Predicting the Type and Target of Offensive Posts in Social Media

Dinesh Kumar M R
CEN Department
Amrita Vishwa Vidyapeetham

K S Paval
CEN Department
Amrita Vishwa Vidyapeetham

Shreya Sanghamitra
CEN Department
Amrita Vishwa Vidyapeetham

Shrish Surya N T
CEN Department
Amrita Vishwa Vidyapeetham

***Abstract*—There has been a great deal of research toward recognizing potentially offensive messages as offensive content has become ubiquitous in social media. However, prior research on this subject concentrated on identifying relatively particular categories of offensive content, such as hate speech, cyberbullying, or cyber-aggression, rather than taking into account the issue as a whole. Here, however, we specifically target a variety of offensive content. We specifically use a hierarchical task model to determine the kind and audience of offensive remarks on social media. To achieve this, we created the Offensive Language Identification Dataset (OLID), a brand-new dataset of tweets that have been carefully reviewed and annotated using a three-layer, fine-grained annotation system. We made this dataset freely available online. We go over the key parallels and divergences between OLID and existing datasets for applications like hate speech recognition, aggressiveness detection, and related ones. We continue to test and evaluate the effectiveness of various machine-learning models on OLID.**

***Index Terms*—OLID, hate speech, social media**

I. Introduction

Detecting and classifying inappropriate language on social media is a critical and difficult issue [1]. The first few papers in this review concentrate on the creation and assessment of machine learning models and algorithms for this goal. The first research, in particular, provides a hybrid technique that employs many sorts of information to anticipate the nature and target of hostile messages [2]. The following studies investigate several deep learning algorithms for identifying and categorizing foul languages, such as CNNs, LSTMs, and Attention Mechanisms [3] [4] [5] [6]. Other publications concentrate on particular subtasks of offensive language detection, such as recognizing personal assaults [7], sarcasm [8], and hate speech [9] [10].

There are additional studies that look at the consequences of various aspects, such as linguistic, semantic, and syntactic features, as well as cross-cultural variances in the perception of offensive language [11] [12]. Finally, there are articles that look at how crowdsourcing and active learning may be used to increase the quality of annotated data, which is essential for training and testing machine learning models [13] [14]. Overall, this underlines the difficulty of detecting objectionable language in social media, as well as the need for more study in this field to meet the issues provided by the dynamic nature of social media platforms and the vast variety of languages and cultures represented online.

A. Problem Definition

The problem addressed in this study is that there is a need for a thorough method of identifying and categorizing offensive messages, including the sort of offensive language used and the target of the offending content because offensive content has grown prevalent in social media. There is a dearth of studies on the intended audience of offensive language, despite previous studies focusing on specific categories of offensive content, such as hate speech or cyberbullying.

II. Materials & Methodology

A. Dataset

A hierarchical dataset called OLID is used to pinpoint the nature and audience of inflammatory texts on social media. The Twitter data was gathered and is openly accessible. There are 14,100 total tweets, of which 860 are in the test set and 13,240 are in the training set. It can be visualized in Table 1. We utilize a hierarchical annotation schema divided into three levels in the OLID dataset to differentiate between whether or not the language is offensive (A), its type (B), and its target (C).

1) Level A: Detection of profanity: The following categories of tweets are differentiated at Level A:

- *Not offensive (NOT):* Posts devoid of insults or vulgar language;
- *Offensive (OFF):* Posts that contain any objectionable language (profanity) or an explicit or covert offense. This includes comments that are derogatory, obscene or involve threats.

2) Level B: Categorization of Offensive Language: Level B classifies the offense's type as follows:

- *Targeted Insult (TIN):* Posts that criticize or threaten a specific person, group, or other people
- *Untargeted (UNT):* Posts containing non-targeted profanity and cursing. Posts with generic profanity contain unacceptable language but are not targeted.

3) Level C: Target Identification for Offensive Language: The targets of insults and threats are categorized as Level C:

- *Individual (IND):* Posts that are directed at a certain individual, an identified person, or an unnamed conversational participant. Cyberbullying is frequently defined as insults and threats directed at specific people.

- *Group (GRP):* Posts that are directed at a group of people that are seen as a unit because they have a common racial, ethnic, gender, sexual orientation, political, or other belief. The majority of taunts and threats directed at a group fit the definition of hate speech.
- *Other (OTH):* These objectionable posts do not refer to any of the organizations, situations, events, or problems mentioned in the first two categories.

