# Assignment 4: Automatic Classification of Hate Speech (Methods)

Paul Constantinescu, Andres Garcia Rincon,
Mithat Can Ozgun, Liza Darwesh
Group 12

4 October 2024

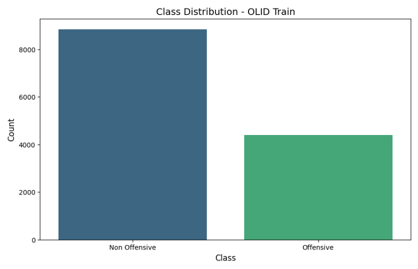
## 1 Data

### 1.1 OLID Dataset

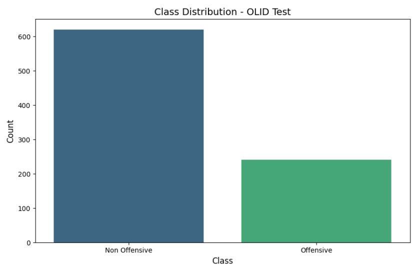
The OLID (Offensive Language Identification Dataset) was introduced as part of the SemEval-2019 Task 6 [3]. It consists of over 14,000 English tweets annotated for offensive language detection. The dataset features a hierarchical structure with three annotation levels: (A) offensive vs. non-offensive, (B) targeted vs. untargeted offense, and (C) the target of the offense (group, individual, or other). This study focuses on the binary classification task at Level A, which distinguishes between offensive and non-offensive tweets.

| Statistic (Test Set) | Non Offensive | Offensive |
|---|---|---|
| Count | 8840 (620) | 4400 (240) |
| Distribution | 66.7% (72%) | 33.3% (28%) |

Table 1: Statistics for the OLID train dataset. Test set values in parenthesys.



(a) OLID Train Set Distribution



(b) OLID Test Set Distribution

Figure 1: OLID Train and Test Set Distributions

### 1.1.1 Creation and Source

The data collection process involved a trial annotation conducted by experts to validate the proposed tagset and assess the retrieval method. Following this, crowdsourcing was employed for large-scale annotation, with annotators tasked with labeling tweets across all three annotation levels. To ensure quality, multiple annotators reviewed each instance, and a consensus was reached through majority voting in cases of disagreement. Approximately 30% of the dataset is labeled as offensive, achieved by specifically targeting keywords that are indicative of offensive content.

The primary aim of OLID is to provide a comprehensive resource for various offensive language identification and characterization tasks, supporting both binary and fine-grained classification. The dataset's annotations enable exploration of specific characteristics of offensive language, such as the nature of the offense and the targeted entity, which are crucial for developing more sophisticated and context-aware hate speech detection systems. In this study, we utilize OLID for in-domain experiments, specifically focusing on the binary classification task at Level A.

## 1.2 HASOC Dataset

The HASOC (Hate Speech and Offensive Content Identification) dataset was released for the HASOC 2019 shared task [4]. It includes multilingual data, with a significant portion in English, designed for the identification of hate speech and offensive content. The English segment of this dataset is employed in this study for cross-domain experiments. Similar to OLID, the HASOC dataset also contains multi-level annotations; however, we utilize it for binary classification to maintain consistency.

| Statistic | Non Offensive | Offensive |
|---|---|---|
| Count | 3591 | 2261 |
| Distribution | 61.4% | 38.6% |

Table 2: Statistics for the HASOC dataset

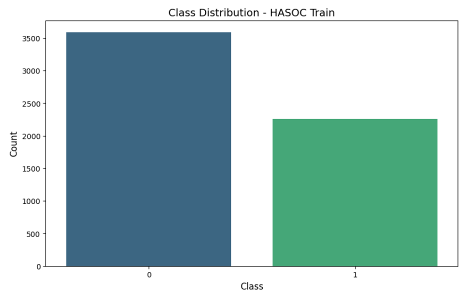


Figure 2: HASOC Train Set Distribution

#### 1.2.1 Creation and Source

The HASOC dataset was developed to support research on hate speech and offensive content across multiple languages, specifically targeting Indo-European languages: English, Hindi, and German. Data collection involved both Twitter and Facebook to ensure representation from diverse social media platforms. The dataset was created using a combination of hashtag and keyword-based crawling to gather content containing offensive language, supplemented by additional sampling of user posts to enhance diversity and reduce bias. The dataset's primary purpose is to provide a benchmark for the automatic identification of hate speech and offensive language, facilitating the development of robust multilingual classification models.

