

(CEP)
TOXIC COMMENT CLASSIFICATION AND
DETECTION



ML

Name :Muhammad Ibrahim Shah

CMS ID :56946

Name :Mudasir Irshad

CMS:55945

Name:Muhammad Haris

CMS:56639

Department:Software Engineering(SE5)

SUBMITTED TO:
MAM SANIYA ASHRAF

Department of Software Engineering,
Faculty of Information & Communication Technology, BUITEMS, Que




COMPLEX ENGINEERING PROBLEM

Course Code	C	E	3	0	2	Course Title	Machine learning
--------------------	---	---	---	---	---	---------------------	------------------

Course Learning Outcomes

Course Learning Outcomes (CLOs)				
S#	CLO	Domain	Taxonomy level	PLO
1	<i>Discuss</i> how web standards impact software development and the constraints that the web puts on developers.	Cognitive	1	1
2	<i>Design</i> and implement a simple web application.	Cognitive	4	3
3	<i>Analyze</i> an existing web application against a current web standard.	Cognitive	4	4

Table of Contents

Abstract:.....	4
Introduction:	4
Literature review Related Work:.....	5
Datasets and Tasks:.....	5
3.1kaggle dataset:	6
	
Kaggle dataset:.....	7
Common Challenges:	7
Methods and Ensemble;	7
4.1: Neural Networks:	7
4.2 Convolutional Neural Networks:.....	8
Result and discussion:.....	8
Future work:.....	14
Conclusion:.....	15
References:.....	15

Abstract:

Toxic comment classification has become an active research field with many recently proposed approaches. However, while these approaches address some of the task's challenges others still remain unsolved and directions for further research are needed. To this end, we compare different deep learning and shallow approaches on a new, large comment dataset and propose an ensemble that outperforms all individual models. Further, we validate our findings on a second dataset. The results of the ensemble enable us to perform an extensive error analysis, which reveals open challenges for state-of-the-art methods and directions towards pending future research. These challenges include missing paradigmatic context and inconsistent dataset labels.

Introduction:

Keeping online conversations constructive and inclusive is a crucial task for platform providers. Automatic classification of toxic comments, such as hate speech, threats, and insults, can help in keeping discussions fruitful. In addition, new regulations in certain European countries have been established enforcing to delete illegal content in less than 72 hours.¹

Active research on the topic deals with common challenges of natural language processing, such as long-range dependencies or misspelled and idiosyncratic words. Proposed solutions include bidirectional recurrent neural networks with attention (Pavlopoulos et al., 2017) and the use of pretrained word embeddings (Badjatiya et al., 2017). However, many classifiers suffer from insufficient variance in methods and training data and therefore often tend to fail on the long tail of real world data (Zhang and Luo, 2018). For future research, it is

are already addressed by state-of-the-art classifiers and for which challenges current solutions are still error-prone.

We take two datasets into account to investigate these errors: comments on Wikipedia talk pages presented by Google Jigsaw during Kaggle's Toxic Comment Classification Challenge² and a Kaggle Dataset by Davidson et al. (2017). These sets include common difficulties in datasets for the task: They are labeled based on different definitions; they include diverse language from user comments and Tweets; and they present a multi-class and a multi-label classification task respectively.

On these datasets we propose an ensemble of state-of-the-art classifiers. By analysing false negatives and false positives of the ensemble we get insights about open challenges that all of the approaches share. Therefore, our main contributions are:

- 1) We are the first to apply and compare a range of strong classifiers to a new public multilabel dataset of more than 200,000 user comments. Each classifier, such as Logistic Regression, bidirectional RNN and CNN, is meant to tackle specific challenges for text classification. We apply the same classifiers to a dataset of Tweets to validate our results on a different domain.
- 2) We apply two different pretrained word embeddings for the domain of user comments and Tweets to compensate errors such as idiosyncratic and misspelled words.