A	B	C	Training	Test	Total
OFF	TIN	IND	2,407	100	2,507
OFF	TIN	OTH	395	35	430
OFF	TIN	GRP	1,074	78	1,152
OFF	UNT	—	524	27	551
NOT	—	—	8,840	620	9,460
All			13,240	860	14,100

Fig. 1: Label combination distribution in OLID

B. Exploratory Data Analysis (EDA)

The analysis of each sub-task has been visualized using histograms in our approach, it can be viewed in fig 2, 3, and 4. From these images, we can come to know that there are NOT - 8840, OFF - 4400 in sub-task A; TIN - 3876, UNT - 524 in sub-task B; IND - 2407, GRP - 1074, OTH - 395 in sub-task C.

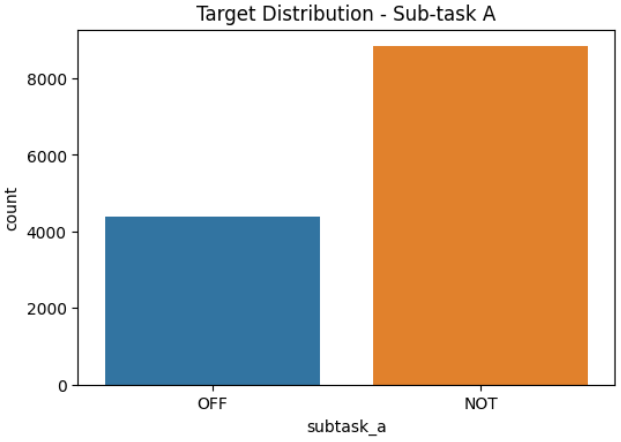

Fig. 2: Target Distribution (Sub-task A)

After this, we visualized the distribution of text length in fig 5. Finally, we have made a word frequency analysis where we have displayed the top 20 most frequent words in fig 6.

C. Data preprocessing

Now coming to the data preprocessing part where we :

- *remove Twitter handles or mentions from the text by replacing the string '@USER' with an empty string.*

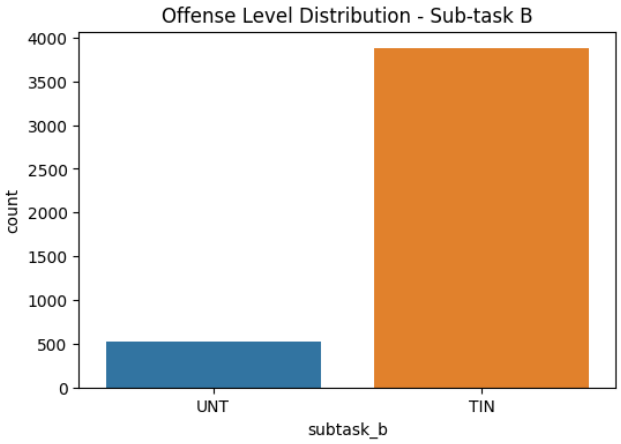

Fig. 3: Offense Level Distribution (Sub-task B)

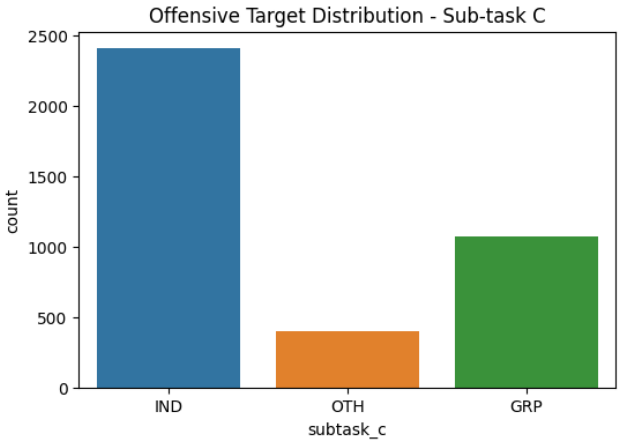

Fig. 4: Offensive Target Distribution (Sub-task C)

