# Task 1.1 - Cellula Technologies NLP Internship
# Toxic Topic Classification using LSTM
# Task Report

Abdelrahman Elaraby

September 2025

## Contents

# 1 Exploring the Data

This project is a text classification one. I was given a dataset of around 3000 samples that span 9 classes. Each row of the CSV file contains 3 columns; query, image descriptions, and Toxic Category.

**The nine classes are:** Child Sexual Exploitation, Elections, Non-Violent Crimes, Safe, Sex-Related Crimes, Suicide & Self-Harm, Unknown S-Type, Violent Crimes, and unsafe.

After investigating the dataset, the first thing I noticed is the presence of 1000 duplicated samples. So I removed the duplicates and only kept one instance. Then I noticed another problem, which is the dataset being imbalanced with some classes containing only 2 or 3 samples. You can see the distribution of samples across classes in Table 1.

| Toxic Category | Count |
|---|---|
| Safe | 879 |
| Violent Crimes | 691 |
| Non-Violent Crimes | 205 |
| unsafe | 141 |
| Unknown S-Type | 84 |
| Suicide & Self-Harm | 3 |
| Elections | 2 |
| Sex-Related Crimes | 2 |
| Child Sexual Exploitation | 2 |

Table 1: Distribution of samples across toxic categories.

So, I resorted to the data augmentation techniques, particularly data generation using LLMs.

# 2 Data Generation

I simply used ChatGPT to generate more samples for me. For each of the five classes containing few samples: Unknown S-Type, Suicide & Self-Harm, Elections, Sex-Related Crimes, and Child Sexual Exploitation. For each class I made a CSV file containing the existing samples, and I uploaded this file to ChatGPT along with a detailed prompt. Below I show an example prompt that was used to generate the data for "Unknown S-Type" class.

**The Prompt:**

```
given the attached CSV file (rare_1.csv) containing the Columns "query", "image
    descriptions ", and "Toxic Category". I need to synthesize more training examples
    for a text classifier that I am building. I removed the other classes for you and in
     that file you are only left with "Unknown S-Type" class. I need you to give me 200
    new examples (plus the 84 existing so total of 284).
for the query column generate texts of similar average length and containing similar
    sentences that talk about "Unknown S-Type" which is a generic class of sentences.
    make sure that the sentences are unique. don't make them too similar but also make
    them talk about the same topics and contexts discussed in the given 84 examples. for
     the image description column add one of the three examples:
        this 'A family enjoying a picnic in the park.',
        or this 'People walking through a peaceful city street.',
        or this  'A child playing in a sunny meadow.'
make their existence balanced across all the generated samples. for the Toxic Category
    column just keep the class name "Unknown S-Type".
```

After doing a similar thing to all the rare classes, I ended up with 5 CSV files. I merged those files with the one containing original unduplicated data and shuffled it. Finally I saved the new dataset in a new CSV file provided with the code in the LSTM folder.

# 3 Preprocessing and Word Embeddings

For preparing the data to the model, I decided to merge the two columns "query" and "image descriptions" into a single new column named "input". Then as a simple preprocessing step I decided to remove some of the stop words (subset of the nltk library stop words). Following that, I decided to use GloVe word embedding (Word2vec) as features for representing the words to the neural networks. The reason I chose this approach instead of arbitrary numerical representations such as (random integer encoding, bag-of-words, etc.) is that GloVe word embeddings are rich and trained on a much larger dataset, while the data I have is very limited.
For the labels (Toxic Category) I represented them as one-hot-encoded vectors. Then finally I splitted the data into 80% Training, 20% Validation, and 20% Testing shuffled datasets.

# 4 Model Building and Training

Buliding my classifier, I used a Bidirectional LSTM instead of a vanilla LSTM because it can capture both past and future context in the sequence. Since my dataset is fully available (not streamed word by word), I don't need to wait for sequential input, making BiLSTM a better choice for richer contextual representation. I used 128 units of LSTM (64 forward and 64 backward), and I added dropout and recurrent dropout to it. Moreover I added a 64 Feed Forward Dense layer with 64 units and dropout. Then the final Softmax layer with 9 neurons. In Table 2 you can see the model architecture and the number of parameters and size of each component.

| **Layer (type)** | **Output Shape** | **Param #** |
|---|---|---|
| Embedding (embedding_1) | (None, 120, 100) | 495,000 |
| Bidirectional (bidirectional_1) | (None, 128) | 84,480 |
| Dense (dense_2) | (None, 64) | 8,256 |
| Dropout (dropout_1) | (None, 64) | 0 |
| Dense (dense_3) | (None, 9) | 585 |
| **Total params** | | 1,764,965 (6.73 MB) |
| **Trainable params** | | 588,321 (2.24 MB) |
| **Non-trainable params** | | 0 (0.00 B) |
| **Optimizer params** | | 1,176,644 (4.49 MB) |

Table 2: Model summary of the LSTM-based classifier.

For the training part, I used Adam Optimizer and ran it for 20 epochs, with batch size 32. I also used class weights for weighted categorical cross entropy loss to focus on the classes containing fewer samples. You can see the learning curves in Figure 1. I took the best model across the learning curve which was at epoch 14.

