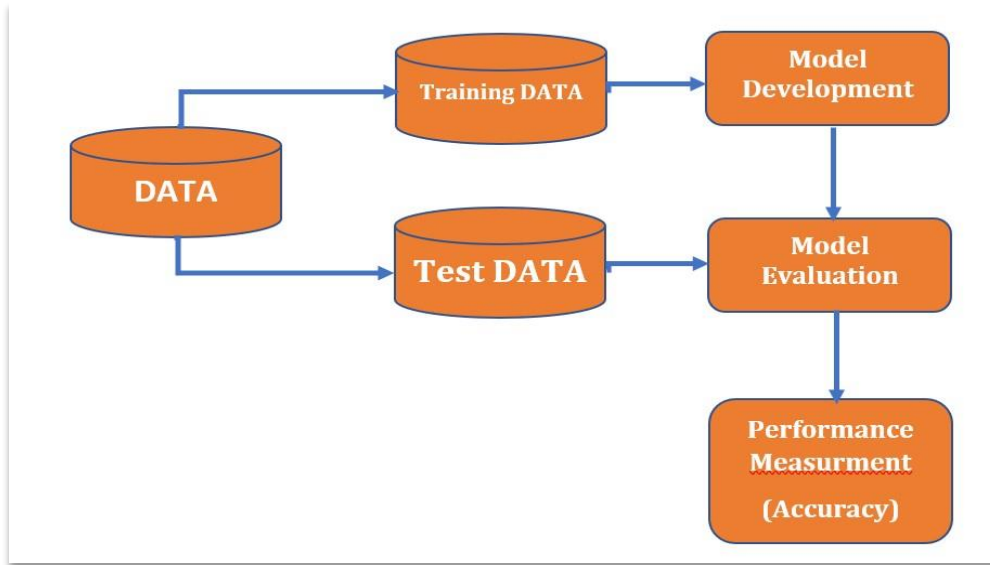
These stuffs impact very negatively on social media. Individual product/service selling businesses, or young generation or even society community receives negative impact. This kind of work must need to be avoided, detected and removed by Comment service providers.

But Detecting "Manually"; "pernicious comments" present in Comment or not, is too tedious and nearly impossible task in this current fast immerging era. Thus, requires techniques of Machine Learning algorithms^[1] to detect and report about its presence.

we have use Machine Learning models (BERT, CNN, K-fold cross validation) to Data Set obtained online. Analyse their result and report to classify comments into six different categories^[2] like: toxic, severe toxic, obscene, threat, insult, and identity-hate. Also, measing algorithm's accuracy and losses.

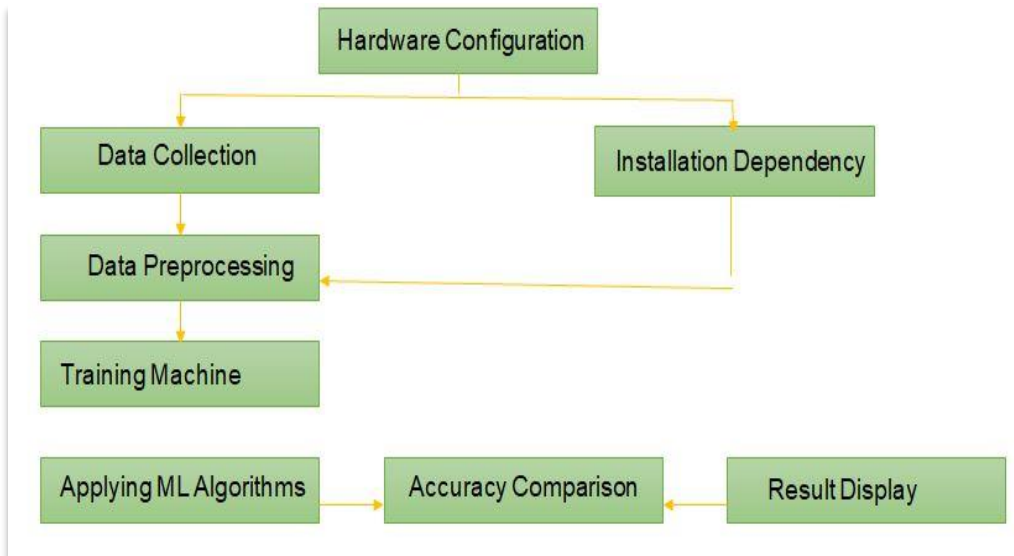
3. METHODOLOGY

our research is focused on supervised learning model; here training data will be fed to Machine Learning model; BERT is used to pre train model. Then also evaluating their result to print the confusion matrix and f1 score, precision and recall, accuracy. later generating report of that.