- *eliminate URLs or website links from the text by removing the string 'URL'.*
- *convert the HTML entity '&' back to the word 'and'.*
- *remove the '&lt' and '&gt' entities from the tweets.*
- *convert all characters in the text to lowercase for normalization purposes.*
- *remove punctuation marks from the tweets.*
- *remove all punctuation marks from the text.*
- *removes any non-ASCII characters from the tweets.*
- *remove any sequence of digits from the tweets.*
- *remove leading and trailing whitespace from the text.*

D. Methods implemented

We have implemented various models:

1) SVM: After the pre-processing of the data we split the data into training and test sets for each subtask. Then using the TF-IDF vectorizer converts the text data into numerical feature vectors for each subtask. Now we initialize separate SVM models for each subtask. We are using C-SVM here which is the C-Support Vector Classification (C-SVC) for multi-class classification tasks. C-SVC is a variant of SVM that is suitable for solving multi-class classification problems.

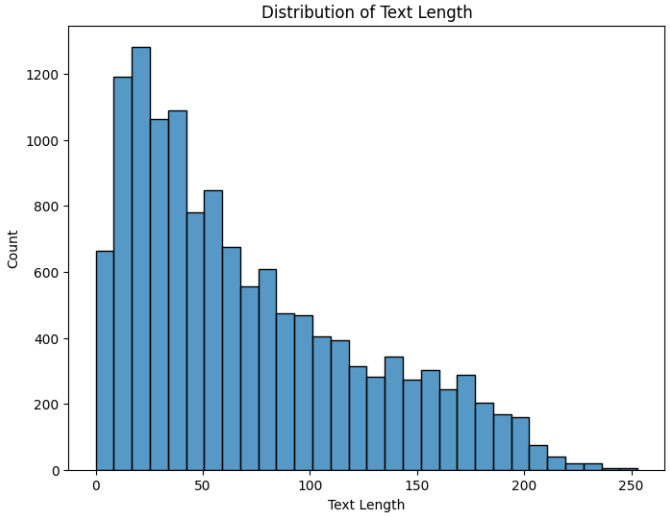

Fig. 5: Distribution of Text Length

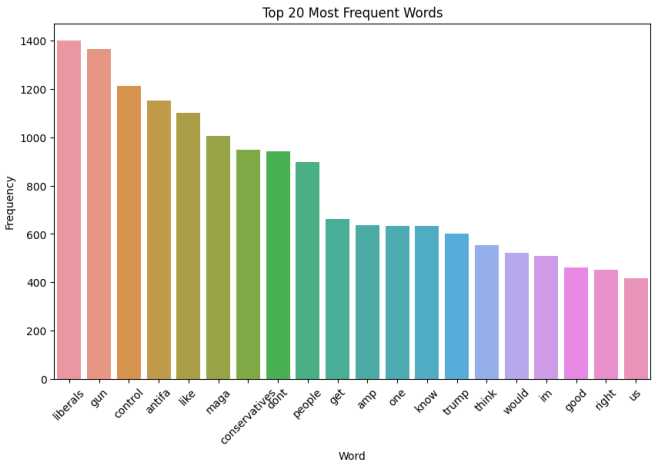

Fig. 6: Top 20 Most frequent words

It constructs a hyperplane or set of hyperplanes in a high-dimensional feature space to separate the different classes. The C parameter, which is not explicitly specified in the code, determines the trade-off between maximizing the margin and minimizing the classification error. The default value of C is 1.0. After this, we Train the SVM models using the training data and corresponding labels. Then we vectorize the test data using the TF-IDF vectorizer for each subtask. Now with the help of the trained SVM models, we predict the labels for the test data. The flowchart can be visualized in fig 7.

2) BiLSTM: The Bidirectional Long Short-Term Memory (BiLSTM) model is a type of recurrent neural network (RNN) architecture that is well-suited for processing sequential data such as text.

After the pre-processing steps, the tweets are tokenized and encoded using a tokenizer. The encoded tweets are padded to ensure they have the same length for modeling.