In the HASOC 2019 shared task, participants received datasets for three subtasks: (1) a binary classification task distinguishing between offensive (HOF) and non-offensive (NOT) content, (2) a fine-grained classification of offensive content into hate speech, offensive, and profane categories, and (3) identification of the target of the offense, available for English and Hindi. This study focuses on the binary classification (subtask A), aligning with the OLID dataset.

## 2 Methods

This section outlines the models employed in our experiments, detailing their architectures and relevant characteristics. The methods are categorized into transformer-based models and traditional neural network architectures, each with distinct advantages for the task of hate speech detection.

### 2.1 Transformer-Based Models

Transformer-based models have revolutionized natural language processing by leveraging self-attention mechanisms to capture contextual information effectively. The following models were utilized in this study:

#### 2.1.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) [1] was one of the primary models used. It uses a bidirectional approach to learn language representations by pre-training on large corpora and then fine-tuning for downstream tasks like classification.

#### 2.1.2 RoBERTa

RoBERTa [2] is an optimized version of BERT, trained with longer sequences and larger batch sizes. It is recognized for its strong performance across a variety of NLP tasks, and its enhancements over BERT make it a compelling choise for our classification objectives.

#### 2.1.3 HateBERT

HateBERT [5] is a BERT model fine-tuned on datasets containing hate speech and offensive language. This model is particularly effective for detecting hate speech and offensive content, providing improved accuracy in this domain.

#### 2.1.4 DistilBERT

DistilBERT [6] is a smaller, distilled version of BERT that retains 97% of BERT's performance while offering efficiency in terms of speed and memory usage. It is ideal for scenarios where computational resources are limited, making it a practical option for rapid experimentation.

#### 2.1.5 XLNet

XLNet [7] introduces a permutation-based modeling approach that captures both bidirectional context and autoregressive properties. This dual capability aids its performance in understanding nuanced text relationships, which is beneficial for tasks involving hate speech detection.

### 2.2 Other Models

In addition to transformer-based models, we also employed traditional neural network architectures, which have proven effective in text classification tasks.

#### 2.2.1 CNN

Convolutional Neural Networks (CNNs) have traditionally been used in image processing tasks but have shown remarkable potential in text classification, including offensive language detection. CNNs are capable of capturing local patterns within text by applying convolutional filters over word embeddings, identifying important n-grams or combinations of words that indicate offensive content.

Furthermore, CNNs offer computational advantage, as they are faster to train compared to transformer-based models, making them particularly useful in scenarios where computational resources are limited or where rapid experimentation is required. However, they might struggle with tasks that require more complex, long-term dependencies in language, which transformer models excel at.

#### 2.2.2 LSTM

Long Short-Term Memory (LSTM) networks [9] are a type of Recurrent Neural Network (RNN) that are well-suited for sequential data, such as text. LSTMs are designed to handle long-range dependencies by mitigating the vanishing gradient problem that traditional RNNs face. Each LSTM cell contains gates to control the flow of information, allowing it to retain or forget specific aspects of the input data.

LSTMs have been widely used in text classification tasks, including offensive language detection, due to their ability to capture context over long sequences. However, compared to transformer-based models, LSTMs can be slower to train and often require more tuning to achieve competitive results.

#### 2.2.3 BiLSTM

Bidirectional LSTM (BiLSTM) [10] extends LSTM by processing the input sequence in both forward and backward directions. This allows the model to capture context from both past and future words in a sentence, which can be particularly useful for tasks requiring a more nuanced understanding of the text.

For offensive language detection, BiLSTMs are advantageous as they can grasp not only the preceding words but also the succeeding words, leading to a more comprehensive understanding of the sentence. Like LSTMs, however, BiLSTMs may struggle with efficiency and scalability compared to transformer models.

## 3 Experimental Setup

In this section, we shed light on the experimental setup for our classification models, including the architecture of the models used, the training and evaluation procedures, and the hyperparameter configurations for both transformer-based and traditional neural network models.

### 3.1 In-Domain Experiments

In the in-domain experiments, we fine-tuned all models on the OLID training set and evaluated them on the OLID test set. We experimented with different hyperparameter settings, including varying learning rates (1e-5, 2e-5, and 3e-5), batch sizes (16 and 32), and warmup steps (500 and 1000). The evaluation metrics include accuracy, F1 score, MCC, AUROC, and AUPRC.

