UNIVERSITA' DEGLI STUDI DI MILANO BICOCCA

A.Y. 2024/2025
Master's Degree
Informatica

Toxic Comments Challenge

Advanced Machine Learning Project – February 2025

Group Partecipants:

Gargiulo Elio - 869184
Gervasi Alessandro - 866140
Piacente Cristian - 866020

Introduction

Context – The Challenge

This project follows the general aim of the Kaggle competition "Toxic Comments Challenge".

The challenge is to build a model that's capable of detecting different types of Wikipedia comments:

- Toxic
- Severe Toxic
- Obscene
- Threat
- Insult
- Identity Hate

Context – Overview

The focus of this project is therefore to:

- Analyze the Dataset
- Preprocess the Comments
- Build the features for the models
- Train the baseline models
- Train the regularized models
- Evaluate the models
- Test on custom comments and Conclusions

kaggle

Dataset – Train Set

The Dataset has been already split into train and test sets, as provided by Kaggle.

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my username Hardcore Metallica Fan were reverted? They weren't vandalisms, just closure on some GAs after I voted at New York Dolls FAC. And please don't remove the template from the talk page since I'm retired now.89.205.38.27	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC)	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It's just that this guy is constantly removing relevant information and talking to me through edits instead of my talk page. He seems to care more about the formatting than the actual info.	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on improvement - I wondered if the section statistics should be later on, or a subsection of ""types of accidents"" -I think the references may need tidying so that they are all in the exact same format ie date format etc. I can do that later on, if no-one else does first - if you have any preferences for formatting style on references or want to do it yourself please let me know.\n\nThere appears to be a backlog on articles for review so I guess there may be a delay until a reviewer turns up. It's listed in the relevant form eg Wikipedia:Good_article_nominations#Transport "	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember what page that's on?	0	0	0	0	0	0

The training set records are composed of:

- id
- comment_text
- The six toxic labels

Each target label is set to **1** if the comment belongs to the corresponding toxic category and **0** if it does not.

It is a multi-binary classification task.

Dataset – Test Set

The test set contains the same format as the training set, except that the labels are stored separately in another CSV file, ready to be merged.

	id	comment_text
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll ever be whats up with you and hating you sad mofuckas...i should bitch slap ur pethedic white faces and get you to kiss my ass you guys sicken me. Ja rule is about pride in da music man. dont diss that shit on him. and nothin is wrong bein like tupac he was a brother too...fuckin white boys get things right next time.,
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is, IMO.
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lapland — / "
3	00017563c3f7919a	:If you have a look back at the source, the information I updated was the correct form. I can only guess the source hadn't updated. I shall update the information once again but thank you for your message.
4	00017695ad8997eb	I don't anonymously edit articles at all.

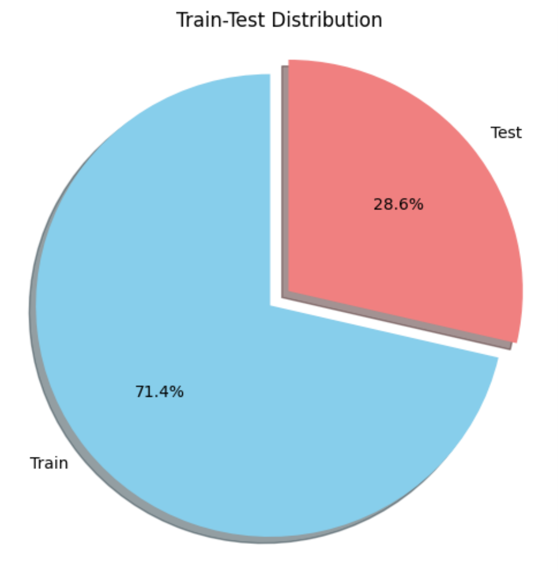
	id	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	00001cee341fdb12	-1	-1	-1	-1	-1	-1
1	0000247867823ef7	-1	-1	-1	-1	-1	-1
2	00013b17ad220c46	-1	-1	-1	-1	-1	-1
3	00017563c3f7919a	-1	-1	-1	-1	-1	-1
4	00017695ad8997eb	-1	-1	-1	-1	-1	-1

Before merging with the test set, some labels must be removed because they have a value of **-1**, which indicates that they are not used for evaluation.

Dataset – Train Test Distribution

After merging with the labels and removing those with a value of **-1**, the train and test sets exhibit the following Distribution:

- ~70% of Train Set (159571 records)
- ~30% of Test Set (63978 records)



Exploratory Data Analysis

Data Cleaning Checks

Before proceeding with the EDA, data cleaning checks were performed on the dataset to ensure that the data is not noisy or inconsistent:

- Check of Duplicated Rows
- Check of Missing Values

```
Number of duplicate rows in the training dataset: 0
  id  comment_text  toxic  severe_toxic  obscene  threat  insult  identity_hate
```