The BiLSTM model is built using the Keras library. It

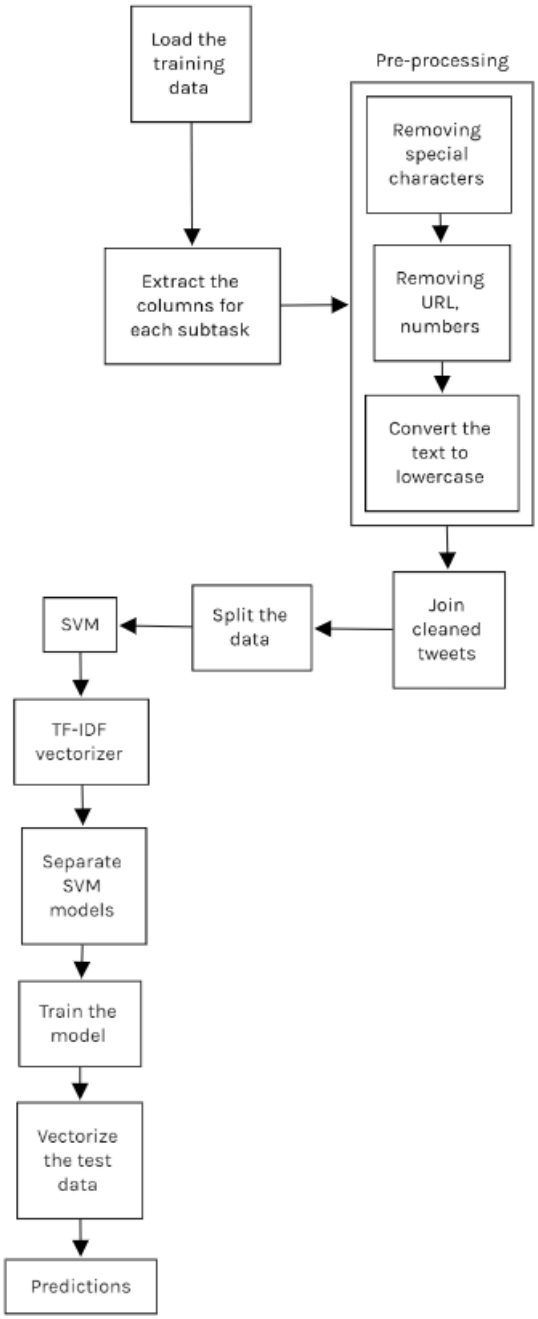

Fig. 7: Flow of SVM

consists of an embedding layer that learns dense representations of the words, followed by a Bidirectional LSTM layer that processes the sequential information in both forward and backward directions. This allows the model to capture contextual information from both past and future words. The LSTM layer is followed by a dense layer with ReLU activation and a dropout layer to prevent overfitting. The final layer uses sigmoid activation for binary classification. The model architecture can be seen in fig 8

The model is compiled with binary cross-entropy loss and the Adam optimizer. During training, the model is fit on the

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 500, 128)	1280000
bidirectional (Bidirectional)	(None, 256)	263168
dense (Dense)	(None, 128)	32896
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129

Total params: 1,576,193
Trainable params: 1,576,193
Non-trainable params: 0

Fig. 8: Model architecture of BiLSTM model

training data and validated on the validation data. The training process is performed for a specified number of epochs.

After training, the model is evaluated on the test data. The evaluation metrics include loss, binary accuracy, precision, and recall. Finally, the model is used to make predictions on the test data, and a classification report is generated.

3) RNN: Once the data is pre-processed The tweets and labels are joined together to create the training data. The RNN model is built using the Keras library. The model architecture consists of an embedding layer, an LSTM layer, dense layers, and an output layer.

The first layer is an embedding layer, which learns dense representations of words. It maps the words to a lower-dimensional vector space, capturing semantic relationships between them. The embedding layer helps the model understand the meaning and context of words in the text.

Following the LSTM layer, there are dense layers with ReLU activation. These layers introduce non-linearity to the model and help transform the extracted features into higher-level representations. The number of neurons in the dense layers can be adjusted based on the complexity of the classification task.

The output layer consists of a dense layer with a sigmoid activation function. This layer produces the binary classification output, indicating the predicted probability of the positive class. For multi-class classification, the number of units in the output layer would correspond to the number of classes, and a softmax activation function would be used instead. The model's architecture can be seen in fig 9.

After defining the model architecture, it is compiled with the binary cross-entropy loss function, which is suitable for binary classification problems. The RMSprop optimizer is used for training the model.

After the training phase of the model, it's been evaluated on the test data. It provides metrics such as loss and accuracy, which indicate how well the model performs on unseen data.

