Mini Project Report

on

A Common Framework for Hostile Content Detection

Submitted by

Annamaneni Rohit- 20bcs019

Gali Yaswanth - 20bcs046

Gudiseva Deepak Sujay- 20bcs049

Malem Vishnu Teja - 20bcs084

Under the guidance of

Dr Sunil Saumya

Assistant Professor

DEPARTMENT OF DATA SCIENCE AND INTELLIGENT SYSTEMS

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DHARWAD

11/05/2023

CERTIFICATE

It is certified that the work contained in the project report titled "A Common Framework for Hostile Content Detection" by "Annamaneni Rohit (Roll No: 20bcs019)", "Gali Yaswanth (Roll No:20bcs046)", "Gudiseva Deepak Sujay (Roll No: 20bcs049)" and "Malem Vishnu Teja (Roll No: 20bcs084)" has been carried out under my/our supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor(s)

Name(s)

Department(s)

(May, 2023)

Contents

List of Figures

List of Tables

1 Introduction

In today's digital age, social media platforms have become a powerful tool for people to share and connect with others. We have different platforms for different modalities of content like Twitter and Google for Text, Instagram and Facebook for Images, Youtube for videos and with the rise of hate and fake content, these platforms have also become a breeding ground for hostile content that can cause harm and division in society. Although there exists many content moderation solutions which are deployed by various MNCs to tackle this issue but they only work well for pure native languages.There exists very little content moderation systems for code-mixed datasets, where multiple languages and scripts are used together, making it challenging to detect and mitigate hostile content effectively. Code-Mixing (CM) is a natural phenomenon of embedding linguistic units such as phrases, words or morphemes of one language into an utterance of another[2].One such example of code mixed dataset is Hinglish where the semantic and linguistic meaning is in Hindi but it is expressed in English text.Following are some examples of the Hinglish text which represent Hate or Fake or Both:

T1: *"Koy is bhosidike ko bholo ki match Haar jayenge"*
T2: *"Arre bhai aj tho trump mar gaya"* **T3:** *"Kriti ke apne kutte ko m*
T4: *"Sarkar banne ke badh is desh worst ban gaye"*

Detecting hostile content in code-mixed datasets requires a approach that takes into account the complex linguistic and cultural context in which such content is produced. It involves leveraging advanced technologies such as machine learning, natural language processing, and data mining to identify patterns and trends that can help distinguish between genuine content and hostile content. The goal is to develop robust algorithms and tools that can automatically detect and flag hostile content in real-time, thereby protecting users from harm and fostering a safer, more inclusive online environment. In this article, we will explore some of the latest research and developments in the field of hostile content detection for code-mixed datasets.

2 Related Work

Elnaggar et al. [5] explored the application of multi-task learning (MTL) models in the legal domain, demonstrating how these models can overcome the challenge of requiring large amounts of labeled data for deep learning methods. By sharing knowledge between different tasks, MTL models enhance the learning for each task, making them a popular choice in the deep learning community.The rise of transformers, which was first proposed by Vaswani et al. [9], had a significant impact on the machine learning community by overcoming the bottleneck of computational efficiency for sequential data. BERT models, as introduced in Devlin et al. [4], have been shown to outperform existing models in a variety of classification tasks, such as toxicity detection in text comments, as demonstrated by Angel et al. [3]. Additionally, studies have found that monolingual BERT models have an advantage over multilingual BERT models in toxicity detection in text comments in their pre-trained language

While MTL has been explored in the NLP domain, mixed results have been reported, and little is known about the specific task relations that can lead to gains from MTL models over single-task setups, as noted in Bingel and Søgaard [1]. However, recent studies have proposed deep architectures, such as those proposed by Liu et al. [6], that can be trained jointly on multiple related tasks using an external memory shared by several tasks. These architectures have been shown to improve the performance of a task with the help of other related tasks. Similarly, Liu et al. [7] proposed three different mechanisms of sharing information to model text with task-specific and shared layers based on recurrent neural networks, showing that the entire network can be trained jointly on all tasks, and that the proposed models can improve the performance of a task with the help of other related tasks. In GerEval 2021 [8] empirically Morgan et al. proves that Multitask learning outperforms Single task learning where the task is multi labels classification with lables being "Toxic", "Engaging" and "Fact-Claiming"

3 Data and Methods

The task of identifying fake and hate speech presents a multi-label classification problem where the presented text is to be classified under hate and fake speech. The initial data set contained labels for only hate speech in an attempt to explore the performance of various models and the effects of various pre-processing techniques. As of now there is no publicly available CodeMixed dataset which has both Hate and Fake labels so we are using a pseudo-labels for identifying Fake.