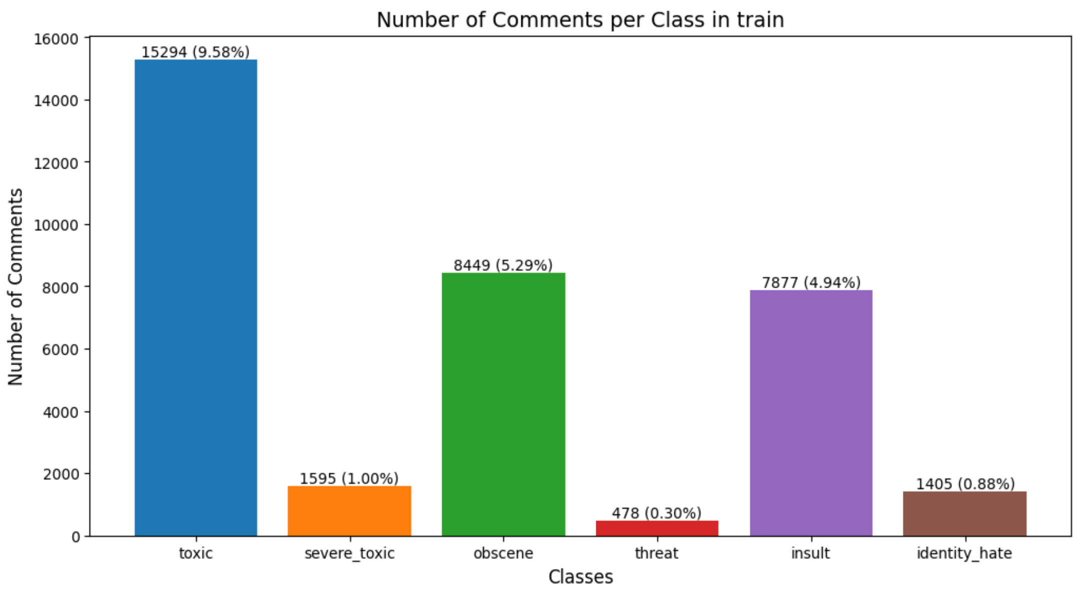
	Missing values sum in train
id	0
comment_text	0
toxic	0
severe_toxic	0
obscene	0
threat	0
insult	0
identity_hate	0

There are no duplicated rows or missing values.

Univariate Analysis – Class Distribution

Most of the comments are classified as **Toxic**, while **Threat** and **Identity Hate** are very **rare**.

From the percentages, it is clear that most of the dataset consists of **clean** comments.



Univariate Analysis – Word Cloud

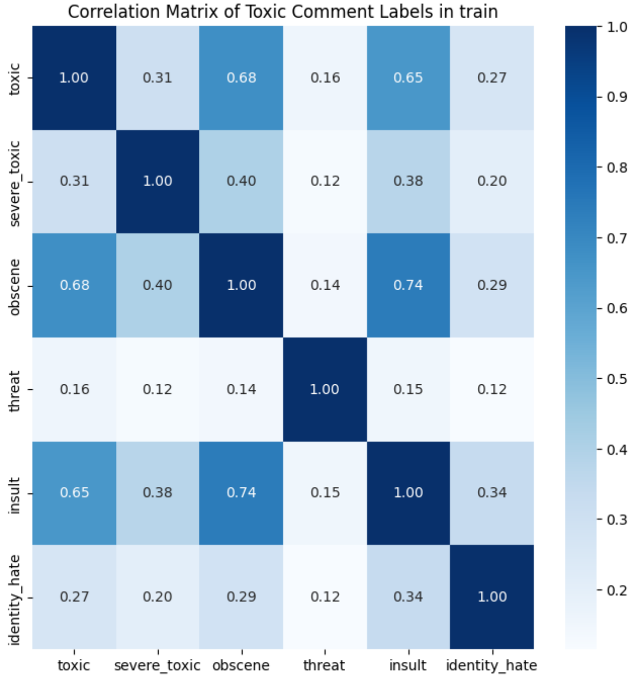
The word cloud shows the most frequent words used in the comments, some of them:

- Wikipedia – since it's where comments are from (Wikipedia talk pages)
- Edit, Article, Discussion – domain specific terms

Covariate Analysis – Correlation Matrix

Insult and **Obscene** have a high correlation (> **0.7**), meaning that usually a comment containing an insult is also obscene.

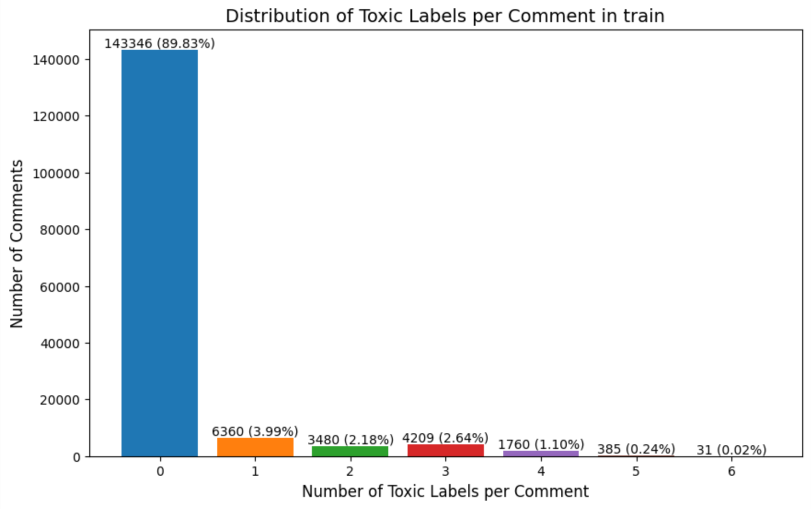
Also, **Toxic** is **correlated** with the labels discussed above.