4) BERT Meta-Learning: It combines the powerful BERT (Bidirectional Encoder Representations from Transformers)

Layer (type)	Output Shape	Param #
inputs (InputLayer)	[(None, 150)]	0
embedding_1 (Embedding)	(None, 150, 50)	50000
lstm_1 (LSTM)	(None, 64)	29440
FC1 (Dense)	(None, 256)	16640
activation_2 (Activation)	(None, 256)	0
dropout_1 (Dropout)	(None, 256)	0
out_layer (Dense)	(None, 1)	257
activation_3 (Activation)	(None, 1)	0

Total params: 96,337
Trainable params: 96,337
Non-trainable params: 0

Fig. 9: Model architecture of RNN model

model with a meta-learner architecture to perform multi-class text classification.

The implementation follows a two-step process. First, the base BERT model is used to extract contextualized representations of the input text. These representations are obtained by encoding the input sequences with the BERT tokenizer and passing them through the base BERT model. The output is the last hidden state of the model, which represents the contextualized embeddings of the input tokens.

Next, the meta-learner model is defined, which takes the contextualized embeddings from the base BERT model as input. The meta-learner model consists of a few dense layers with ReLU activation, followed by a final dense layer with a softmax activation function. This architecture enables the meta-learner to learn and make predictions on the text classification task. The architecture can be seen in fig

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 512)	393728
dense_1 (Dense)	(None, 256)	131328
dense_2 (Dense)	(None, 2)	514

Total params: 525,570
Trainable params: 525,570
Non-trainable params: 0

Fig. 10: Model architecture of Meta-Learning model

The training loop involves iterating over the training dataset and performing a forward pass through the base BERT model. The contextualized embeddings obtained from the base BERT model are then fed into the meta-learner model, and the loss

is computed using sparse categorical cross-entropy. The gradients are calculated using backpropagation, and the optimizer is used to update the meta-learner model's parameters.

After training, the model is evaluated on the validation dataset. The contextualized embeddings are obtained from the base BERT model for the validation inputs, and the meta-learner model generates predictions based on these embeddings. The predicted labels are compared to the true labels, and a classification report is generated.

In summary, this implementation utilizes BERT for extracting contextualized embeddings and employs a meta-learner architecture to perform multi-class text classification. It combines the strengths of both models, allowing for effective representation of learning and classification in a single framework.

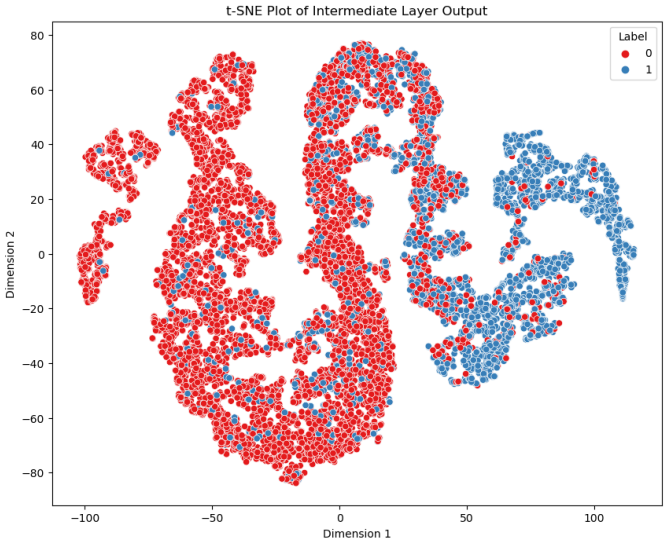

Fig. 11: TSNE plot for Sub-task A (RNN)

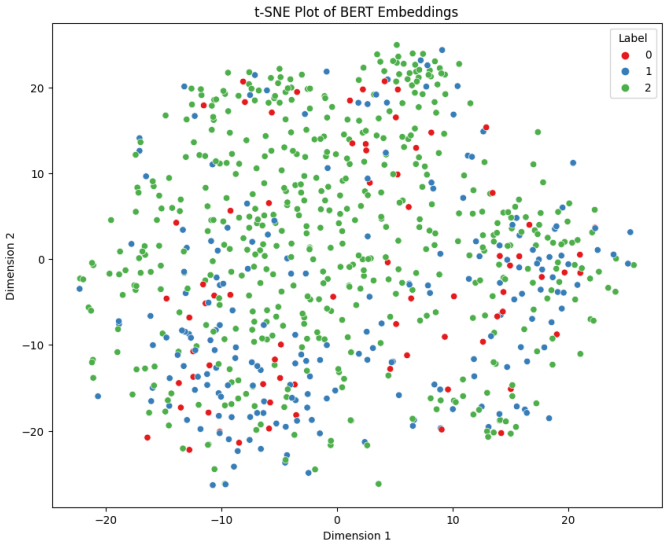

Fig. 12: TSNE plot for Sub-task C (BERT)