[Figure1 : Methodology]

In the following diagram; we have represented steps of sequences that are followed to comment classification.



[Figure2: Steps of Methodology]

4. TOOLS AND TECHNOLOGY:

- o Programming Language & Version: Python 3.9
- o Open-Source Software Distribution: Anaconda3
- o IDE: Pycharm Community Edition 2023
- o Libraries: Pandas, Numpy, Scipy, Matplotlib, NLTK, Sklearn, Tokenizers, Pytorch
- o Technology: AI
- o AI Backend model training framework: Pytorch

Machine learning algorithms can only make accurate predictions by learning from previous examples. For that Dataset must be obtained and trained well for this purpose. The Dataset is downloaded from Kaggle.com [2]. Python is mostly used by developers and data scientist who work with machine learning models.

Pandas: The Pandas library is very robust piece code and having various advantages.

NumPy: NumPy is a python package. It is mainly used for scientific and numerical computing. NumPy is a conjunction of two words, “Numerical” and “Python”. It is very well famous for data scientists and analysts. It has great efficiency (run time speed) and support wide range of array operations.

Matplotlib: Matplotlib is also a python library. It is used to create 2D graphs and plots. It has module “pyplot” which makes easy plotting, line styles, font attributes, axes formation etc. It supports huge variety of graphs and plots. It is used for generally drawing - histogram, bar charts, power spectra, error charts etc. It can use NumPy environment as well. It can also plot graphics.

NLTK: NLTK is most popular in education and research. It has led to many breakthroughs in text analysis. It has a lot of pre-trained models and corpora which helps us to analyse things very easily. It is an excellent library when we require a specific combination of algorithms

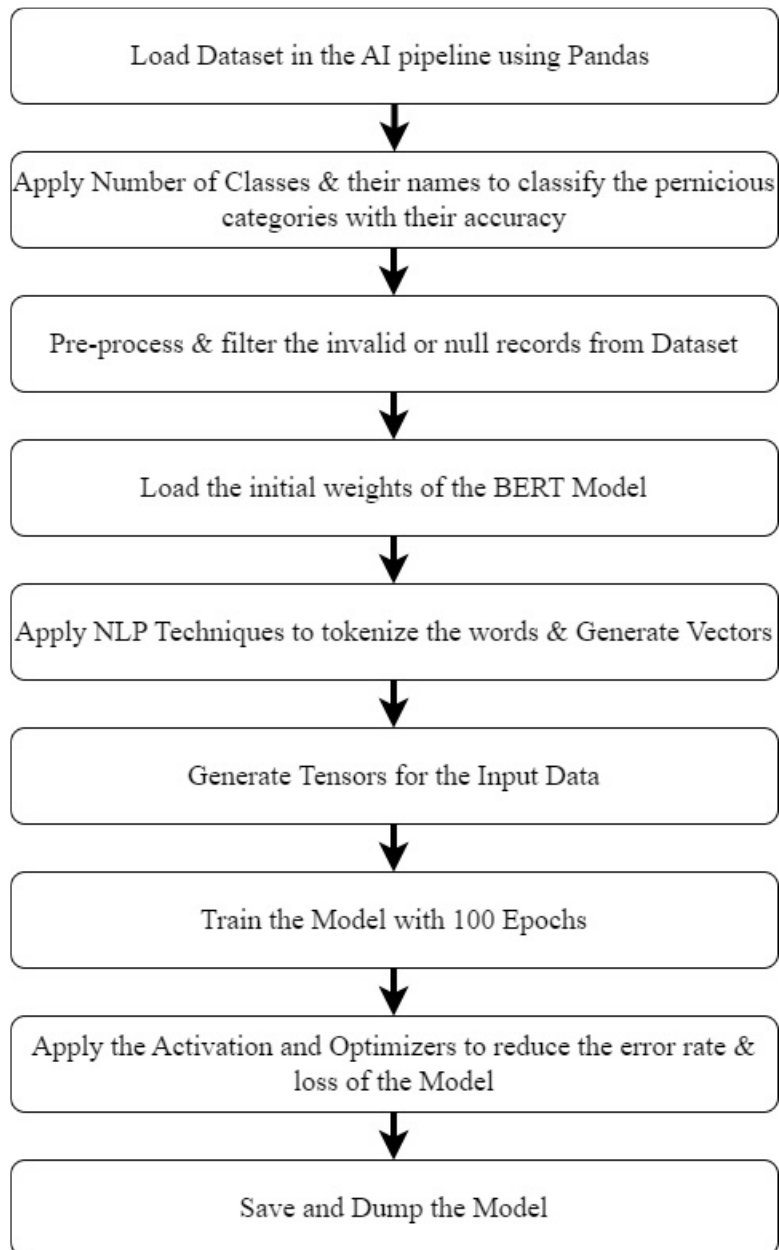
Sklearn: SkLearn is an open-source data analysis library, and the very popular for Machine Learning (ML) in the Python system. Key concepts and features include:

PyTorch: PyTorch is open-source machine learning framework for Python. It is mostly used for developing Machine Learning and Deep Learning models.