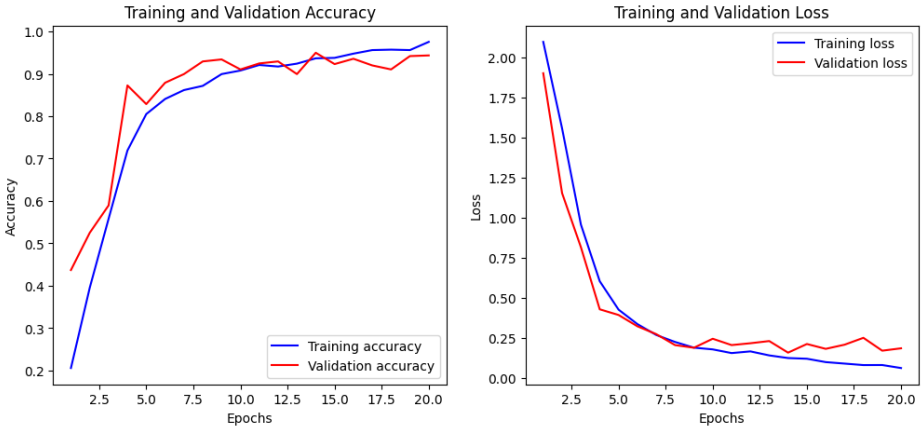


Figure 1: Training accuracy and loss curves over epochs.

# 5 Evaluation and Testing

For evaluating and testing this model on the test dataset, I used the following metrics: Precision, Recall, F1 Score, and Accuracy. Also I calculated the weighted and Macro Averages for Precision, Recall, and F1 Score. You can see all the calculated metrics in Table 3.

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Child Sexual Exploitation | 0.980 | 0.980 | 0.980 | 51 |
| Elections | 0.979 | 1.000 | 0.989 | 46 |
| Non-Violent Crimes | 1.000 | 1.000 | 1.000 | 41 |
| Safe | 0.899 | 0.915 | 0.907 | 176 |
| Sex-Related Crimes | 1.000 | 1.000 | 1.000 | 50 |
| Suicide & Self-Harm | 0.980 | 0.980 | 0.980 | 49 |
| Unknown S-Type | 0.774 | 0.719 | 0.745 | 57 |
| Violent Crimes | 1.000 | 1.000 | 1.000 | 138 |
| unsafe | 1.000 | 1.000 | 1.000 | 28 |
| **Accuracy** | | | 0.948 | 636 |
| **Macro avg** | 0.957 | 0.955 | 0.956 | 636 |
| **Weighted avg** | 0.947 | 0.948 | 0.948 | 636 |

Table 3: Classification report results.

I also visualized the confusion matrix between the 9 classes. It looks pretty good and the only two classes misclassified as each other were "Safe" and "Unknown S-Type" which is not a surprising thing for the fact that the classes share being similar with respect to being generic content. The Confusion Matrix is shown in Figure 2.

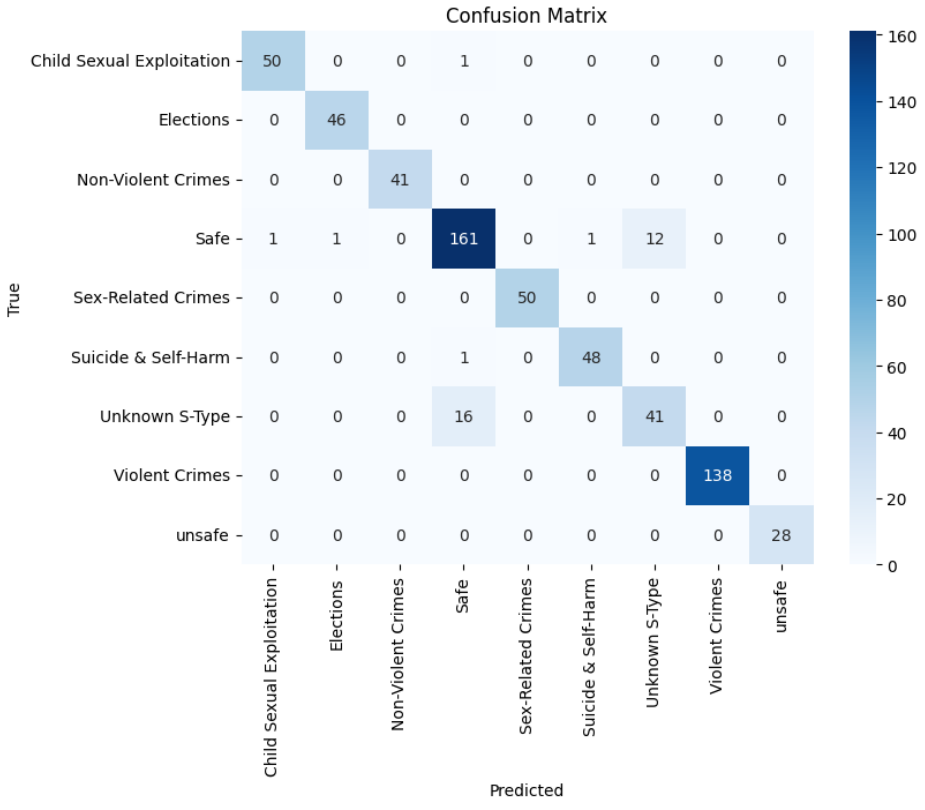


Figure 2: Confusion matrix of the classifier results.

# 6 Few Concerns

I am just skeptic about few points, and I wanted to list them here for discussion. The first concern is the possibility of spurious correlations and shortcut learning or also known as Clever Hans effect. Where the shortcut will be learning to classify based on the contents of "image descriptions" part of the row. This row only contains 12 examples, which is a very limited number, and maybe those samples are divided among the classes in a way that makes it easier for the model to identify those patterns instead of focusing on the high-variance part of the input (which is the query part). The second concern is that the DATA IS VERY LIMITED, and in order to build a real classifier to be deployed in a real production system the data needs to be big and representative.