3.1 Data and preprocessing

The text in the data set was presented in a code-mixed hinglish format. The features of the data set are shown in Table 1.

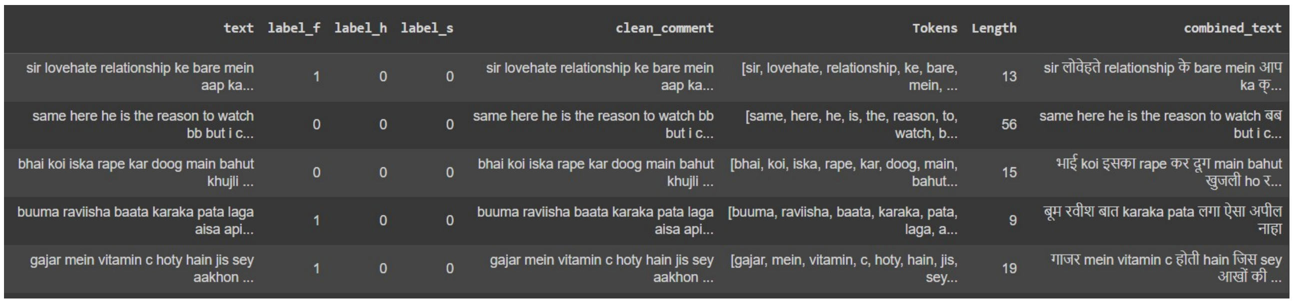
3.1.1 Pre-processing tasks

There were several pre-processing tasks at hand. Since the text was obtained from twitter posts, the texts contained emojis,urls and hyperlinks which were filtered and removed. Another task was to remove the stop words from the text and this proved quite difficult as there was no corpus available in order to detect stop words in hinglish format.

Text	Label_f	Label_h	Label_s
sir lovehate relationship ke bare mein aap ka kya vichar hai pictwittercom kqp-gaf	1	0	0
same here he is the reason to watch bb but i cant hate him because ladai unhi se karte hain jinki baaton ka humein asar ho and he is not opportunist.	0	0	0
bhai koi iska rape kar doog main bahut khujli ho rahi hai https twittercom rishibagree status	0	0	0
buuma raviisha baata karaka pata laga aisa apiila naha	1	0	0

Table 1
Uncleaned Dataset

Moreover, the variations in the spellings used in the text caused further issues in identifying and removing hinglish stop words. The nltk corpus of stop words was used to remove the stop words which are in pure English.Bohra et al. [2] provides more insights into the various pre-processing tasks involved with tweet data.
In order to deal with the issues caused by misspelling of words, transliteration tasks was performed to modify the dataset and rectify those errors. The dataset after cleaning and transliteration is given in figure 1



text	label_f	label_h	label_s	clean_comment	Tokens	Length	combined_text
sir lovehate relationship ke bare mein aap ka...	1	0	0	sir lovehate relationship ke bare mein aap ka...	[sir, lovehate, relationship, ke, bare, mein, ...	13	sir लोवेहते relationship के bare mein आप ka क्...
same here he is the reason to watch bb but i c...	0	0	0	same here he is the reason to watch bb but i c...	[same, here, he, is, the, reason, to, watch, b...	56	same here he is the reason to watch बब but i c...
bhai koi iska rape kar doog main bahut khujli ...	0	0	0	bhai koi iska rape kar doog main bahut khujli ...	[bhai, koi, iska, rape, kar, doog, main, bahut...	15	भाई koi इसका rape कर दूग main bahut खुजली ho र...
buuma raviisha baata karaka pata laga aisa api...	1	0	0	buuma raviisha baata karaka pata laga aisa api...	[buuma, raviisha, baata, karaka, pata, laga, a...	9	बूम रवीश बात karaka pata लगा ऐसा अपील नाहा
gajar mein vitamin c hoty hain jis sey aakhon ...	1	0	0	gajar mein vitamin c hoty hain jis sey aakhon ...	[gajar, mein, vitamin, c, hoty, hain, jis, sey...	19	गाजर mein vitamin c होती hain जिस sey आखों की ...