Pycharm: PyCharm is a dedicated Python Integrated Development Environment (IDE). It provides a wide range of tools for Python developers. It has integrated and convenient environment for Python programs, web, and data science development.

Anaconda: Anaconda is fast and convenient environment. Having significant advantage is ease of installation. Installing packages and libraries are easier. Below are other advantages of using anaconda.

5. TRAINING PLAN SEQUENCE:



[Figure 3: Training Flow Sequence]

- **Pernicious Comments (DataSet):** Dataset will be obtained from Kaggle.com. Data Set is pre classified into toxic, sever_toxic, obscene, threat, insult, identity_hate.

- **Load Dataset in AI pipeline using Pandas:**

We have loaded our dataset using pandas; here we have used `pandas.read_csv()` function.

code to load dataset:

```
import pandas as pd
df = pd.read_csv('train.csv')
```

- **Apply the number of classes and their names to classify the pernicious category with their accuracy:**

Here, to classify the pernicious categories with their accuracy in Python, I have used a classification algorithm “Logistic Regression”.

- **Pre-process and filter the invalid or null records from the dataset:**

Once the dataset is loaded into a framework using `pandas.read_csv()`; it is important step to pre-process and filter the invalid and null records. Thus, Pre-processing and filtering invalid or null records from the dataset is very important step in data cleaning and preparation activity. Here, we have used following methods for this task:

IsNull() : used for checking for null or missing values in dataset.

Dropna() : to drop the rows and columns containing missing and null values.

To_csv() : to save pre-processed and filtered dataset in to a new CSV file.

Once; the pre-processing is done; We are ready with perfect dataset for further processing.

- **Load the initial weights to the BERT Model:**

Using PyTorch library I have loaded initial weights into a BERT model. BertModel and BertConfig modules are also used to set up configuration.

Once Model is configured; `load_state_dict()` is used to pre-trained weights. and finally `torch.load()` loads weights into model.

BERT: BERT (Bidirectional Encoder Representations from Transformers) is very popular pre-training model for NLP. It was developed by Google in 2018. BERT is pre-trained on a huge set of text using a masked language modeling (MLM) and next sentence prediction (NSP) objective. NSP is the model trained to predict whether two sentences are consecutive or not.

- **Apply NLP techniques to tokenize the words and generate vectors:**

NLP techniques “tokenization” and “vectorization” are very important for NLP tasks. These techniques play crucial role in text classification, machine translation, sentiment analysis, and named entity recognition.

Tokenization: This process breaks the text into individual tokens, such as words, sentences or phrases. These are further processed by other NLP techniques in text classification. Here each token is assigned a weight that represent its importance in degerming the category of the text.

Vectorization: This process represents each token as a vector of numbers. It is necessary because it makes machines to work with text data. By representing text into vectors; machines can perform mathematical operations on the vectors. Vectorization is critical in NLP tasks because it makes machines to learn from the data and make predictions.

Efficiency: Tokenization and vectorization techniques reduces the amount of data needed to process, so as a result it significantly improves the efficiency. By converting text into individual tokens; later representing each token as a vector; AI models focus on only the important information in the text also ignores irrelevant details. Thus, Training time is reduced “Faster training” as well as accuracy is also improved.

- **Generate the tensors for the input data:**

In text classification, generating tensors for given data is converting text data into a numerical representation. This representation is faded into a machine learning model. Tensors are multi-dimensional arrays. These arrays hold numerical data and are commonly used in ML.

- **Train Model with 100 Epochs:**

Training a ML model with epochs is involving “iterating over the entire training dataset multiple times”. Each iteration on the training dataset is called an epoch. During each epoch or iteration, the model updates its weights on the basis of the loss between the “predicted output” and the “actual output”. Using 100 epochs, I

make model to learn more thoroughly from the training data. As a result, it leads to improved performance on the test or validation data.

To make model learns more efficiently balance between underfitting and overfitting is also considered. Training the model with too few epochs can place in to underfitting, due to this model is not able to learn enough. While, On the other hand, training the model with so many epochs can bring to overfitting ^[11], where the model memorizes the training data and performs badly on new data. Overfitting also increase computational cost. So, important to keep a balance between underfitting and overfitting.