Covariate Analysis – Clean & Toxic Labels

The plot shows the number of labels set to 1 in every single row:

- Most of them are confirmed to be **clean** comments (**0 labels** set to 1)
- Comments with multiple labels are rarer than the ones with a single label



Covariate Analysis – Toxic vs Others

The relationships between Toxic and the others were analyzed, here two comparisons:

- **Toxic vs Severe Toxic** – the second implies the first, there can't be a Severe Toxic comment that is not Toxic too
- **Toxic vs Insult** – most of the insults are also toxic, but not all

Toxic \ Severe	0	1
0	144277	0
1	13699	1595

Toxic \ Insult	0	1
0	143744	533
1	7950	7344

Text Processing

Text Processing - Steps

To prepare the textual data for analysis and later ML, the following steps were performed:

- **Tokenization** – every word is a token
  - TweetTokenizer is used, with parameters to
    - Convert to lowercase
    - Reduce repeated letters in words
    - Remove the citations (e.g., @User)
- **Normalization** – remove non-alphanumeric tokens
  - IP addresses, Links, emoticons are affected
- **Stop Words Removal**
- **Lemmatization** – keep the root form while making sure it's a valid English word

Text Processing - Example

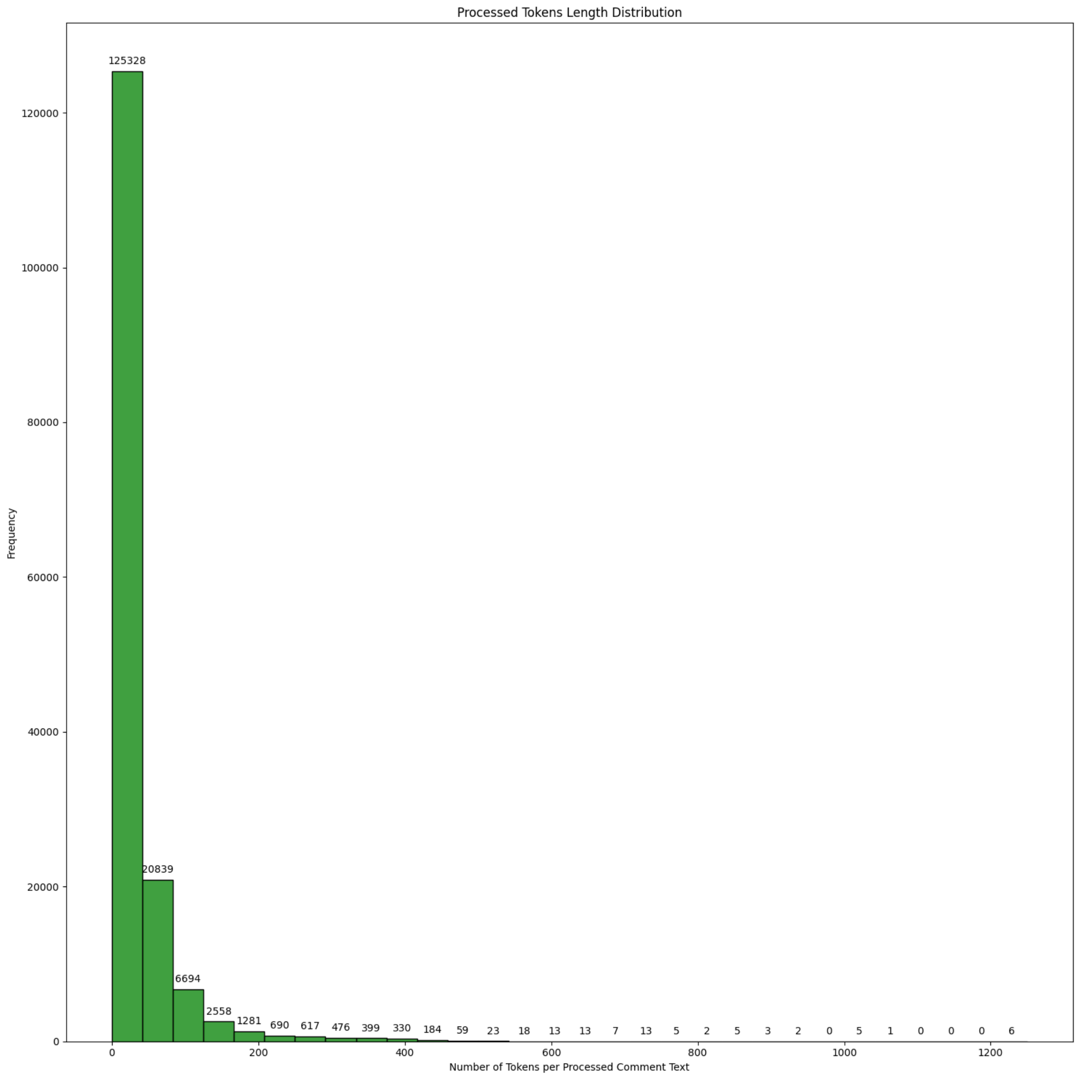
In this example there is an IP address in the comment.

It gets removed by the normalization step.

```
Original comment:
                                                                                      comment_text
27886   Plot\n\nWhat the heck is the movie about? What are the main story lines? The plot description in the article tells me nothing! The trailer for
        the movie makes little sense. I get the idea of past lives. But what are the stories in the movie? Marc S. Dania Fl. 206.192.35.125

dtype: object

Processed comment:
                                                                                      comment_text
27886   plot heck movie main story line plot description article tell nothing trailer movie make little sense idea past life story movie marc s dania fl
```



Processed Tokens Analysis

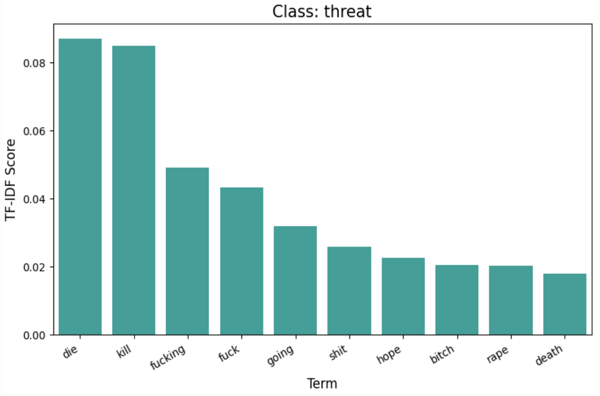
There are tons (~120k) of comments containing a low number of tokens, which are a very high percentage with respect to the training set (less than 160k).