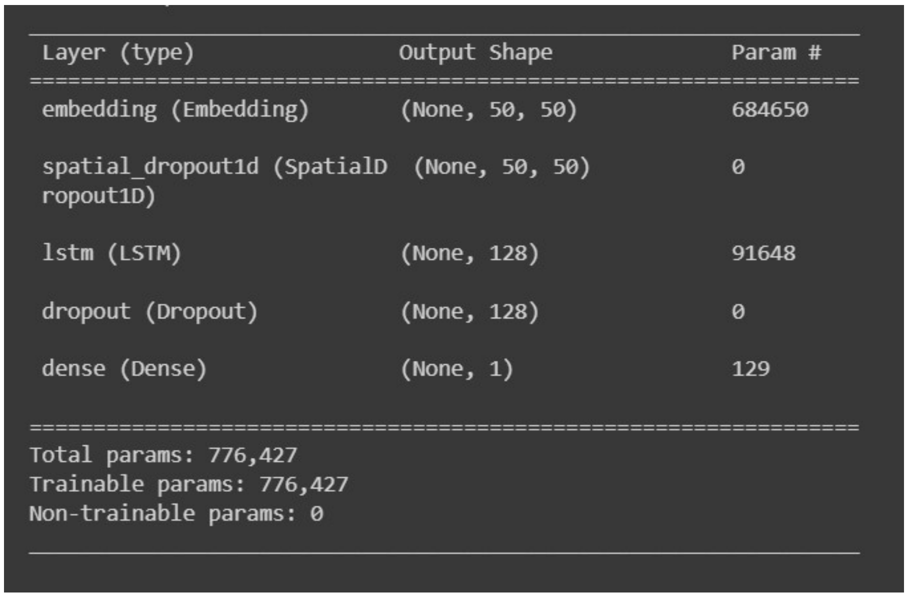
Figure 1. Dataset after cleaning and transliteration

3.2 Single-Task Training

We first used a dataset consisting of a single label and constructed models to predict on just the single model to analyze performance on isolated tasks. In this case, we performed binary classification on hate speech and use the following methods.

3.2.1 LSTM model using word2vec embeddings

For the LSTM model, we attempted to analyze the performance when custom word2vec embeddings were used as inputs. The word2vec embeddings were created using gensim and the vocabulary belonging to the dataset. The LSTM model uses an embedding layer, followed by an LSTM layer and then a dropout layer. The output from the dropout layer is then fed to a dense layer using the sigmoid function for classification. The model summary is as shown in figure 2.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 50, 50)            684650

 spatial_dropout1d (SpatialD  (None, 50, 50)           0
 ropout1D)