## 3.2 Cross-Domain Experiments

For cross-domain experiments, we trained the models on the HASOC dataset and evaluated them on the OLID test set. This setup simulates a real-world scenario where models trained on one domain (hate speech) are applied to a different domain (offensive language in tweets). We applied the same hyperparameter variations as in the in-domain experiments to ensure consistency.

## 3.3 Transformer Based Models

### 3.3.1 Model Architecture

All the transformer-based models used in our experiments share a similar architecture, consisting of the following key components:

- **Embedding Layer**:The input text is tokenized and converted into dense embeddings. BERT-based models utilize WordPiece tokenization, while XLNet employs a SentencePiece tokenizer to facilitate this process.
- **Multi-Head Self-Attention**: This core component of the transformer architecture allows the model to attend to various parts of the input text simultaneously, learning relationships between tokens irrespective of their distance. The mechanism is bidirectional in BERT-based models and autoregressive in XLNet, which captures both forward and backward dependencies through permutation-based modeling.
- **Feed-Forward Layers**: Following the self-attention mechanism, the output is passed through feed-forward layers, which assist in transforming the embeddings and learning complex representations.
- **Pooling and Classification Layer**: For classification tasks, the final hidden states from the last transformer layer are typically pooled (using the representation of the [CLS] token for BERT-like models) and then passed through a fully connected layer to predict the output label (offensive or non-offensive).

### 3.3.2 Training and Evaluation Procedure

We fine-tuned the pre-trained transformer models on task-specific datasets: *OLID* for in-domain experiments and *HASOC* for cross-domain experiments. The fine-tuning process utilized *binary cross-entropy loss* as the objective function, employing the *AdamW optimizer* due to its proven effectiveness in large-scale text classification tasks (both settings being the default in the Simple-Transformers library). The models were trained for 5 epochs, and performance was evaluated on the OLID test dataset. The evaluation metrics included *Accuracy*, *F1-Score*, *Matthews Correlation Coefficient (MCC)*, *Area Under the Receiver Operating Characteristic (AUROC)*, and *Area Under the Precision-Recall Curve (AUPRC)* to comprehensively assess the models' effectiveness.

#### 3.3.3 Hyperparameters

We explored multiple sets of hyperparameters for each model to determine the optimal configuration. The primary hyperparameters tested include:

- **Learning Rate**: We evaluated learning rates of 2e-5, 3e-5, and 1e-5, which are standard for fine-tuning transformer-based models.
- **Batch Size**: We used batch sizes of 16 and 32 were used during training. Larger batch sizes help stabilize the gradients, especially when fine-tuning large models like BERT and XLNet.
- **Warmup Steps**: We tested 500 and 1000 warmup steps to allow the learning rate to gradually increase and help the model stabilize during training.

## 3.4 CNN

### 3.4.1 Model Architecture

The CNN architecture used in our experiments consists of the following layers:

- **Embedding Layer**: Input text is transformed into dense word embeddings, allowing the network to represent each word in a continuous vector space, capturing semantic relationships between words.
- **Convolutional Layer**: A 1D convolutional layer with multiple filters is applied to the embedded text. The filters (or kernels) slide over the input text to detect key features, such as n-grams, which serve as strong indicators of offensive or non-offensive language.
- **Max-Pooling Layer**: Following the convolutional layer, max-pooling is applied to downsample the feature maps, keeping only the most significant features. This layer reduces data dimensionality while retaining important information for classification.
- **Fully Connected Layer**: The pooled features are flattened and passed through a fully connected dense layer, serving as the classifier. This layer assigns a probability to each class (offensive or non-offensive) based on the extracted features.
- **Output Layer**: A sigmoid activation function is applied at the output layer for binary classification, interpreting the output as the probability of the input text being offensive.

### 3.4.2 Training Procedure

The CNN was trained using *binary cross-entropy loss* as the objective function, with *Adam* as the optimizer, due to its adaptive learning rate capabilities. The model was trained for a fixed number of epochs. We used a 90-10 split for training and validation during both in-domain and cross-domain experiments.