E. Feature Visualization

t-Distributed Stochastic Neighbor Embedding (t-SNE) is an unsupervised, non-linear technique primarily used for data exploration and visualizing high-dimensional data. In simpler terms, t-SNE gives you a feel or intuition of how the data is arranged in a high-dimensional space.

F. Performance Metrics

1) Accuracy: It measures the overall correctness of the model's predictions.

The formula for calculating accuracy is:

$$\text{Accuracy} = \frac{\text{Number of correctly classified instances}}{\text{Total number of instances}} \quad (1)$$

2) Precision: It measures the proportion of correctly predicted positive instances out of all instances predicted as positive. It provides an indication of the model's ability to avoid false positives.

The formula for calculating precision is:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (2)$$

3) Recall (Sensitivity): It measures the proportion of correctly predicted positive instances out of all actual positive instances. It provides an indication of the model's ability to capture all positive instances.

The formula for calculating recall is:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (3)$$

4) F1 Score: It is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance by considering both precision and recall. F1-score combines precision and recall into a single metric.

The formula for calculating the f1 score is:

$$\text{F1 score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

In the context of multi-class text classification, these metrics are calculated by considering the performance of the model across all classes. The true positives, false positives, and false negatives are determined for each class individually, and then the metrics are calculated by aggregating the values across all classes using appropriate averaging methods (e.g., macro-average or weighted average).

III. Results and Discussion

In this section, we present the results of our experiments using the SVM, BiLSTM, RNN, and BERT Meta-Learning methods. We proceeded to divide the dataset into training and validating groups in an 80:20 ratio.

A. Offensive Language Detection

The performance on discriminating between offensive (OFF) and non-offensive (NOT) posts are reported in Table 1. We can see that all models perform significantly better than chance, with the neural models performing substantially better than the SVM. The RNN and the BERT model, achieve an accuracy of 0.77. We have visualized the confusion matrix for this task for the RNN model in fig 13.

TABLE I: Results for offensive language detection (Level A). We report Precision (P), Recall (R), and F1 for each model/baseline on all classes (NOT, OFF), and weighted averages. Accuracy is also listed (best in bold).

Model	NOT			OFF			Weighted Average			Accuracy
	P	R	F1	P	R	F1	P	R	F1	
SVM	0.73	0.97	0.84	0.85	0.34	0.48	0.78	0.75	0.71	0.74
BiLSTM	0.83	0.80	0.81	0.53	0.58	0.55	0.75	0.74	0.74	0.75
RNN	0.80	0.90	0.85	0.63	0.43	0.51	0.76	0.77	0.76	**0.77**
BERT	0.77	0.90	0.83	0.71	0.48	0.58	0.75	0.75	0.74	**0.77**

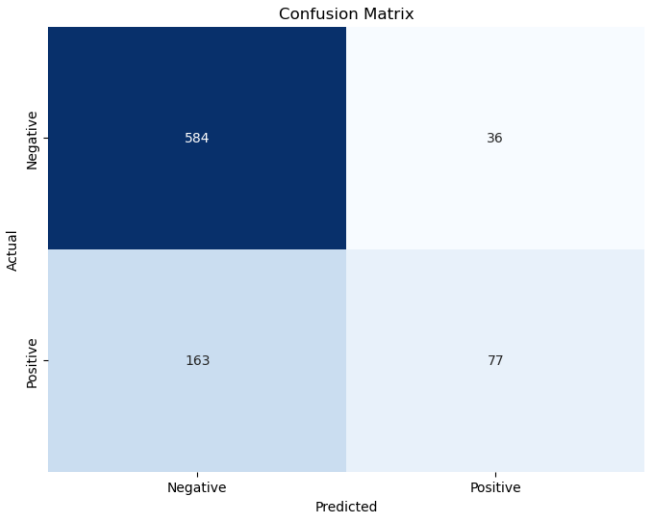

Fig. 13: Confusion matrix of Sub-task A (RNN)