 lstm (LSTM)                 (None, 128)               91648

 dropout (Dropout)           (None, 128)               0

 dense (Dense)               (None, 1)                 129

=================================================================
Total params: 776,427
Trainable params: 776,427
Non-trainable params: 0
_________________________________________________________________
```

Figure 2. LSTM model for single task learning

However, the word2vec embeddings present various challenges. Firstly, the mistakes in the spellings cause the same word to have different embeddings and may not even be similar in terms of cosine similarity. Secondly, the corpus formed by the dataset is not large enough to accurately model the word similarity.

3.2.2 Bert model

BERT(Bidirectional Encoder Representation of Transformer)[4] is a SOTA Transformer Architecture which has been pretrained on a large corpus and trained for several days in multi GPU environment to perform various downstream tasks. This model has been used for both

Single Task learning and Multi-Task Training. The model performed moderately well with a Binary Cross Entropy loss of 1.04,however it still needs a lot of improvement.

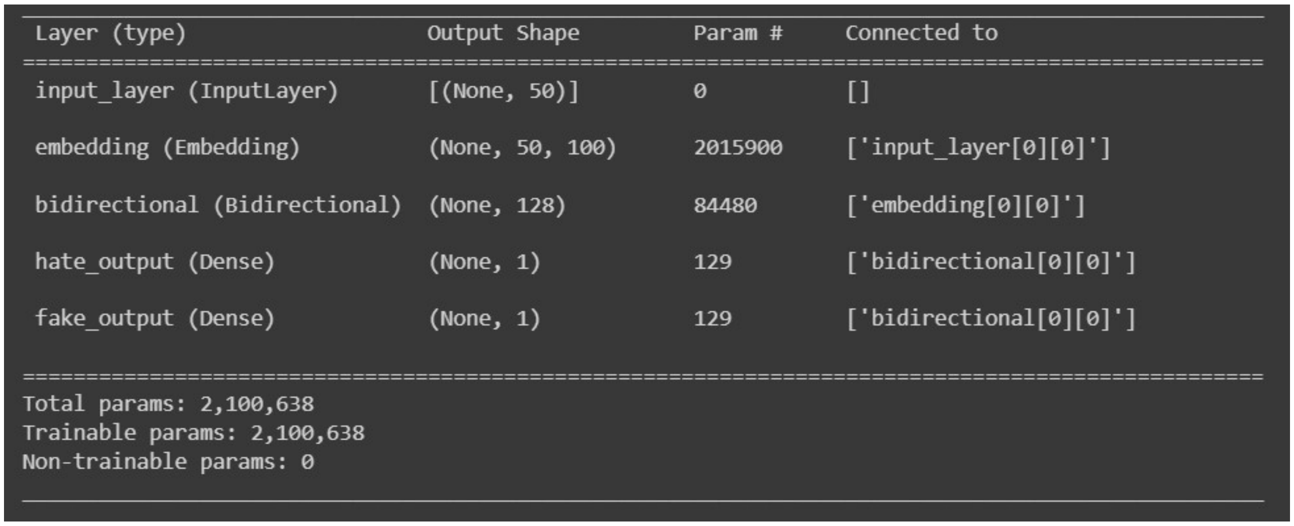
3.3 Multi-Task Training

Multi-task learning is a machine learning approach that involves training a single model to perform multiple related tasks simultaneously. In this approach, the model learns to share and transfer knowledge between the different tasks, which can improve the overall performance of each individual task.The idea behind multi-task learning is that the shared representations learned by the model can capture commonalities between the different tasks, which can lead to better generalization and higher accuracy than training separate models for each task.

For our project, detecting hate and fake content are two related tasks that can benefit from multi-task learning. By training a single model to perform both tasks, the model can learn to identify the common features that are indicative of hate and fake content, which can improve the metrics of both tasks.

3.3.1 BiLSTM model

Before feeding the data to the BiLSTM model, we must tokenize the input sentences and then pad or truncate the sentences to convert them to padded sequences of constant length. The constructed model architecture summary is given in Figure 4. The model consists of one input layer, one embedding layer and a Bidirectional LSTM layer. The model also uses two dense layers both of which take input directly from the same bidirectional LSTM layer for classification. One dense output layer is to classify for fake output while the other is intended for hate output.



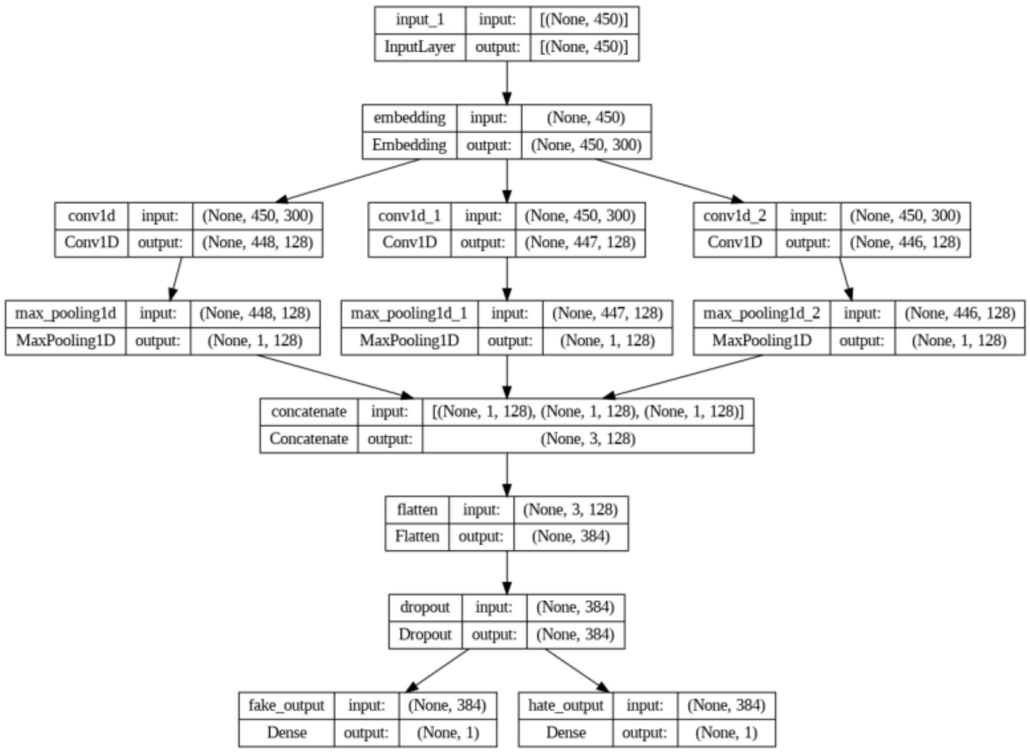
```
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_layer (InputLayer)       [(None, 50)]         0           []