We compare the classifiers' predictions and show that they make different errors as measured by Pearson correlation coefficients and F1measures. Based on this, we create an ensemble that improves macro-averaged F1-measure especially on sparse classes and data common errors of all current approaches. We propose directions for future work based on these unsolved challenges

Literature review Related Work:

Task definitions. Toxic comment classification is not clearly distinguishable from its related tasks. Besides looking at toxicity of online comments (Wulczyn et al., 2017; Georgakopoulos et al., 2018), related research includes the investigation of hate speech (Badjatiya et al., 2017; Burnap and Williams, 2016; Davidson et al., 2017; Gamback and Sikdar, 2017; Njagi et al., 2015;

Schmidt and Wiegand, 2017; Vigna et al., 2017; Warner and Hirschberg, 2012), online harassment (Yin and Davison, 2009; Golbeck et al., 2017), abusive language (Mehdad and Tetreault, 2016; Park and Fung, 2017), cyberbullying (Dadvar et al., 2013; Dinakar et al., 2012; Hee et al., 2015; Zhong et al., 2016) and offensive language (Chen et al., 2012; Xiang et al., 2012). Each field uses different definitions for their classification, still similar methods can often be applied to different tasks. In our work we focus on toxic comment detection and show that the same method can effectively be applied to a hate speech detection task.

Requirement gathering Datasets and Tasks:

The task of toxic comment classification lacks a consistently labeled standard dataset for comparative evaluation (Schmidt and Wiegand, 2017). While there are a number of annotated public datasets in adjacent fields, such as hate speech (Ross et al., 2016; Gao and Huang, 2017), racism/sexism (Waseem, 2016; Waseem and Hovy, 2016) or harassment (Golbeck et al., 2017)

detection, most of them follow different definitions for labeling and therefore often constitute different problems.

```
[6]: df = pd.read_csv(os.path.join('jigsaw-toxic-comment-classification-challenge', 'train.csv', 'train.csv'))

[7]: df.head()
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

```
[8]: df.tail()
```

```
[8]:
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
159566	f1e987279560d7ff	"And for the second time of asking, when ...	0	0	0	0	0	0
159567	f1ee4adeec384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0
159568	f1ee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0
159569	f1f125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0
159570	f1f46fc426af1f9a	\n\nAnd ... I really don't think you understand...	0	0	0	0	0	0

3.1kaggle dataset:

We analyse a dataset published by kaggle Google Jigsaw in December 2017 over the course of the ‘Toxic Comment Classification Challenge’ on Kaggle. It includes 223,549 annotated user comments collected from Wikipedia talk pages and is the largest publicly available for the task. These comments were annotated by human raters with the six labels ‘toxic’, ‘severe toxic’, ‘insult’, ‘threat’, ‘obscene’ and ‘identity hate’. Comments can be associated with multiple classes at once, which frames the task as a multi-label classification problem. Jigsaw has not published official definitions for the six classes. But they do state that they defined a toxic comment as “a rude, disrespectful, or unreasonable comment that is likely to make you leave a discussion”.¹

The dataset features an unbalanced class distribution, shown in Table 1. 201,081 samples fall under the majority ‘clear’ class matching none of the six categories, whereas 22,468 samples

belong to at least one of the other classes. While the ‘toxic’ class includes 9.6% of the samples, only 0.3% are labeled as ‘threat’, marking the smallest class.