B. Categorization of Offensive Language

In this set of experiments, the models were trained to discriminate between targeted insults and threats (TIN) and untargeted (UNT) offenses, which generally refer to profanity. The results are shown in Table 2. We can see that the RNN performs better than the SVM, with an accuracy of 0.89 and the BERT model outperforms the RNN model with an accuracy of 0.90 Note that all models perform better at identifying TIN compared to UNT. We have visualized the confusion matrix for this task for the SVM model in fig 14.

TABLE II: Results for offensive language categorization (Level B). We report Precision (P), Recall (R), and F1 for each model/baseline on all classes (TIN, UNT), and weighted averages. Accuracy is also listed (best in bold).

Model	TIN			UNT			Weighted Average			Accuracy
	P	R	F1	P	R	F1	P	R	F1	
SVM	0.87	1.00	0.93	1.00	0.02	0.04	0.89	0.88	0.82	0.88
BiLSTM	0.94	0.76	0.84	0.29	0.67	0.40	0.86	0.75	0.79	0.75
RNN	0.94	0.90	0.92	0.32	0.63	0.42	0.88	0.86	0.87	0.89
BERT	0.87	1.00	0.93	0.42	0.50	0.64	0.86	0.87	0.91	**0.90**

C. Offensive Language Target Identification

The results of the offensive target identification experiment are shown in Table 3. Here the models were trained to distinguish between three targets: a group (GRP), an individual (IND), or others (OTH). We can see that all three models

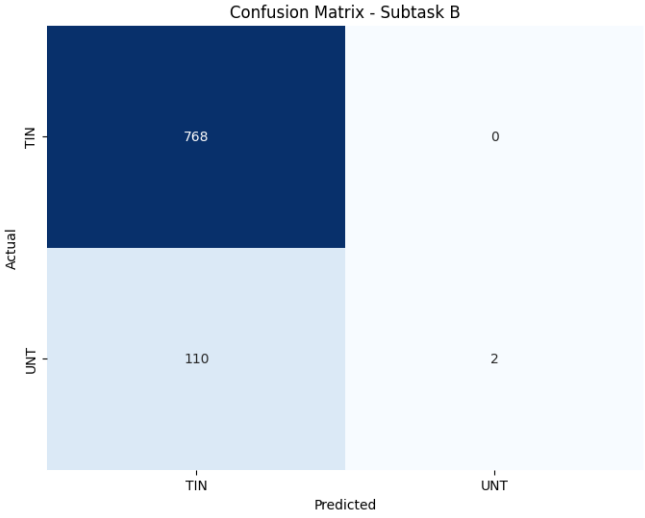

Fig. 14: Confusion matrix of Sub-task B (SVM)

achieved similar results, far surpassing the random baselines, with a slight performance edge for the neural models.

The performance of all models for the OTH class is 0, which can be explained by two factors. First, unlike the two other classes, OTH is a heterogeneous collection of targets. It includes offensive tweets targeted at organizations, situations, events, etc., thus making it more challenging for models to learn discriminative properties for this class. Second, there are fewer training instances for this class compared to the other two: there are only 395 instances for OTH vs. 1,075 for GRP and 2,407 for IND. We have visualized the confusion matrix for this task for the BERT-Meta model in fig 15.

TABLE III: Results for offensive target identification (Level C). We report Precision (P), Recall (R), and F1 for each model/baseline on all classes (GRP, IND, OTH), and weighted averages. Accuracy is also included.

	GRP			IND			OTH			Weighted Average			
Model	P	R	F1	P	R	F1	P	R	F1	P	R	F1	Accuracy
SVM	0.60	0.48	0.53	0.75	0.92	0.82	0.00	0.00	0.00	0.64	0.72	0.64	0.72
BiLSTM	0.32	0.24	0.27	0.43	0.71	0.54	0.00	0.00	0.00	0.31	0.40	0.34	0.40
RNN	0.75	0.60	0.67	0.63	0.94	0.75	0.00	0.00	0.00	0.57	0.66	0.60	0.67
BERT	0.54	0.56	0.55	0.78	0.86	0.82	0.21	0.04	0.07	0.67	0.70	0.68	**0.74**

IV. Conclusion and Future Work

In this paper, we have proposed a meta-learning approach for text classification using a combination of a base model and a meta-learner model. The base model, based on the BERT architecture, is used to extract contextualized representations of text, while the meta-learner model leverages these representations to make predictions for the target task.