 embedding (Embedding)          (None, 50, 100)      2015900     ['input_layer[0][0]']

 bidirectional (Bidirectional)  (None, 128)          84480       ['embedding[0][0]']

 hate_output (Dense)            (None, 1)            129         ['bidirectional[0][0]']

 fake_output (Dense)            (None, 1)            129         ['bidirectional[0][0]']

==================================================================================================
Total params: 2,100,638
Trainable params: 2,100,638
Non-trainable params: 0
```

Figure 3. BiLSTM model summary for MTL

Figure 4. testunfksh

3.3.2 CNN model

The CNN model uses the same steps of preprocessing before feeding the input as the BiLSTM model. The inputs are tokenized and brought to fixed lengths using tokenization and padding. The model consists of one input and embedding layer followed by three convolutional layers each of which is seperately connected to the embedding layer. The results of the convolutional layer are then passed to max pooling layers for down sampling. This is then concatenated, flattened and passed onto two separate dense layers one of which classifies the input as fake and the other classifies for hate. The model summary is given in figure 5



input_1 | input: | [(None, 450)]
InputLayer | output: | [(None, 450)]
embedding | input: | (None, 450)
Embedding | output: | (None, 450, 300)
conv1d | input: | (None, 450, 300)
Conv1D | output: | (None, 448, 128)
conv1d_1 | input: | (None, 450, 300)
Conv1D | output: | (None, 447, 128)
conv1d_2 | input: | (None, 450, 300)
Conv1D | output: | (None, 446, 128)
max_pooling1d | input: | (None, 448, 128)
MaxPooling1D | output: | (None, 1, 128)
max_pooling1d_1 | input: | (None, 447, 128)
MaxPooling1D | output: | (None, 1, 128)
max_pooling1d_2 | input: | (None, 446, 128)
MaxPooling1D | output: | (None, 1, 128)
concatenate | input: | [(None, 1, 128), (None, 1, 128), (None, 1, 128)]
Concatenate | output: | (None, 3, 128)
flatten | input: | (None, 3, 128)
Flatten | output: | (None, 384)
dropout | input: | (None, 384)
Dropout | output: | (None, 384)
fake_output | input: | (None, 384)
Dense | output: | (None, 1)
hate_output | input: | (None, 384)
Dense | output: | (None, 1)

Figure 5. CNN Model Architecture

4 Results

In this study, we investigated the effectiveness of three different models, namely BiLSTM, CNN, and Bert. We trained all three models for 10 epochs and measured their performance based on the F1-score. The loss curves for each model are shown in figure 6. Despite having a significantly lower training loss curve compared to BiLSTM, the validation loss of CNN shows a steep increase, indicating that the model is overfitting. It can be observed that, despite being pre-trained on a large dataset, Bert is not yielding the best results in this study. This is due to the code-mixed nature of the dataset being used, which does not fully leverage the pre-trained knowledge of Bert.

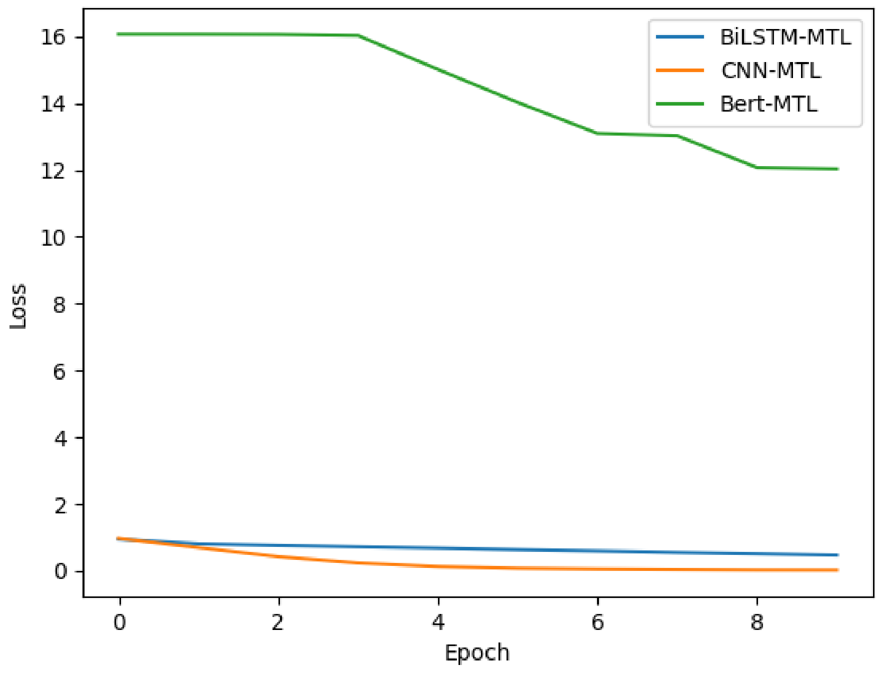


Figure 6. Loss Curve