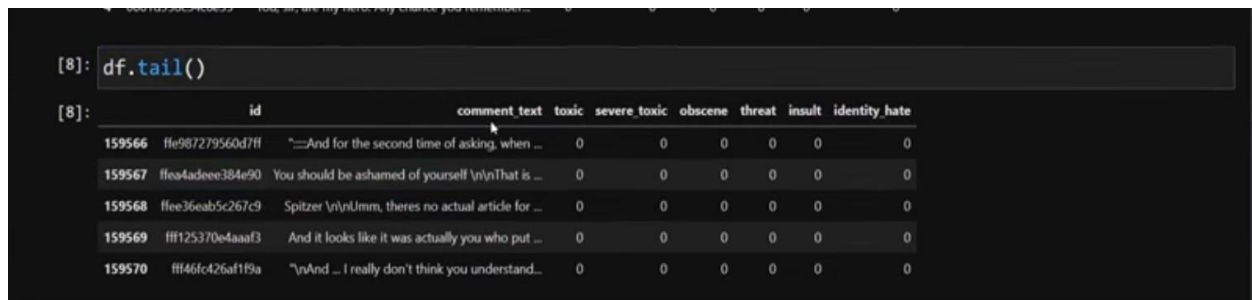
Comments were collected from the English Wikipedia and are mostly written in English with some outliers.

```
[6]: df = pd.read_csv(os.path.join('jigsaw-toxic-comment-classification-challenge', 'train.csv', 'train.csv'))
```

```
[7]: df.head()
```

```
[7]:
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9c6b60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	\n\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0



```
[8]: df.tail()
```

	id	comment text	toxic	severe toxic	obscene	threat	insult	identity hate
159566	f1e987279560d7ff	"...And for the second time of asking, when ...	0	0	0	0	0	0
159567	f1ee4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0
159568	f1ee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0
159569	f1f125370e4aaa3	And it looks like it was actually you who put ...	0	0	0	0	0	0
159570	f1f46fc426af19a	\n\nAnd ... I really don't think you understand...	0	0	0	0	0	0

Kaggle dataset:

Additionally we investigate a dataset introduced by Davidson et al. (2017). It contains 24,783 comments fetched using the Kaggle API and annotated by CrowdFlower workers with the labels ‘hate speech’, ‘offensive but not hate speech’ and ‘neither offensive nor hate speech’. Table 2 shows the class distribution. We observe a strong bias towards the offensive class making up 77.4% of the comments caused by sampling tweets by seed keywords from Hatebase.org. We choose this dataset to show that our method is also applicable to multi-class problems and works with Tweets, which usually have a different structure than other online user comments due to character limitation.

Common Challenges:

We observe three common challenges for Natural Language Processing in both datasets:

Out-of-vocabulary words. A common problem for the task is the occurrence of words that are not present in the training data. These words include slang or misspellings, but also intentionally obfuscated content.

Long-Range Dependencies. The toxicity of a comment often depends on expressions made in early parts of the comment. This is especially problematic for longer comments (>50 words) where the influence of earlier parts on the result can vanish.

Multi-word phrases. We see many occurrences of multi-word phrases in both datasets. Our algorithms can detect their toxicity only if they can recognize multiple words as a single (typical) hateful phrase.

Functions Methods and Ensemble;

In this section we apply baseline methods for the above mentioned common challenges. Further, we propose our ensemble learning architecture. Its goal is to minimize errors by detecting optimal methods for a given comment.

4.1: Neural Networks:

Neural Networks (RNNs) interpret a document as a sequence of words or character ngrams. We use four different RNN approaches: An LSTM (Long-Short-Term-Memory Network), a bidirectional LSTM, a bidirectional GRU (Gated Recurrent Unit) architecture and a bidirectional GRU with an additional attention layer.

LSTM. Our LSTM model takes a sequence of words as input. An embedding layer transforms one-hot-encoded words to dense vector representations and a spatial dropout, which randomly masks 10% of the input words, makes the network more robust. To process the sequence of word embeddings, we use an LSTM layer with 128 units, followed by a dropout of 10%. Finally, a dense layer with a sigmoid activation

makes the prediction for the multi-label classification and a dense layer with softmax activation makes the prediction for the multi-class classification.

Bidirectional LSTM and GRU. Bidirectional RNNs can compensate certain errors on long range dependencies. In contrast to the standard LSTM model, the bidirectional LSTM model uses two LSTM layers that process the input sequence in opposite directions. Thereby, the input sequence is processed with correct and reverse order of words. The outputs of these two layers are averaged. Similarly, we use a bidirectional GRU model, which consists of two stacked GRU layers. We use layers with 64 units. All other parts of the neural network are inherited from our standard LSTM model. As a result, this network can recognize signals on longer sentences where neurons representing words further apart from each other in the LSTM sequence will 'fire' more likely together.

4.2 Convolutional Neural Networks:

Convolutional Neural Networks (CNNs) are recently becoming more popular for text classification tasks. By intuition they can detect specific combinations of features, while RNNs can extract orderly information (Zhang and Luo, 2018). On character level, CNNs can deal with obfuscation of words. For our model we choose an architecture comparable to the approach of Kim (2014).

Result and discussion:

As shown in the results below our ensemble outperforms the strongest individual method on the Wikipedia and kaggle dataset by approximately one percent F1-measure. We see that the difference in F1-measure between the best individual classifier and the ensemble is higher on the Wikipedia dataset as on the Twitter dataset. This finding is accompanied by the results in Table 4 which show that most classifier combinations present a high correlation on the Twitter dataset and are therefore less effective on the ensemble. An explanation for this effect is that the text sequences within the Twitter set show less variance than the ones in the Wikipedia dataset. This can be reasoned from 1) their sampling strategy based on a list of terms, 2) the smaller size of the dataset and 3) less disparity within the three defined classes than in the six from the Wikipedia dataset. With less variant data one selected classifier for a type of text can be sufficient.

0. Install Dependencies and Bring in Data