### 3.4.3 Hyperparameters

The primary hyperparameters configured for the CNN model include:

- **Filter Sizes**: 128 filters of size 5 were utilized.
- **Dropout**: A dropout rate of 0.5 was implemented to prevent overfitting by randomly setting a fraction of the input units to zero during training.
- **Batch Size**: 32
- **Epochs**: 5 epochs with early stopping based on the validation performance was configured.

CNNs are particularly well-suited for text classification tasks where local word patterns (e.g., specific offensive phrases) are important. However, they might not capture long-range dependencies as effectively as transformer models. Despite its limitations, CNNs offer a computationally efficient approach and can be trained relatively quickly compared to transformers, making them useful for rapid experimentation and deployment in resource-constrained environments.

## 3.5 LSTM and BiLSTM

### 3.5.1 Model Architecture

The LSTM architecture for both LSTM and BiLSTM models used in our experiments comprise of the following layers:

- **Embedding Layer**: The input text is first transformed into dense word embeddings. This layer allows the network to represent each word in a continuous vector space, capturing semantic relationships between words. Padding tokens are also included to ensure uniform sequence length during batch processing.
- **LSTM Layer**:The core of the architecture consists of one or more LSTM layers, which are capable of learning long-range dependencies in sequential data. Each LSTM cell includes input, output, and forget gates that regulate the flow of information, enabling the model to retain relevant information over long sequences while mitigating the vanishing gradient problem that traditional RNNs encounter.
- **Bidirectional LSTM Layer**: In the BiLSTM architecture, two LSTM layers are used, where one is processing the input sequence in the forward direction and the other in the backward direction. This allows the model to capture context from both past and future tokens in the input sequence, aiding the model to understand nuanced meanings within the text.
- **Fully Connected Layer**: The output from both LSTM and BiLSTM layers is flattened, and in the case of the BiLSTM both LSTM (forward and backward) outputs concatenated, and passed through a fully connected

dense layer. This layer functions as the classifier, assigning probabilities to each class (offensice or non-offensive) based on the features extracted from the preceding layers.

- **Dropout Layer**: To prevent overfitting, a dropout layer with a dropout rate of 0.3 is applied for both LSTM and BiLSTM models before the output layer, which randomly sets a fraction of the input units to zero during training.
- **Output Layer**: The output from the fully connected layer is used to assign probabilities to each class (offensive or non-offensive). A *softmax* activation function is applied for classification.

### 3.5.2 Training Procedure

The procedure for both the LSTM and BiLSTM models were identical. The models were both trained using *cross-entropy loss* as the objective function, and the *Adam* optimizer was employed due to its adaptive learning rate capabilities, which helps stabilize the training process. The training was conducted for a fixed number of epochs, during which the model's parameters were updated iteratively to minimize the loss function. A batch size of 32 was used to balance computational efficiency and stability during training.

The training procedure involved splitting the data into training and test sets, where the model was trained on the training data and evaluated on the held-out test data. The evaluation metrics included *accuracy* and *classification report* metrics (precision, recall, F1-score) to assess the model's performance in distinguishing offensive and non-offensive content.

### 3.5.3 Hyperparameters

We chose equal numbers for the hyperparameters used for the LSTM and BiLSTM model to create a fair comparison. These hyperparameters include:

- **Embedding Dimension**: 100. This dimension determines the size of the word embeddings used to represent each word in the input text.
- **Hidden Dimension**: 128. The hidden dimension defines the size of the hidden state in the LSTMs, which affects the model's capacity to learn complex patterns in the data.
- **Dropout Rate**: 0.3. A dropout rate of 0.3 was used to prevent overfitting by randomly dropping a proportion of the input units during training.
- **Batch Size**: 32. This value balances computational efficiency and ensures stable gradient updates.
- **Learning Rate**: 0.001. The learning rate controls the step size during gradient descent, with 0.001 used to ensure gradual convergence of the loss function.

- **Epochs**: 20. The model was trained for 20 epochs, providing ample opportunity for the model to learn the underlying patterns in the data while monitoring for overfitting.

These hyperparameters were chosen to balance model complexity, training stability, and performance, allowing the LSTM to effectively capture the sequential dependencies in the text for offensive language detection.

## 4 Results and Analysis

### 4.1 Analysis (Transformer Based Models)

#### 4.1.1 In-Domain Results (OLID)

The in-domain results show that transformer-based models perform well when trained and evaluated on the same domain. Among the models, HateBERT achieved the highest accuracy (84.07%) and MCC (0.6046), closely followed by RoBERTa and BERT. These models consistently performed well across different hyperparameter settings, with F1 scores ranging from 0.76 to 0.8. XLNet also performed well, with its best model achieving an accuracy of 83.84% and MCC of 0.5988.