While our meta-learning approach has shown promising results, there are several avenues for future research and improvement. Here are some potential directions for future work:

- *Exploring different meta-learning algorithms*
- *Investigating different base models*

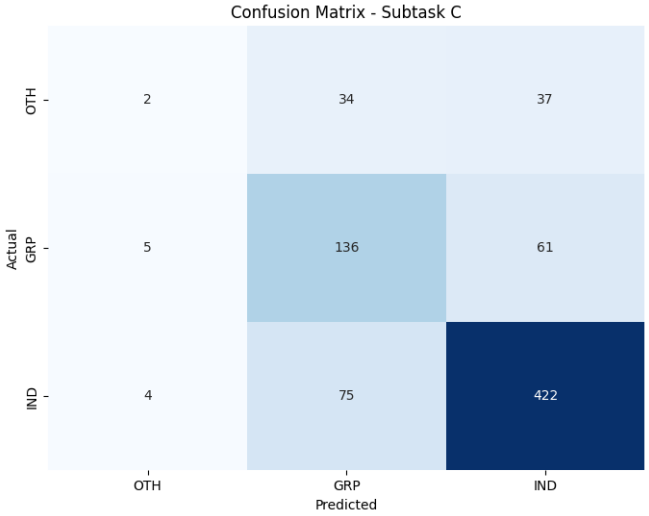

Fig. 15: Confusion matrix of Sub-task C (META)

- *Handling imbalanced datasets*
- *Applying the approach to other text classification tasks*
- *Interpretability and explainability*

In conclusion, our meta-learning approach for text classification has shown promising results, and future work can focus on exploring different meta-learning algorithms, investigating alternative base models, addressing imbalanced datasets, applying the approach to various text classification tasks, and enhancing the interpretability of the model's predictions. These advancements can contribute to the development of more robust and accurate text classification models.

REFERENCES

[1] Burnap, P., & Williams, M. L. (2015). Cyber hate speech on Twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2), 223-242.

[2] Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., & Kumar, R. (2019). Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 114-123).

[3] Djuric, N., Zhou, D., Morris, R. R., & Grbovic, M. (2015). Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web* (pp. 29-30).

[4] Zhang, Z., Luo, L., & Wang, K. (2018). Deep Learning for Offensive Language Detection in Tweets. In *Proceedings of the 2018 IEEE/WIC/ACM International Conference on Web Intelligence* (pp. 736-742).

[5] Nobata, C., Tetreault, J., Thomas, A., & Meurers, D. (2016). Abusive language detection in online user content. In *Proceedings of the First Workshop on Abusive Language Online* (pp. 47-56).

[6] Qian, L., Zhang, Y., & Wu, L. (2020). Leveraging Hierarchical Attention Networks and External Knowledge for Abusive Language Detection. *IEEE Access*, 8, 19763-19774.

[7] Wulczyn, E., Thain, N., & Dixon, L. (2017). Ex machina: Personal attacks seen at scale. arXiv preprint arXiv:1708.00163.

[8] Barbieri, F., Ballesteros, M., & Saggion, H. (2018). What do we know about sarcasm detection? A survey. arXiv preprint arXiv:1802.00471.

[9] Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017). Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media* (pp. 512-515).

[10] Fortuna, P., Nunes, S., & Rodrigues, P. (2018). A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4), 1-30.

[11] Pavalanathan, U., & Eisenstein, J. (2015). Confounds and consequences in geotagged twitter data. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (pp. 2361-2370).

[12] Fersini, E., Pozzi, F., & Messina, E. (2018). Cross-lingual and cross-cultural hate speech analysis: A case study on Italian and Greek. *Information Processing & Management*, 54(4), 605-621.

[13] Waseem, Z., & Hovy, D. (2016). Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter. In *Proceedings of the NAACL Student Research Workshop* (pp. 88-93).

[14] Ross, B., Alex, B., Andrew, M., Lucas, N., & Sridhar, V. (2017). Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (pp. 526-537).