```
[1]: !pip install tensorflow tensorflow-gpu pandas matplotlib sklearn

Requirement already satisfied: tensorflow in d:\youtube\12-08-2021 - toxic comment detector\toxic\lib\site-packages (2.8.0)
Requirement already satisfied: tensorflow-gpu in d:\youtube\12-08-2021 - toxic comment detector\toxic\lib\site-packages (2.8.0)
Requirement already satisfied: pandas in d:\youtube\12-08-2021 - toxic comment detector\toxic\lib\site-packages (1.4.2)
Requirement already satisfied: matplotlib in d:\youtube\12-08-2021 - toxic comment detector\toxic\lib\site-packages (3.5.1)
Requirement already satisfied: sklearn in d:\youtube\12-08-2021 - toxic comment detector\toxic\lib\site-packages (0.0)
Requirement already satisfied: wrapt>=1.11.0 in d:\youtube\12-08-2021 - toxic comment detector\toxic\lib\site-packages (from tensor
flow) (1.14.0)

[3]: import os
import pandas as pd
import tensorflow as tf
import numpy as np

[5]: os.path.join('jigsaw-toxic-comment-classification-challenge', 'train.csv', 'train.csv')

[5]: 'jigsaw-toxic-comment-classification-challenge\\train.csv\\train.csv'
```



```
[6]: df = pd.read_csv(os.path.join('jigsaw-toxic-comment-classification-challenge', 'train.csv'), 'train.csv')
[7]: df.head()
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

```
[8]: df.tail()
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
159566	f1e987279560d7ff	"And for the second time of asking, when ...	0	0	0	0	0	0
159567	f1ea4adeec384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0
159568	f1ee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0
159569	f1f125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0
159570	f1f46fc426af1f9a	\nAnd ... I really don't think you understand...	0	0	0	0	0	0

For Preprocessing:

We as a group use vectorization which give each word in the dictionary a specific token number and then act according to that.

1. Preprocess

```
[24]: !pip list
...
[23]: from tensorflow.keras.layers import TextVectorization
[25]: TextVectorization??
```

Init signature:

```
TextVectorization(
    max_tokens=None,
    standardize='lower_and_strip_punctuation',
    split='whitespace',
    ngrams=None,
    output_mode='int',
    output_sequence_length=None,
    pad_to_max_tokens=False,
    vocabulary=None,
    idf_weights=None,
    sparse=False,
    ragged=False,
    **kwargs,
```