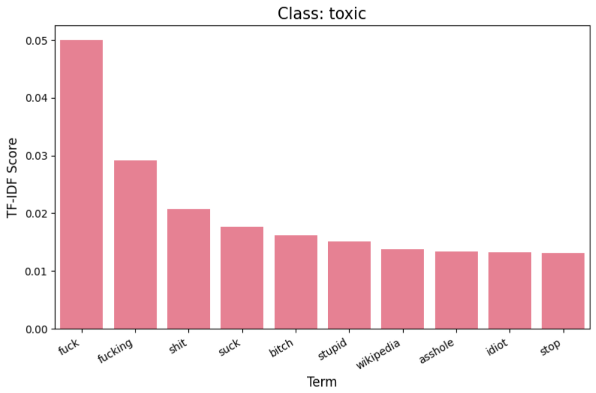
Processed Tokens Analysis

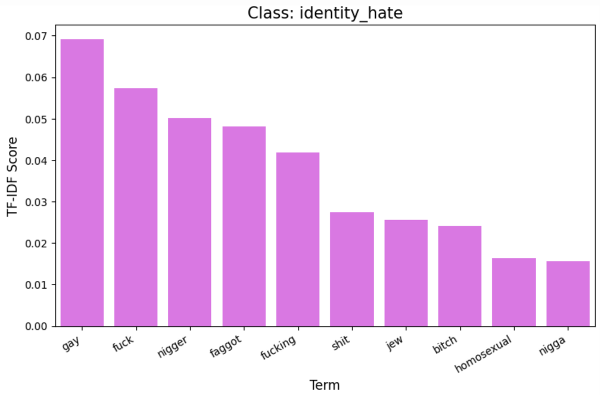
Using TF-IDF weights, the top 10 unigrams and bigrams for each label were plotted, to show the effects of Text Processing.
For brevity, only unigrams and 3 labels are shown.

Some examples are:

- In Threat, "kill"
- In Identity Hate, "homosexual"





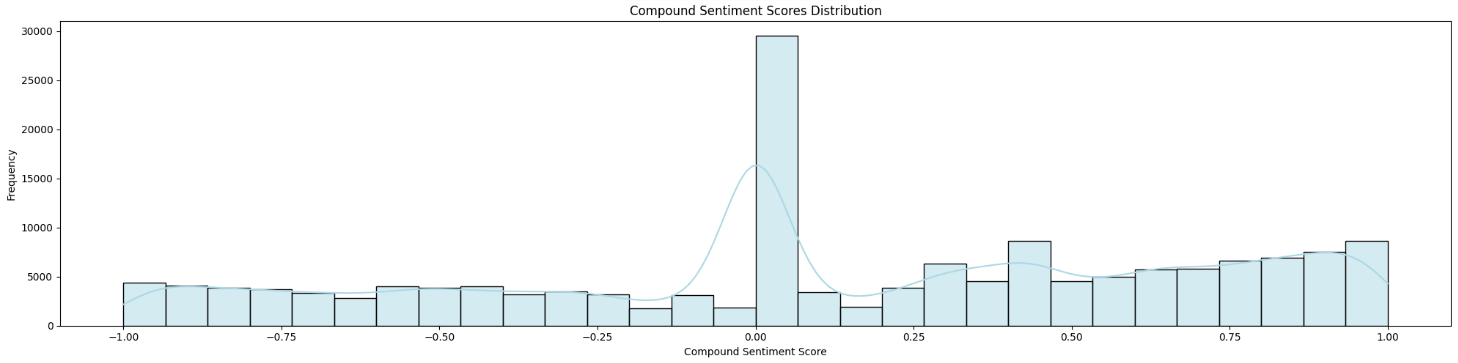


Processed Tokens Analysis

Furthermore, the Sentiment Scores were calculated to provide an overall sentiment analysis:

- **Negative** – negative sentiment from 0 to 1
- **Neutral** – neutral sentiment from 0 to 1
- **Positive** – positive sentiment from 0 to 1
- **Compound** – overall sentiment score from -1 to 1,
  -1 being the strong negative sentiment and 1 being the strong positive sentiment

Here only the Compound plot is shown: it confirms most of the comments are neutral (clean).



Feature Engineering

Vocabulary & Sequences Creation

Once the raw comments have been processed, the tokens obtained need to be converted into numerical sequences (the actual features) before being used as input for deep learning models:

- **Vocabulary Building** – only the 10,000 most frequent words were retained, with an additional token for out-of-vocabulary (<OOV>) words
- **Sequence Encoding** – each token was mapped to a unique integer ID
- **Padding and Truncation** – all sequences were set to a fixed length of 200 tokens using post-padding (zeros added at the end if shorter)
- **Embedding Layer** – reduces the input dimension to 50

Vocabulary & Sequences Creation

```
Preview of the first 20 elements in the vocab:
[('article', 1), ('page', 2), ('wikipedia', 3), ('talk', 4), ('will', 5),
```

A preview of the vocabulary elements

```
array([  443,    50,    54,   514,  4180, 10001,   638,   217,   110,
        5466,  2231,  2534,    44,  1003, 10001,  2413,     7,   136,
         172,     4,     2,  3047,    19,     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,     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,     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,     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=int32)
```

An example of Padded Sequence

Machine Learning Workflow

Methodology – Overview

Due to the sequential nature of text data, various RNN architectures were implemented. Additionally, a simple Fully-Connected Neural Network was used as a benchmark.

- **Fully-Connected Neural Network (Benchmark)**
- **LSTM (Baseline)**
- **GRU (Baseline)**
- **Bidirectional LSTM and GRU (Baseline and Regularized)**
- **Ensemble using Average Predictions of Regularized Models**

For the **Input layer**, each model uses an **Embedding** Layer.
For the **Output layer**, each model uses **six** units with the **sigmoid** function as the prediction is performed on six distinct binary labels.