- **Apply the Activation and Optimizers to reduce the error rate and loss of the model:**

In ML, Activation functions and Optimizers are used to reduce the error rate as well as loss of a model during training.

Activation functions: we have used Activation function is Sigmoid ^[12]. This is used to introduce non-linearity into the output of (NN) neural network ^[15]. Sigmoid is applied to the output of each neuron in the network; and it transforms the input into the output.

Optimizers: we have used Adam Optimizers here. It updates the weights of NN during training. And minimizes the loss between the predicted output and the actual output.

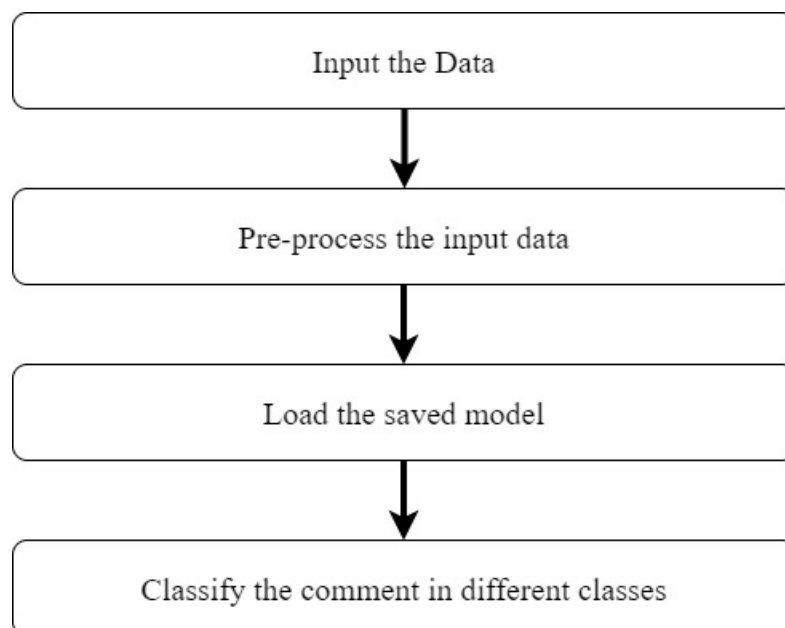
In ML text classification goes through defining the model, compiling model, training the model, evaluating the model, and applying activation function and optimizer ^[17] during training. as a result, it reduces the error rate and loss of the model. Provides better performance on the new data.

- **Save and dump the model:**

In ML, saving and dumping model is refers to process of keeping trained model to disk. So, it can be loaded and used later on. With the use of python libraries, we have saved and dump the model. Standard library for these is pickle.

Dumping the model: Dumping is referring to exporting the trained model.

6. TESTING PLAN SEQUENCE:



[Figure 4: Testing Flow Sequence]

- **Input data:**

Provide new comment by running our model and input comment into the model. Also observe the pernicious content predicted by the model.

- **Pre-process the input data:**

Lower case conversion, removal of non-text elements like URL, removal of all punctuation and brackets. removal of abbreviations, remove repeated letters like Sooooooooo goooooood to so good, remove special symbols like @, #, \$, removal of unnecessary white spaces, removal of NA or Empty elements. Thus, making each comment clean for processing.

- **Load the Saved Model:**

We have loaded the saved model, and it can be achieved in following way:

Here we have used "pickle" library to save and load models. pickle.load() method is used to load model. This method returns instance of saved model. And this instance is used to predict on new data.

- **Classify the pernicious comment classes:**

The model is classifying new given data into percentages of pernicious classes. Here our model is classifying comments into toxic, sever toxic, obscene, threat, insult and identity hate.

7. KEY FEATURES OF WORK:

AI Technique	AI Algorithm	Activation function	Optimizer	Backend Model Training Framework
Multi Class Classification, K-fold cross validation	Binary Cross Entropy	Sigmoid	Adam	Pytorch

[Table 1: Key Features of work]

- **K-fold cross validation:**

K-fold cross-validation is a popular technique used in machine learning to best performance of a model on new dataset. It does partition the dataset into K equal-sized folds, thus called “k-fold”, and then training and evaluating the model K times.

Here are some steps for K-fold cross-validation:

Partition the data: the original dataset is split into K equal-sized folds. The general value of K is taken as 5 or 10.

Train the model: Once the dataset has been partitioned, next step is to train the model K times. In each cycle (iteration), use K-1 (previous) fold for training the model and use the remaining folds for testing the model and get performance.

Evaluate the model: After training, next is evaluate its performance on the test set. Measure the model's performance in terms of accuracy or F1-score or mean squared error.