Table 3: In-Domain Results (Trained and Evaluated on OLID)

| Model | Accuracy | F1 Score | MCC | AUROC | AUPRC |
|---|---|---|---|---|---|
| **BERT** | 0.8267 | 0.7861 | 0.5722 | 0.8738 | 0.7466 |
| **RoBERTa** | 0.8267 | 0.7882 | 0.5768 | 0.8783 | 0.7541 |
| **HateBERT** | 0.8457 | 0.8023 | 0.6046 | 0.8674 | 0.7668 |
| **DistilBERT** | 0.8174 | 0.7699 | 0.5401 | 0.8543 | 0.7349 |
| **XLNet** | 0.8384 | 0.7994 | 0.5988 | 0.8895 | 0.7950 |

In the in-domain setup, the models trained and evaluated on the OLID dataset demonstrated strong performance across all metrics. HateBERT achieved the highest accuracy (84.57%) and F1-score (0.8023), showing that models specifically fine-tuned on offensive language detection tasks generalize well within the same domain.

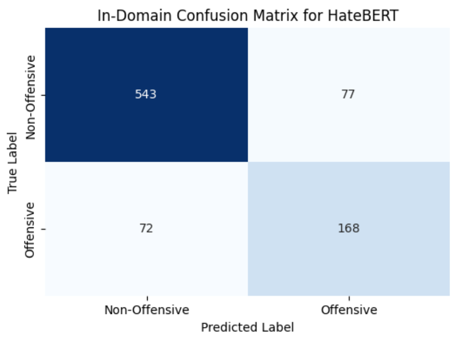


Figure 3: Confusion Matrix for HateBERT (In-Domain)

The macro-averaged precision, recall, and F1-score for the in-domain setup are as follows:

- Precision: 0.7975 (Non-Offensive), 0.7895 (Offensive)
- Recall: 0.8556 (Non-Offensive), 0.7000 (Offensive)
- F1-Score: 0.8254 (Non-Offensive), 0.7424 (Offensive)

#### 4.1.2 Cross-Domain Results (HASOC to OLID)

When trained on HASOC and evaluated on OLID, the models experienced a noticeable drop in performance. RoBERTa remained one of the best-performing models, achieving an accuracy of 79.88% and MCC of 0.4969. HateBERT and XLNet also performed reasonably well in the cross-domain setup, with XLNet's best performance achieving an accuracy of 81.05% and MCC of 0.4874. However, there was still a performance drop across models, likely due to differences in language usage and dataset characteristics between HASOC and OLID.

Table 4: Cross-Domain Results (Trained on HASOC, Evaluated on OLID)

| **Model** | **Accuracy** | **F1 Score** | **MCC** | **AUROC** | **AUPRC** |
|---|---|---|---|---|---|
| **BERT** | 0.7895 | 0.6890 | 0.4216 | 0.7681 | 0.6221 |
| **RoBERTa** | 0.7988 | 0.7484 | 0.4969 | 0.8185 | 0.6668 |
| **HateBERT** | 0.7837 | 0.7263 | 0.4531 | 0.7913 | 0.6730 |
| **DistilBERT** | 0.7965 | 0.7257 | 0.4619 | 0.7861 | 0.6674 |
| **XLNet** | 0.8105 | 0.7584 | 0.5168 | 0.8258 | 0.6895 |

In the cross-domain setup, where the models were fine-tuned on HASOC and evaluated on OLID, the overall performance dropped. The best-performing model in this setup was XLNet, achieving an accuracy of 81.05% and an F1-score of 0.7584. While RoBERTa also performed well with an accuracy of 79.88%, its performance lagged behind its in-domain results.