Train – Validation Split

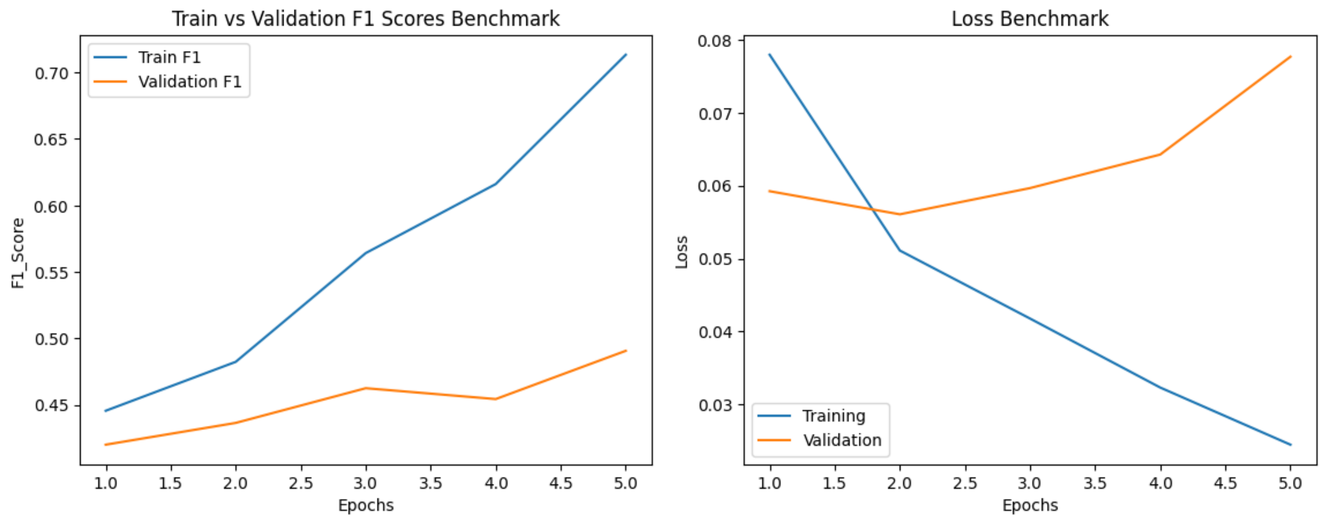
The dataset provided by Kaggle was already split into training and test sets.

The splitting has been done following traditional splitting percentages:

- **Train – 80%**
- **Validation – 20%**

Benchmark Model

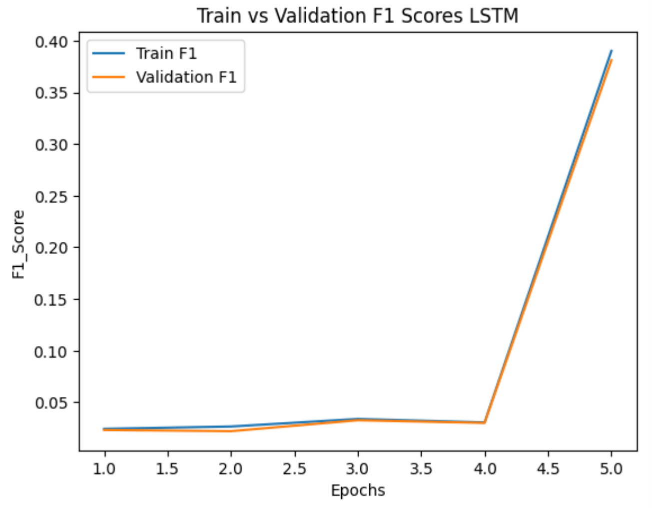
- **One Hidden Layer: 64** Units using **relu** as activation function
- **128 Batch Size**
- **5 Epochs**
- **Default Learning Rate using Adam**

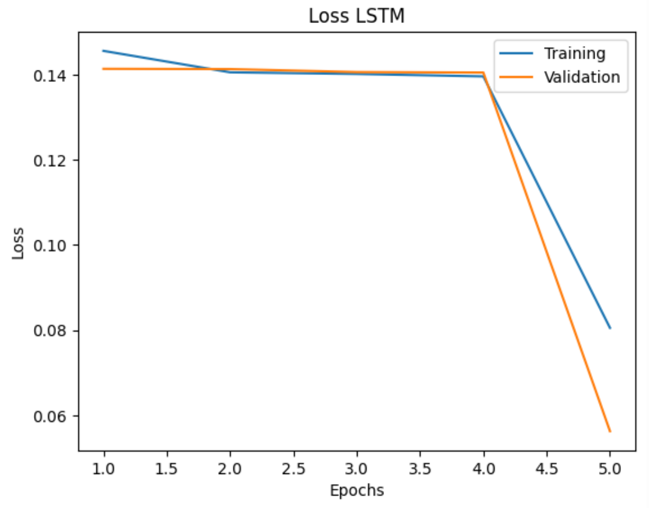


The benchmark model shows a lot of **overfitting** but already decent **F1 score.**

LSTM – Non-Regularized

- **One LSTM Hidden Layer: 64** Units using **tanh** as activation function
- **128 Batch Size**
- **5 Epochs**
- **Default Learning Rate using Adam**