Table 2 presents the results of our experiments using the CNN-MTL and BiLSTM-MTL models. The table shows the loss values for both hate and fake labels, as well as the overall loss value for each model. The CNN-MTL model has a higher overall loss value of 0.8962, with a hate loss of 0.8484 and a fake loss of 0.7064. In contrast, the BiLSTM-MTL model has a lower overall loss value of 0.4947, with a hate loss of 0.1786 and a fake loss of 0.31619. These results indicate that the BiLSTM-MTL model performed better than the CNN-MTL model in terms of hate, fake and overall loss.

Table 2
Results table

MODEL	Loss	Hate Loss	Fake Loss
CNN-MTL	0.9742	0.3856	0.5887
BiLSTM-MTL	0.4745	0.1746	0.2998

After determining that the BiLSTM model performed the best among the three models tested, we generated a classification report to evaluate its performance. Table 3 is the classification report for the BiLSTM model for both fake and hate. The Table 3a shows precision, recall, and f1-score for both non-hate and hate whereas Table 3b shows same for fake.

(a) Hate Classification Report

	precision	recall	f1-score
0	0.88	0.92	0.90
1	0.51	0.39	0.45
macro avg	0.70	0.66	0.67
weighted avg	0.82	0.83	0.83

(b) Fake Classification Report

	precision	recall	f1-score
0	0.60	0.56	0.58
1	0.74	0.77	0.75
macro avg	0.67	0.66	0.67
weighted avg	0.68	0.69	0.69

Table 3
Classification report

Figure 7 describes the confusion matrix for hate whereas Figure 8 describes the confusion matrix for fake. The rows of the matrix represent the actual class labels of the data, while the columns represent the predicted class labels.

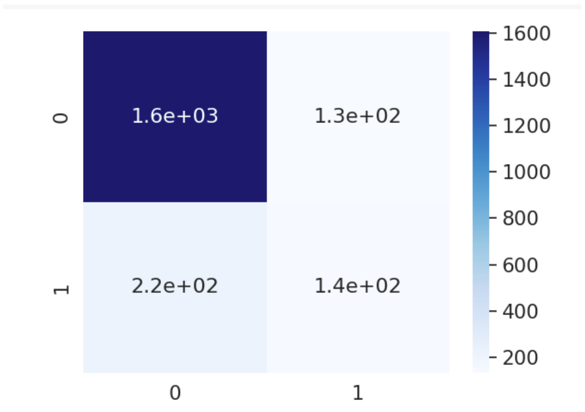


Figure 7. Confusion matrix for hate

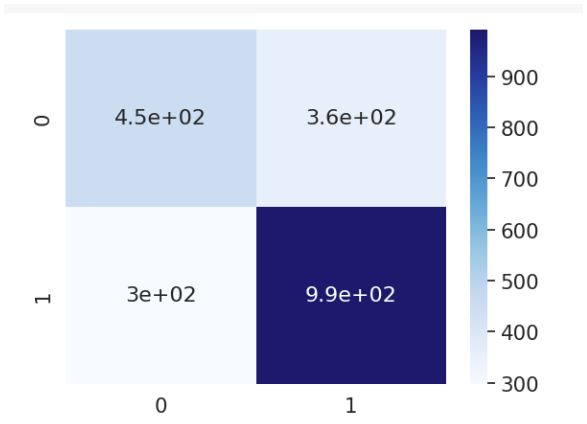


Figure 8. Confusion matrix for fake

5 Discussion

Initially, we only had hate-labelled dataset, on which we built our baseline transformer model along with RNN, but both of them failed to achieve the desired results. Afterwards, we decided to extract embeddings using the BERT model and used these embeddings as inputs for our LSTM with attention model, which also performed on par with the previous two models. Later on, we obtained our pseudo-dataset containing both hate and fake labels. Through a study of various research papers, we learned that multi-task learning would work better in this case. Therefore, we used three MTL models: BERT-MTL, BiLSTM-MTL, and CNN-MTL models.

One limitation of our model is that it requires multi-label data containing both fake and hate labels for training. However, the dataset we used for our experiments only had hate labels and the fake labels were obtained through a pseudo-labeling process. As a result, the quality of the dataset may not be optimal for training the model to detect both hate and fake labels. This limitation highlights the need for more high-quality multi-label datasets to improve the performance of the model in real-world scenarios.