Repeat steps “Train the model” and “Evaluate the model” K times. Then do average of the performance metrics generated in each iteration, and obtain a final estimate of the model's performance.

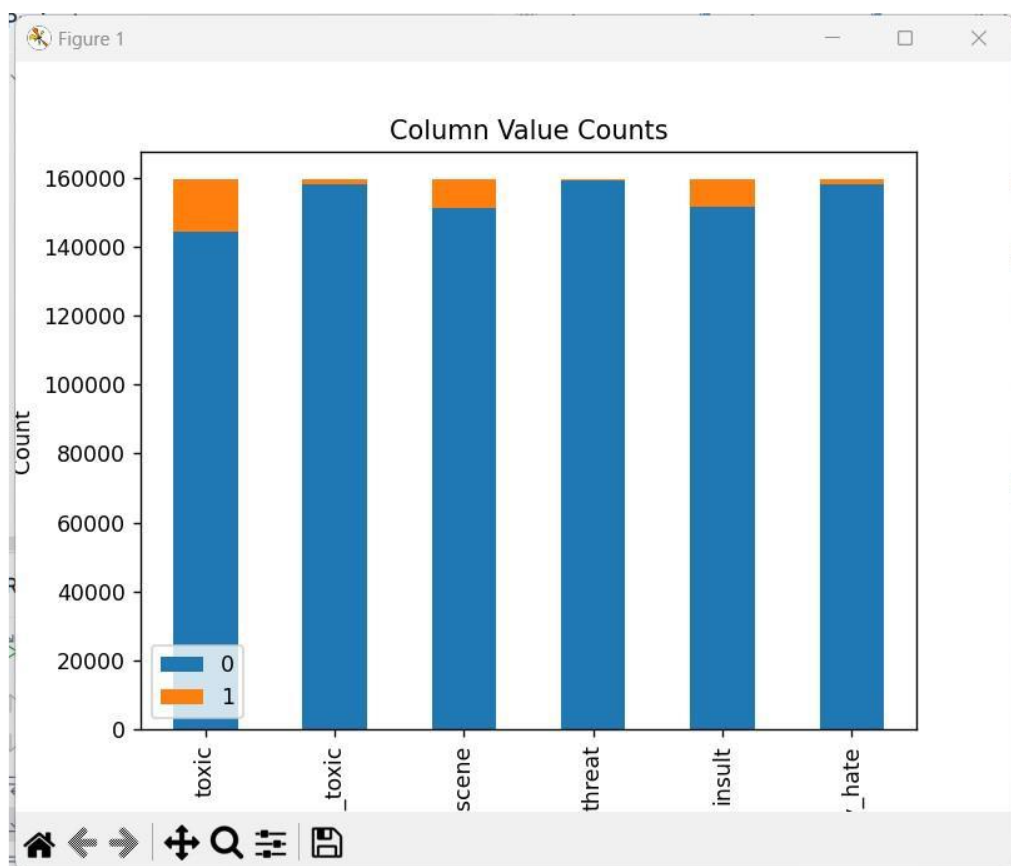
Assess the model: After the completion of the K-fold cross-validation process, the estimated performance metrics used to assess the model's performance [20].

The main advantage of using K-fold cross-validation is that it gives us a precise and more accurate estimation of the model's performance on new unknown data; compare to traditional hold-out validation techniques.

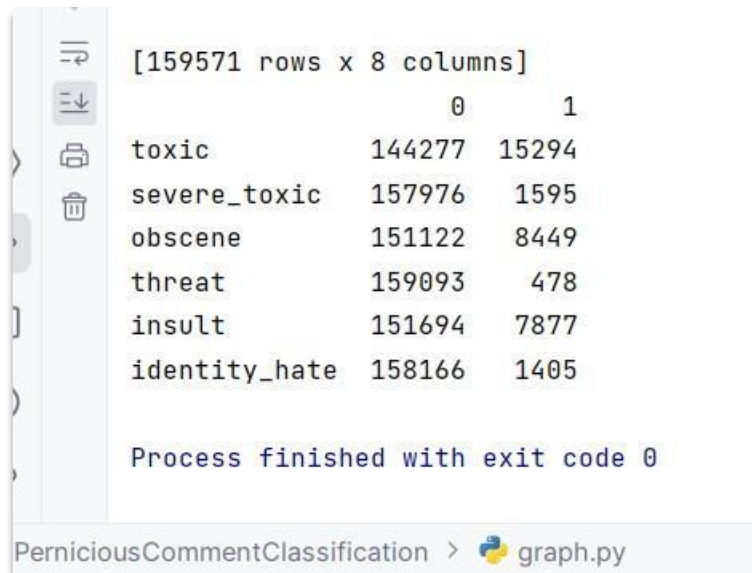
8. LIST OF FIGURES:

	A	B	C	D	E	F	G	H
43	00173958f46763a2	TFD	0	0	0	0	0	0
44	001810bf8c45bf5f	You are gay or antisemitian?	1	0	1	0	1	1
45	00190820581d90ce	FUCK YOUR FILTHY MOTHER IN THE ASS, DRY	1	0	1	0	1	0
46	001956c382006abd	I'm Sorry	1	0	0	0	0	0
47	001b2dd65d9d925c	I don't believe the Lisak criticism present	0	0	0	0	0	0
48	001c419c445b5a59	You had a point, and it's now ammended with	0	0	0	0	0	0
49	001c557175094f10	In other words, you're too lazy to actually poi	0	0	0	0	0	0
50	001cadfd324f8087	"	0	0	0	0	0	0
51	001d874a4d3e8813	":::Jmabel; in regards to predominant	0	0	0	0	0	0
52	001d8e7be417776a	"	0	0	0	0	0	0
53	001dc38a83d420cf	GET FUCKED UP. GET FUCKEED UP. GOT A [1	0	1	0	0	0
54	001e89eb3f0b0915	Are you threatening me for disputing neutralit	0	0	0	0	0	0
55	001ee16c46a99262	Thanks! Undeletion was more than I'd hoped	0	0	0	0	0	0
56	001ffdcc3e7fb49c	Awesome! Then I'll simply disregard your noti	0	0	0	0	0	0
57	0020e7119b96eeeb	Stupid peace of shit stop deleting my stuff ass	1	1	1	0	1	0
58	0020f96ed3b8c8b	=Tony Sidaway is obviously a fistfuckee. He lc	1	0	1	0	1	0
59	00218d74784ce50b	"	0	0	0	0	0	0
60	0021fe88bc4da3e6	My Band Page's deletion. You thought I was	1	0	1	0	0	0
61	002264ea4d5f2887	Why can't you believe how fat Artie is? Did	1	0	0	0	0	0
62	00229d44f41f3acb	Locking this page would also violate WP:NEW	0	0	0	0	0	0
63	0022cf8467ebc9fd	A Bisexual, like a homosexual or a heterosexu	0	0	0	0	0	0
64	0023daf96917e0d0	REDIRECT Talk:Frank Herbert Mason	0	0	0	0	0	0
65	002746baedcfff10	"	0	0	0	0	0	0
66	00280c0d0652b366	"	0	0	0	0	0	0
67	0028d62e8a5629aa	All of my edits are good. Cunts like you who r	1	0	1	0	1	0
68	00290e2a171dd073	"	0	0	0	0	0	0

[Figure 5: Data set]



[Figure 6: comments classified into multiple labels]



```
[159571 rows x 8 columns]
```

	0	1
toxic	144277	15294
severe_toxic	157976	1595
obscene	151122	8449
threat	159093	478
insult	151694	7877
identity_hate	158166	1405

Process finished with exit code 0

PerniciousCommentClassification > graph.py

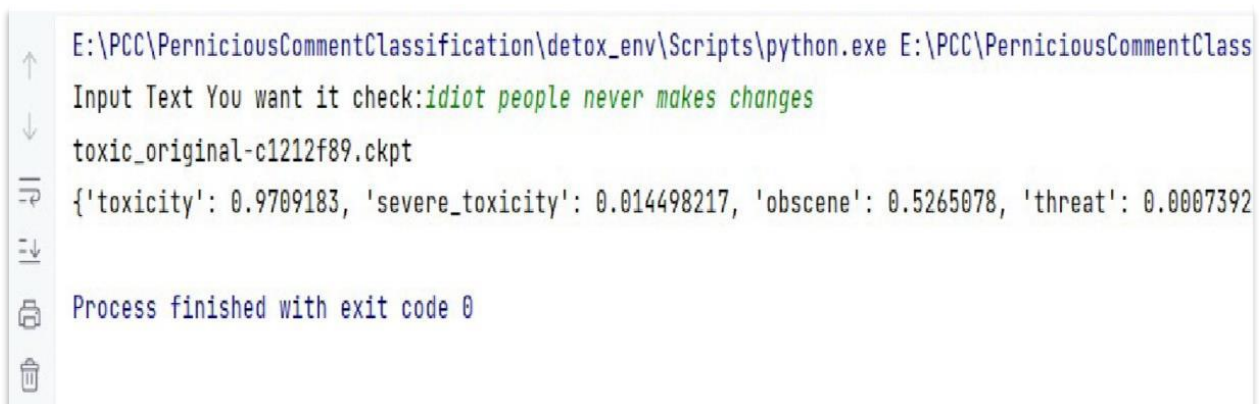
[Figure 7: Number of comments in each label]

$$\frac{\text{Number of correctly classified comments}}{\text{Total number of comments}} = 0.8999307931367684$$