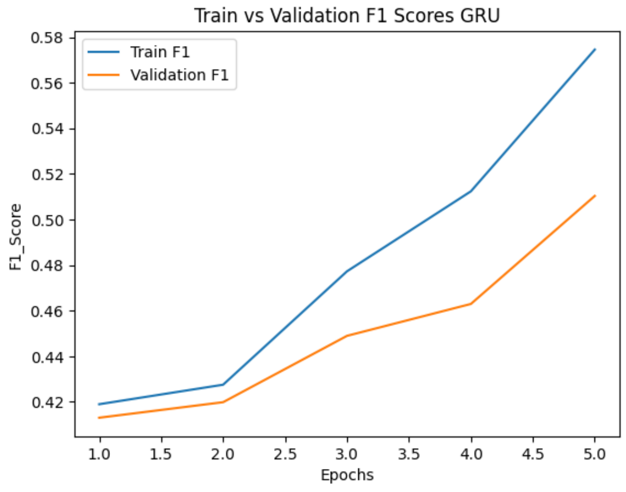


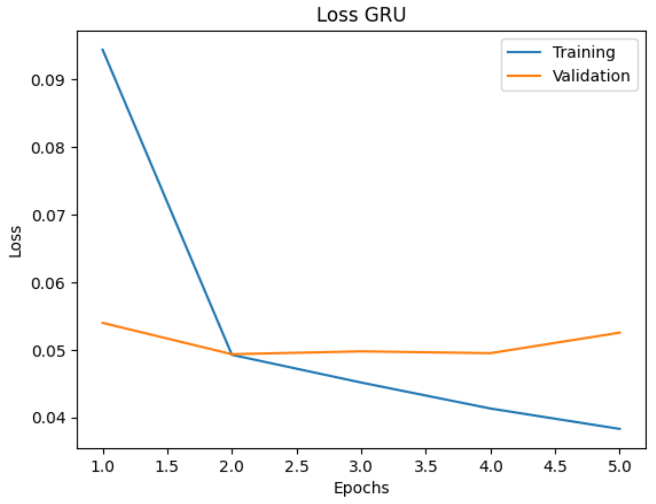


- The model **struggles** to converge until the last epoch.
- There could be some **underfitting**

GRU – Non-Regularized

- **One GRU Hidden Layer: 64** Units using **tanh** as activation function
- **128 Batch Size**
- **5 Epochs**
- **Default Learning Rate using Adam**

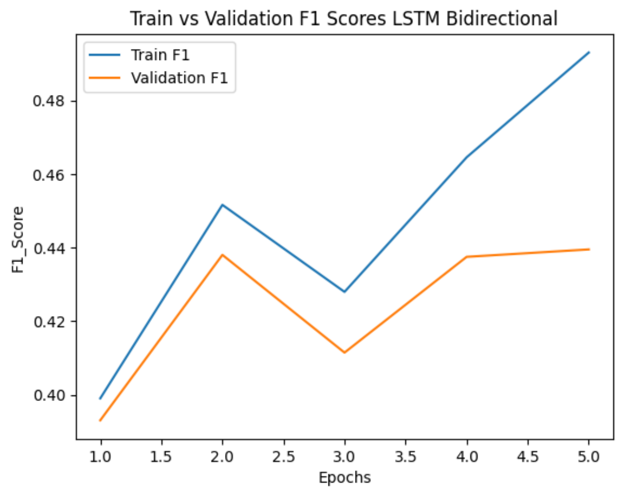


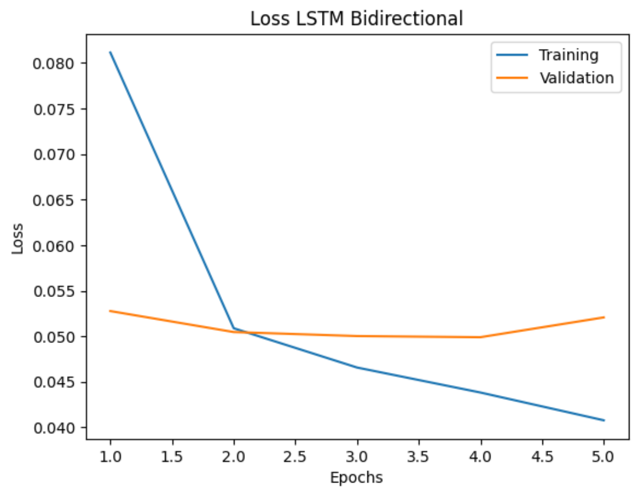


- Using GRU shows **improvements** over the LSTM model

LSTM Bidirectional – Non-Regularized

- **One LSTM Bidirectional Hidden Layer: 64** Units using **tanh** as activation function
- **128 Batch Size**
- **5 Epochs**
- **Default Learning Rate using Adam**

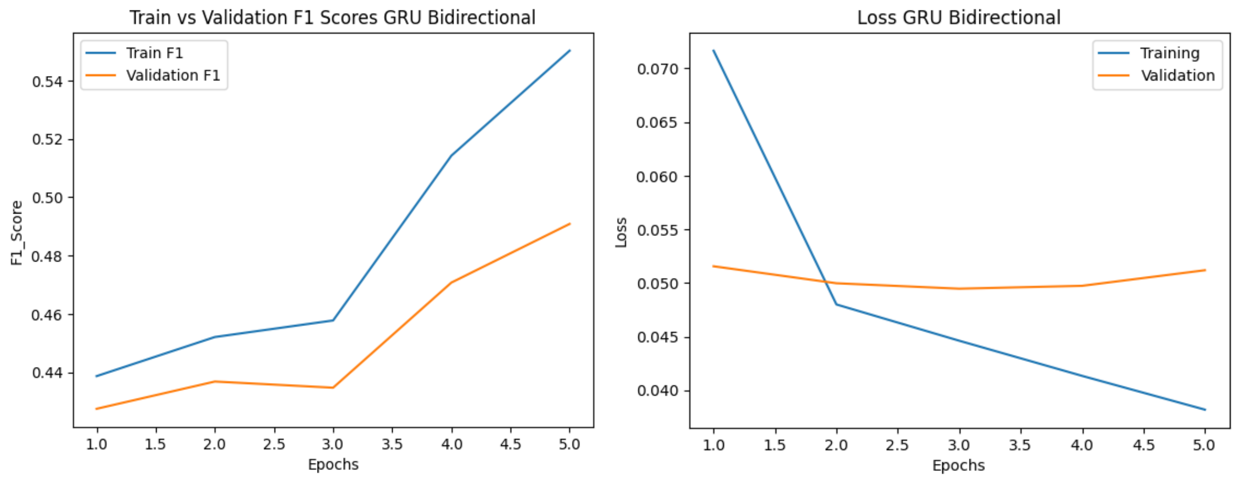




- The LSTM Bidirectional shows improvements compared to the baseline LSTM, with overall better performances and a smoother loss plot.