```
[ ]: X = df['comment_text']
y = df[df.columns[2:]].values

[26]: df.columns

[26]: Index(['id', 'comment_text', 'toxic', 'severe_toxic', 'obscene', 'threat',
          'insult', 'identity_hate'],
          dtype='object')

[27]: df['comment_text']

[27]: 0      Explanation\nWhy the edits made under my usern...
1      D'aww! He matches this background colour I'm s...
2      Hey man, I'm really not trying to edit war. It...
3      "\nMore\nI can't make any real suggestions on ...
4      You, sir, are my hero. Any chance you remember...
...
159566  ":::::And for the second time of asking, when ...
159567  You should be ashamed of yourself \n\nThat is ...
```

Here there are some features and comments .

```
[30]: df[df.columns[2:]].values

[30]: array([[0, 0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0, 0],
          ...,
          [0, 0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0, 0]], dtype=int64)
```

```
[36]: MAX_FEATURES = 200000 # number of words in the vocab

[37]: vectorizer = TextVectorization(max_tokens=MAX_FEATURES,
                                     output_sequence_length=1800,
                                     output_mode='int')

[41]: vectorizer.adapt(X.values)

[43]: vectorizer('Hello world, life is great')

[43]: <tf.Tensor: shape=(1800,), dtype=int64, numpy=array([[286, 261, 305, ..., 0, 0, 0], dtype=int64)>
```

```
[46]: vectorized_text = vectorizer(X.values)

[48]: len(X)

[48]: 159571

[47]: vectorized_text

[47]: <tf.Tensor: shape=(159571, 1800), dtype=int64, numpy=
array([[ 643,    76,     2, ...,    0,     0,     0],
       [   1,    54,  2506, ...,    0,     0,     0],
       [  425,   440,    70, ...,    0,     0,     0],
       ...,
       [32141,  7329,   383, ...,    0,     0,     0],
       [    5,    12,   533, ...,    0,     0,     0],
       [    5,     8,   130, ...,    0,     0,     0]], dtype=int64)>
```

```
[49]: #MCSHBAP - map, chache, shuffle, batch, prefetch from_tensor_slices, list_
dataset = tf.data.Dataset.from_tensor_slices((vectorized_text, y))
dataset = dataset.cache()
dataset = dataset.shuffle(160000)
dataset = dataset.batch(16)
dataset = dataset.prefetch(8) # helps bottlenecks

[50]: dataset.as_numpy_iterator().next()

[50]: (array([[116673,     2,   7183, ...,     0,     0,     0],
              [   76,     9,   5807, ...,     0,     0,     0],
              [   46,    33,    15, ...,     0,     0,     0],
              ...,
              [   168,     4,   7427, ...,     0,     0,     0],
              [   31, 31557,  2825, ...,     0,     0,     0],
              [    8,    47,    52, ...,     0,     0,     0]], dtype=int64),
 array([[0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0]]))
```

Trainning our dataset

```
[59]: int(len(dataset)*.7)

[59]: 6981

[60]: train = dataset.take(int(len(dataset)*.7))
      val = dataset.skip(int(len(dataset)*.7)).take(int(len(dataset)*.2))
      test = dataset.skip(int(len(dataset)*.9)).take(int(len(dataset)*.1))

[64]: train_generator = train.as_numpy_iterator()

[70]: train_generator.next()

...
```

Now applying neural network:

```
[ ]: model = Sequential()
# Create the embedding Layer
model.add(Embedding(MAX_FEATURES+1, 32))
# Bidirectional LSTM Layer
model.add(Bidirectional(LSTM(32, activation='tanh'))))
# Feature extractor Fully connected layers
model.add(Dense(128, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
# Final Layer
model.add(Dense(6, activation='sigmoid'))
...
```