[Figure 8: Correctly classified comments ration]

$$\frac{\text{Number of incorrectly classified comments}}{\text{Total number of comments}} = 0.0184116$$

[Figure 9: Incorrectly classified comments ration]



```
E:\PCC\PerniciousCommentClassification\detox_env\Scripts\python.exe E:\PCC\PerniciousCommentClass
Input Text You want it check: idiot people never makes changes
toxic_original-c1212f89.ckpt
{'toxicity': 0.9709183, 'severe_toxicity': 0.014498217, 'obscene': 0.5265078, 'threat': 0.0007392}
Process finished with exit code 0
```

[Figure 10: Input comment and its result]

	input (1)	toxicity (2)	severe_toxicity (3)	obscene (4)	threat (5)	insult (6)	identity_attack (7)
1	input	toxicity	severe_toxicity	obscene	threat	insult	identity_attack
2	idiot people go off	0.9893934726715088	0.030797988176345825	0.6318760514259338	0.002552563091740012	0.9204726815223694	0.009602933190762997
3	hello abc	0.0007955654291436	0.0001088420904125	0.0001796125579858	0.000109516622615	0.0001775218988768	0.0001389183889841
4	go to hell you bastered	0.990332305431366	0.0578935071825981	0.5667542815208435	0.0375420153141021	0.874151349067688	0.0119125293567776
5	hey stupid person	0.988394856452942	0.0376011095941066	0.7366859912872314	0.0011976503301411	0.9449677467346193	0.0090697715058922
6	Mr. ABC is bull sit	0.8008994460105896	0.0040402584709227	0.4288942217826843	0.0007218722603283	0.148760050535202	0.0040329699404537
7	jigar's address is bull sit to remember	0.2742264866828918	0.0019261323614045	0.1570914834737777	0.0008390200673602	0.0489993579685688	0.003947667311877
8	your address is bull sit	0.8311111330986023	0.0066777057945728	0.5952369570732117	0.0014680721797049	0.2504874765872955	0.0048744669184088
9	sharukh Khan is bludy bitch person	0.9887099862098694	0.1064681112766265	0.9443145990371704	0.0014259211020544	0.9301167726516724	0.0320105068385601
10	hello	0.0009314669296145	0.0001051655926858	0.0001761744206305	0.0001067618650267	0.0001848346146289	0.0001398382155457
11	try it	0.02435525265026	0.0001134612466557	0.0006587671232409	0.0003187741967849	0.0007283838349394	0.0003028775390703
12	helhjhkh	0.0188045855611562	0.0001472783333156	0.0014893960906192	0.0001400625333189	0.0006306421128101	0.000245269620791

[Figure 11: Classification of new input comments]

9. CONCLUSION:

This study focused on the development and evaluation of a machine learning-based system for pernicious comment classification. The objective was to design a model capable of automatically detecting and classifying offensive comments in online platforms, aiding to promote safer online environments.

Throughout the research process, we have observed many findings. First, the utilization of BERT, a state-of-the-art transformer-based model, proved to be highly effective in capturing intricate contextual information within the comments. BERT's pre-training on large-scale text enabled the model to grasp complex linguistic and improve the accuracy of classification.

The implementation of BERT with PyTorch, a popular Machine Learning framework, provided a powerful and flexible environment for training and fine-tuning the model. The flexibility of PyTorch allowed for seamless integration with other libraries, such as NLTK. These preprocessing steps were crucial in enhancing the model's understanding of the comment, and improving classification performance.

The training process involved BERT with the pernicious comment dataset allowed the model to learn from contextual patterns related to harmful language usage. The evaluation of the proposed classification model was conducted using standard evaluation metrics such as accuracy, precision, recall, and F1 score. Additionally, techniques like cross-validation and confusion matrix analysis provided the model's performance across different categories of pernicious comments. The results demonstrated a high level of accuracy and effectiveness in identifying harmful or offensive comments, showcasing the potential for real-world application.

In conclusion, this research presents a comprehensive of machine learning techniques for pernicious comment classification, specifically saying BERT, PyTorch, and NLTK. In addressing the challenges of identifying harmful comments. The deployment of such a system should consider potential and carries ethical implications.

Additionally, future research could focus on exploring ways to improve the model's performance on specific subcategories of harmful language and expanding the model's capability to detect and classify new types of pernicious comments.