GRU Bidirectional – Non-Regularized

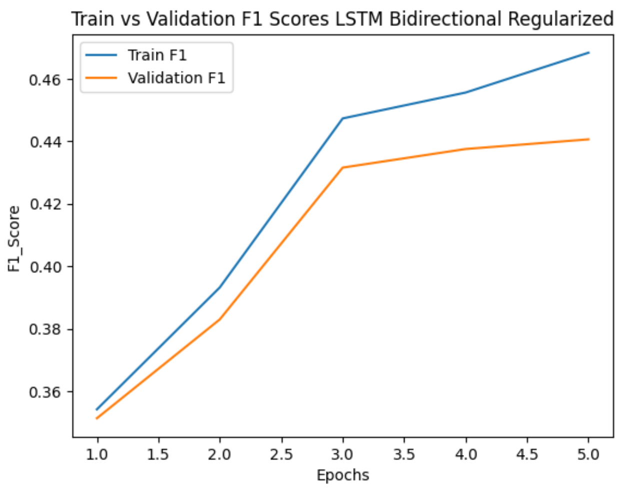
- **One GRU Bidirectional Hidden Layer: 64** Units using **tanh** as activation function
- **128 Batch Size**
- **5 Epochs**
- **Default Learning Rate using Adam**

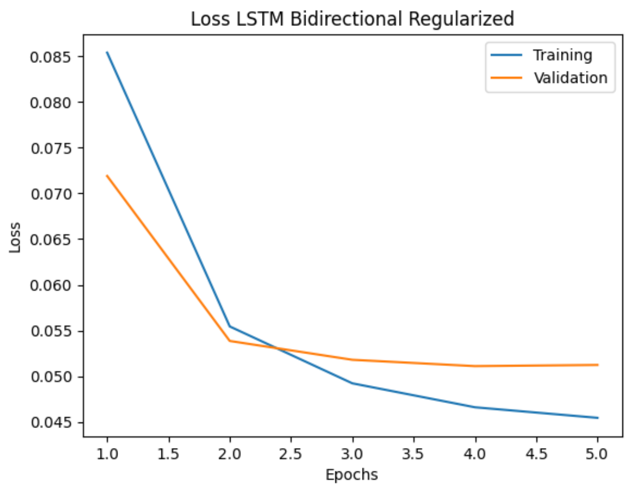


- The GRU Bidirectional shows the best perfomances overall, compared to the previous models, but also the most overfitting (even if the losses are really low).

LSTM Bidirectional – Regularized

- **EarlyStopping on Validation Loss** – Restore Best Weights = True
- **ReduceLROnPlateau** – Starting from 0.001
- **64 Units** for one Bi-LSTM Hidden Layer, using **Dropout of 0.3**
- **5 Epochs** maximum

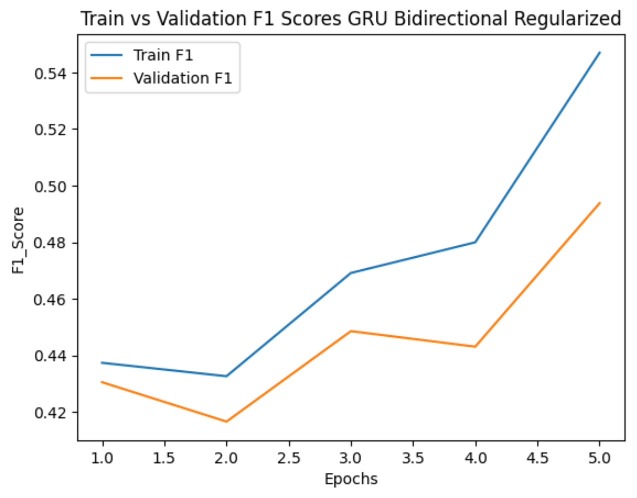


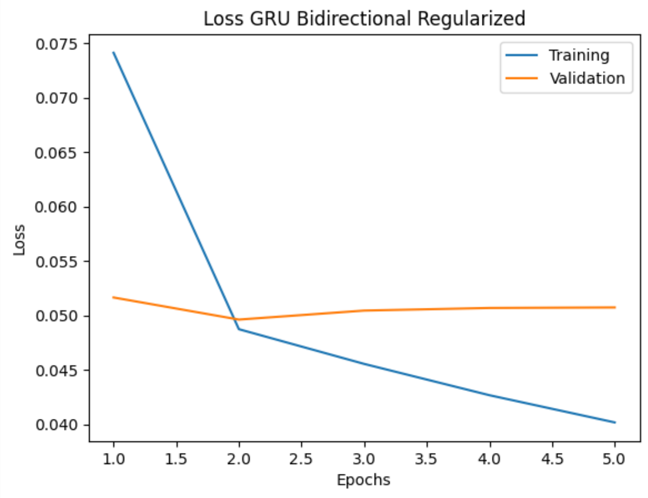


- The **loss curves** are much **closer** than the Baseline models.
- **EarlyStopping** restores the weights from the **fourth** epoch.