```
[78]: model.compile(loss='BinaryCrossentropy', optimizer='Adam')
```

```
[79]: model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, None, 32)	6400032
bidirectional (Bidirectional)	(None, 64)	16640
dense (Dense)	(None, 128)	8320
dense_1 (Dense)	(None, 256)	33024
dense_2 (Dense)	(None, 128)	32896
dense_3 (Dense)	(None, 6)	774

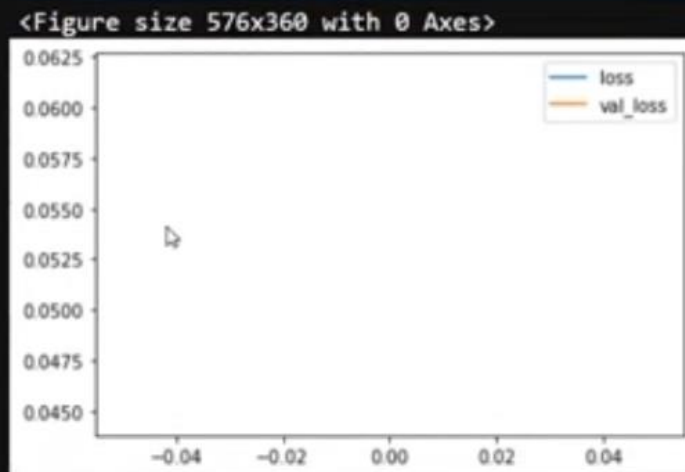
```
=====
Total params: 6,491,686
Trainable params: 6,491,686
Non-trainable params: 0
```

```
[87]: history = model.fit(train, epochs=1, validation_data=val)
6981/6981 [=====] - 1341s 192ms/step - loss: 0.0618 - val_loss: 0.0446
```

Now here we divide our dataset into batches and apply one epochs for training.

```
[88]: from matplotlib import pyplot as plt
```

```
[90]: plt.figure(figsize=(8,5))  
pd.DataFrame(history.history).plot()  
plt.show()
```



Now predicting if it is predicting well.

```
[95]: model.predict(np.expand_dims(input_text,0))
```

```
[95]: array([[0.99490464, 0.22444303, 0.9404793 , 0.04423362, 0.816273 ,  
0.18399473]], dtype=float32)
```

3. Make Predictions

```
[ ]: batch = test.as_numpy_iterator().next()
```

```
[91]: input_text = vectorizer('You freaking suck!')
```

```
[98]: df.columns
```

```
[98]: Index(['id', 'comment_text', 'toxic', 'severe_toxic', 'obscene', 'threat',  
'insult', 'identity_hate'],  
dtype='object')
```

```
[95]: model.predict(np.expand_dims(input_text,0))
```

```
[95]: array([[0.99490464, 0.22444303, 0.9404793 , 0.04423362, 0.816273 ,  
0.18399473]], dtype=float32)
```

4. Evaluate Model

```
[113]: from tensorflow.keras.metrics import Precision, Recall, CategoricalAccuracy
```

```
[ ]: pre = Precision()  
re = Recall()  
acc = CategoricalAccuracy()
```



```
[ ]: for batch in test.as_numpy_iterator():
    # Unpack the batch
    X_true, y_true = batch
    # Make a prediction
    yhat = model.predict(X_true)

    # Flatten the predictions
    y_true = y_true.flatten()
    yhat = yhat.flatten()

    pre.update_state(y_true, yhat)
    re.update_state(y_true, yhat)
    acc.update_state(y_true, yhat)
```

```
print(f'Precision: {pre.result().numpy()}, Recall:{re.result().numpy()}, Accuracy:{acc.result().numpy()}')
Precision: 0.81080162525177, Recall:0.6833667159080505, Accuracy:0.47442325949668884
```

```
print("Accuracy: {:.3f}".format(accuracy))
print("Precision: {:.3f}".format(precision))
print("Recall: {:.3f}".format(recall))
print("F1 score: {:.3f}".format(f1))
```

```
Accuracy: 0.999
Precision: 0.870
Recall: 0.709
F1 score: 0.781
```

Scalability & Future work:

To advance the field of toxic comment classification, researchers and practitioners can explore several avenues for future work. Here are some potential directions:

Fine-grained Classification:

Current models often classify comments into binary categories (toxic or non-toxic). Future work could focus on more fine-grained classifications, such as identifying specific types of toxicity (e.g., hate speech, harassment, misinformation) to provide more nuanced insights.