In summary, the developed machine learning-based pernicious comment classification system utilizing BERT, PyTorch, and NLTK showcases the potential for creating safer online spaces. It is hoped that this research will inspire further advancements in the field, leading to the development of robust and scalable solutions to tackle the challenges associated with harmful online contents

References

- [1]. Giannis Haralabopoulos, Ioannis Anagnostopoulos, Derek Mc. Auley. Ensemble deep learning for multilabel binary classification of user generated content. April 2020 publication : MDPI
- [2]. Eloi Brassard-Gourdeau, Richard Khoury. Using sentiment information for preceptive detection of toxic comments in online conversations. June-2020 Publication: ArXiv
- [3]. Mukul Anand, Dr. R. Eswari. Classification of Abusive comments in Social Media using deep learning. 2019 publication : IEEE
- [4]. Sourabh raja murali, sanketh rangreji, Siddhanth vinay, Gowari Srinivasa. Automated NER, sentiment analysis and toxic comment classification for a goal-oriented chatbot. 2020 publication : IEEE
- [5]. Mladen Karan, Jan Snajder. Preemptive toxic language detection in Wikipedia comments using thread-level content. August 2019.
- [6]. Mai Ibrahim, Marwan Torki, and Nagwa EI-Makky. Imbalance toxic comments classification using Data Augmentation and deep learning. 2018 Publication: IEEE
- [7] H. M. Saleem, K. P. Dillon, S. Benesch, and D. Ruths, "A Web of Hate: Tracking Hateful Speech in Online Social Spaces," 2017, [Online]. Available: <http://arxiv.org/abs/1709.10159>.
- [8] M. Duggan, "Online harassment 2017," Pew Res., pp. 1–85, 2017, doi: 10.2419/4372.
- [9] M. A. Walker, P. Anand, J. E. F. Tree, R. Abbot, and J. King, "A corpus for research on deliberation and debate," Proc. 8th Int. Conf. Lang. Resour. Eval. Lr. 2012, pp. 812–817, 2012.
- [10] J. Cheng, C. Danescu-Niculescu-Mizil, and J. Leskovec, "Antisocial behavior in online discussion communities," Proc. 9th Int. Conf. Web Soc. Media, ICWSM 2015, pp. 61–70, 2015.
- [11] B. Matthew et al., "Thou shalt not hate: Countering online hate speech," Proc. 13th Int. Conf. Web Soc. Media, ICWSM 2019, no. August, pp. 369–380, 2019.
- [12] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, "Abusive language detection in online user content," 25th Int. World Wide Web Conf. WWW 2016, pp. 145–153, 2016, doi: 10.1145/2872427.2883062.
- [13] E. K. Ikonomakis, S. Kotsiantis, and V. Tampakas, "Text Classification Using Machine Learning Techniques," no. August, 2005.
- [14] M. R. Murthy, J. V. Murthy, and P. Reddy P.V.G.D, "Text Document Classification based on Least Square Support Vector Machines with Singular Value Decomposition," Int. J. Comput. Appl., vol. 27, no. 7, pp. 21–26, 2011, doi: 10.5120/3312-4540.
- [15] E. Wulczyn, N. Thain, and L. Dixon, "Ex machina: Personal attacks seen at scale," 26th Int. World Wide Web Conf. WWW 2017, pp. 1391–1399, 2017, doi: 10.1145/3038912.3052591.
- [16] H. Hosseini, S. Kannan, B. Zhang, and R. Poojendran, "Deceiving Google's Perspective API Built for Detecting Toxic Comments," 2017, [Online]. Available: <http://arxiv.org/abs/1702.08138>.
- [17] Y. Kim, "Convolutional neural networks for sentence classification," EMNLP 2014 - 2014 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf., pp. 1746–1751, 2014, doi: 10.3115/v1/d14-1181.
- [18] R. Johnson and T. Zhang, "Effective use of word order for text categorization with convolutional neural networks," NAACL HLT 2015 - 2015 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. Proc. Conf., no. 2011, pp. 103–112, 2015, doi: 10.3115/v1/n15-1011.
- [19] Y. Chen and S. Zhu, "Detecting Offensive Language in Social Media to Protect Adolescents," [Online]. Available: <http://www.cse.psu.edu/~sxz16/papers/SocialCom2012.pdf>.
- [20] A. L. Sulke and A. S. Varude, "Classification of Online Pernicious Comments using Machine Learning," no. October, 2019.
- [21] N. Chakrabarty, "A Machine Learning Approach to Comment Toxicity Classification," Adv. Intell. Syst. Comput., vol. 999, pp. 183–193, 2020, doi: 10.1007/978-981-13-9042-5_16.