GRU Bidirectional – Regularized

- **EarlyStopping on Validation Loss** – Restore Best Weights = True
- **ReduceLROnPlateau** – Starting from 0.001
- **64 Units** for one Bi-GRU Hidden Layer, using **Dropout of 0.3**
- **5 Epochs** maximum





- Like the LSTM Regularized, the **loss curves** are very **close**.
- **EarlyStopping** restores the weights from the **second** epoch.

Ensemble Model

- Finally, the regularized models were used to create a simple ensemble by averaging their predictions.
- While ensembling typically yields better results when combining diverse models, it is still worth evaluating whether this approach provides any improvements, particularly on the test data, which will be analyzed later.

```
ROC AUC for each class:
Class 1: 0.9729
Class 2: 0.9881
Class 3: 0.9895
Class 4: 0.9526
Class 5: 0.9806
Class 6: 0.9592

[Ensemble] Mean Column-wise ROC AUC: 0.9738
```

Mean ROC AUC in Validation: 0.9738

Evaluation Summary – Train & Validation

Model	Train Macro F1	Train Loss	Val Macro F1	Val Loss
NN Benchmark	0.7133	0.0240	0.4907	0.0777
LSTM Baseline	0.3901	0.1078	0.3810	0.0563
GRU Baseline	0.5745	0.0379	0.5103	0.0525
Bi-LSTM Baseline	0.4930	0.0398	0.4395	0.0521
Bi-GRU Baseline	0.5502	0.0377	0.4909	0.0512
Bi-LSTM Regularized (Best Epoch)	0.4556	0.0472	0.4375	0.0511
Bi-GRU Regularized (Best Epoch)	0.4327	0.0477	0.4167	0.0496

- Bi-GRU Regularized shows the best trade-off between performance and overfitting, while GRU Baseline shows the best F1 performance in train and validation overall.

Evaluation Summary – Test

Model	Test Macro F1	Test Loss	ROC AUC
NN Benchmark	0.4696	0.0963	0.9573
LSTM Baseline	0.3446	0.0731	0.9595
GRU Baseline	0.4585	0.0887	0.9663
Bi-LSTM Baseline	0.3817	0.0843	0.9637
Bi-GRU Baseline	0.4470	0.0811	0.9678
Bi-LSTM Regularized (Best Epoch)	0.3667	0.0792	0.9615
Bi-GRU Regularized (Best Epoch)	0.3705	0.0749	0.9634
Ensemble	0.3734	0.0754	0.9643

- Bi-GRU Baseline shows the best performance but a higher loss, while the Ensemble seems the best trade-off.

EXTRA: Kaggle Submissions (Mean AUROC)

Model	Kaggle Private Score	Kaggle Public Score
NN Benchmark	0.9570	0.9589
LSTM Baseline	0.9595	0.9588
GRU Baseline	0.9660	0.9676
Bi-LSTM Baseline	0.9638	0.9630
Bi-GRU Baseline	0.9671	0.9703
Bi-LSTM Regularized (Best Epoch)	0.9614	0.9606
Bi-GRU Regularized (Best Epoch)	0.9635	0.9632
Ensemble	0.9644	0.9631

- The same can be told for the Kaggle Submissions

EXTRA: Custom Comments Evaluation

- The **ensemble model** was tested on **custom-made comments** to assess its classification performance on more unseen data.
- The custom comments were built based on **TF-IDF analysis**, ensuring proper word usage for the evaluation of each label.

```
# Another Sample Comment
example_comment_prediction("Welcome to Wikipedia, I will help you find more knowledge :-)")
```

```
--------------------
Comment Class: Clean
--------------------
```

	Class	Probability
0	toxic	0.002267
1	severe_toxic	0.000022
2	obscene	0.000344
3	threat	0.000290
4	insult	0.000976
5	identity_hate	0.000314

Clean comment

```
# Insult
example_comment_prediction("You are an idiot, fucker")
```

```
--------------------
Comment Class: Toxic
--------------------
```

	Class	Probability
0	toxic	0.996011
1	severe_toxic	0.416533
2	obscene	0.980459
3	threat	0.033969
4	insult	0.897906
5	identity_hate	0.114814

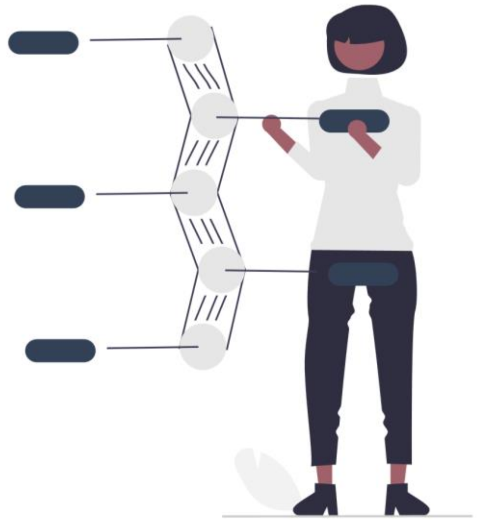
Toxic comment – Insult

Final Considerations

Conclusions and Considerations

Finally, after the results, the following can be observed:

- **GRU with regularization** provides a good trade-off between performance and generalization.
- **Ensemble model** ensured more stable predictions and improved metrics.
- **Bidirectional baseline architectures** increase ROC-AUC but introduce more overfitting than the regularized versions.
- **Regularization** helps mitigate overfitting without compromising too much the performance.



Future Work

As future work, it is possible to explore more approaches:

- Use **oversampling** to improve F1-score for underrepresented classes.
- Explore **CNNs and Transformer models (BERT)** for better feature extraction.
- Leverage **word embeddings (FastText, Word2Vec)** for improved text representation.
- Fine-tune **dropout, learning rate, and model depth** to balance predictions and efficiency.
- Apply **model pruning** to reduce computational complexity.

Thanks for the Attention!