Cross-lingual and Multilingual Approaches:

Extend models to handle multiple languages and dialects to address the global nature of online communication. This can be especially important for platforms with diverse user bases.

Conclusion:

In this work we presented multiple approaches for toxic comment classification. We showed that the approaches make different errors and can be combined into an ensemble with improved F1- measure. The ensemble especially outperforms when there is high variance within the data and on classes with few examples. Some combinations such as shallow learners with deep neural networks are especially effective. Our error analysis on results of the ensemble identified difficult subtasks of toxic comment classification. We find that a large source of errors is the lack of consistent quality of labels. Additionally most of the unsolved challenges occur due to missing training data with highly idiosyncratic or rare vocabulary. Finally, we suggest further research in representing world knowledge with embeddings to improve distinction between paradigmatic contexts.

References:

- [1] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets.
- [2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.
- [3] Pete Burnap and Matthew L. Williams. 2015. Cyber hate speech on twitter : An application of machine classification and statistical modeling for policy and decision making. volume 7, pages 223–242.
- [4] Thomas Davidson, Dana Warmley, Michael W. Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *ICWSM*.
- [5] Spiros V. Georgakopoulos, Sotiris K. Tasoulis, Aristidis G. Vrahatis, and Vassilis P. Plagianakos. 2018. Convolutional neural networks for toxic comment classification. In *SETN*

Github link:

<https://github.com/IbrahimShah123/machine-learning-CEP>

Assessment of CEP (This Section is for instructor's use only)

Category	Very Poor	Poor	Fair	Good	Excellent	Obtained Mark
Percentage	[1-20%]	[21-40%]	[41-60%]	[61-80%]	[81-100%]	
Understanding of the given problem and its relationship with the given situation/ application/ scenario	<i>Unable to understand the given problem</i>	<i>Develops a partial understanding of the given problem</i>	<i>Developed understanding of the given problem, but unable to correlate with</i>	<i>Fully understand the problem but does not understand the implication and</i>	<i>Fully understand the problem and understand the implications and</i>	20% of the Total Assigned Marks
			<i>the assigned application</i>	<i>relationship with the application.</i>	<i>relationship with the application.</i>	
Assigned Marks	0.01-0.10	0.11-0.20	0.21-0.30	0.31-0.40	0.41-0.50	
Obtained Marks						/2
Selection of an appropriate method/ approach and its implementation to solve the given problem with respect to the application	<i>Wrong method selected to solve the problem</i>	<i>Selected an appropriate method but developed wrong approach towards the solution</i>	<i>Selected an appropriate method and approach but unable to implement</i>	<i>Selected correct method and approach also able to implement partially</i>	<i>Selected correct method and approach also able to implement and execute correctly</i>	60% of the Total Assigned Marks
Assigned Marks	0.01-0.20	0.21-0.40	0.41-0.60	0.61-0.80	0.81-1.00	
Obtained Marks						/6
Results of the proposed solution against the given problem	<i>Unable to achieve any results</i>	<i>Achieved some results but unable to solve the problem</i>	<i>Achieved partial results and provided a partial solution to the problem</i>	<i>Achieved Results and provided an acceptable solution to the given problem</i>	<i>Achieved correct results and solved the given problem</i>	20% of the Total Assigned Marks
Assigned Marks	0.01-0.10	0.11-0.20	0.21-0.30	0.31-0.40	0.41-0.50	
Obtained Marks						/2
Instructor's Signature		Date		Total Obtained Marks		

Instructor's Name and Signature	Mam Saniya Ashraf
--	--------------------------