For future plans we can incorporate additional sources of information, such as user features which could potentially improve the model's performance. For example, features such as age, gender, and location may provide useful information for identifying hate and fake content. Additionally we can further investigate the use of other multi-task learning approaches. For instance, instead of training the model to classify hate and fake labels together, the tasks could be trained separately, and then the model could be fine-tuned to improve performance on both tasks simultaneously. Alternatively, transfer learning approaches could be employed, where the model is first pre-trained on a large codemix dataset and then fine-tuned on the hate and fake classification tasks.

6 Conclusion

In conclusion, the digital age has witnessed the immense power of social media platforms in connecting and sharing content. However, along with the positive aspects, there has been a rise in hostile content that can cause harm and division in society. While content moderation solutions exist for native languages, there is a lack of effective systems for code-mixed datasets, where multiple languages are used together. Code-mixing, such as Hinglish, poses challenges for detecting and mitigating hostile content. In both an STL and an MTL environment, we have tested neural transformer models. The MTL environment consistently surpassed the STL environment, indicating that using shared learning techniques enhances the performance of each individual task. The application of multi-task learning models, such as transformers and BERT, help overcome the need for large labeled datasets. These models leverage shared knowledge between tasks to enhance learning and improve performance. Multi-task learning has shown promise for identifying both hate and fake speech simultaneously. The sharing of knowledge and the extraction of common features between tasks can improve the overall performance of each individual task. These encouraging findings motivate us to study more about how multi-task models might transfer learning.

References

[1] Joachim Bingel and Anders Søgaard. Identifying beneficial task relations for multi-task learning in deep neural networks, 2017.

[2] Aditya Bohra, Deepanshu Vijay, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. A dataset of Hindi-English code-mixed social media text for hate speech detection. In *Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media*, pages 36–41, New Orleans, Louisiana, USA, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-1105. URL `https://aclanthology.org/W18-1105`.

[3] Angel Felipe Magnossão de Paula and Ipek Baris Schlicht. Ai-upv at iberlef-2021 detoxis task: Toxicity detection in immigration-related web news comments using transformers and statistical models, 2021.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[5] Ahmed Elnaggar, Bernhard Waltl, Ingo Glaser, Jörg Landthaler,

Elena Scepankova, and Florian Matthes. Stop illegal comments: A multi-task deep learning approach, 2018.

[6] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Deep multi-task learning with shared memory, 2016.

[7] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning, 2016.

[8] Skye Morgan, Tharindu Ranasinghe, and Marcos Zampieri. Wlv-rit at germeval 2021: Multitask learning with transformers to detect toxic, engaging, and fact-claiming comments, 2021.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.