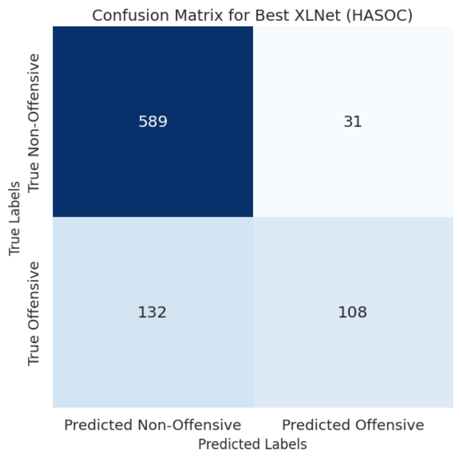


Figure 4: Confusion Matrix for XLNet (Cross-Domain)

The macro-averaged precision, recall, and F1-score for the cross-domain setup are as follows:

- Precision: 0.8073 (Non-Offensive), 0.6891 (Offensive)
- Recall: 0.8421 (Non-Offensive), 0.6662 (Offensive)
- F1-Score: 0.8238 (Non-Offensive), 0.6761 (Offensive)

#### 4.1.3 Performance Drop in Cross-Domain Setup

When comparing the in-domain and cross-domain setups, we observe a clear drop in performance. For instance, XLNet's F1-score dropped from 0.7994 in the in-domain setup to 0.7584 in the cross-domain setup, while HateBERT saw a more significant drop in F1-score, from 0.8023 to 0.7263.

The overall drop in performance across models can be quantified as follows:

- XLNet: F1-score drop of 4.10%, accuracy drop of 2.79%
- RoBERTa : F1-score drop of 4.86%, accuracy drop of 2.68%
- HateBERT : F1-score drop of 7.6%, accuracy drop of 5.7%

Despite the performance drop, XLNet showed stronger generalization capabilities, making it the most effective model in the cross-domain setup.

#### 4.1.4 Dataset Characteristics and Cross-Domain Generalization

The difference in performance between the in-domain and cross-domain setups can be attributed to several dataset characteristics. First, message length is a likely factor. Tweets in OLID are generally shorter and more direct, focusing on specific insults, whereas HASOC includes longer messages that discuss broader hate speech topics.

Additionally, topic focus varies significantly between the two datasets. OLID is more focused on political insults and harassment on social media, while HASOC includes more general hate speech and abusive language. This topical shift may have caused the models trained on HASOC to struggle with the more focused and direct offensive content in OLID.

Furthermore, linguistic style differences (such as the use of slurs, mentions, or hashtags) between the datasets could explain why the models' generalization abilities are limited.

## 4.2 Other Models

### 4.2.1 Model Performance (In-Domain vs Cross-Domain)

Table 5: Performance of CNN, LSTM, and BiLSTM Models

| Metric | Setup | CNN | LSTM | BiLSTM |
|---|---|---|---|---|
| Precision | **In-Domain** | 0.85 (0), 0.49 (1) | 0.80 (0), 0.50 (1) | 0.82 (0), 0.48 (1) |
| | **Cross-Domain** | 0.78 (0), 0.45 (1) | 0.79 (0), 0.44 (1) | 0.80 (0), 0.41 (1) |
| Recall | **In-Domain** | 0.73 (0), 0.66 (1) | 0.82 (0), 0.47 (1) | 0.76 (0), 0.58 (1) |
| | **Cross-Domain** | 0.80 (0), 0.42 (1) | 0.77 (0), 0.47 (1) | 0.70 (0), 0.54 (1) |
| F1-Score | **In-Domain** | 0.79 (0), 0.56 (1) | 0.81 (0), 0.48 (1) | 0.79 (0), 0.53 (1) |
| | **Cross-Domain** | 0.79 (0), 0.44 (1) | 0.78 (0), 0.46 (1) | 0.75 (0), 0.47 (1) |
| Macro Avg | **In-Domain** | 0.67 | 0.65 | 0.66 |
| | **Cross-Domain** | 0.61 | 0.62 | 0.61 |
| Accuracy | **In-Domain** | 0.71 | 0.72 | 0.71 |
| | **Cross-Domain** | 0.70 | 0.68 | 0.66 |

### 4.2.2 Which Approach Performs Best in the In-Domain Setup?

In the in-domain setup, all three models (CNN, LSTM, and BiLSTM) demonstrated similar performance in terms of overall accuracy, but there were notable differences in the metrics across the two classes:

- **Macro-Averaged F1-Score:** - CNN: 0.67 - LSTM: 0.65 - BiLSTM: 0.66
- **Accuracy:** - CNN: 0.71 - LSTM: 0.72 - BiLSTM: 0.71

The LSTM model achieved slightly higher accuracy (0.72) compared to CNN and BiLSTM (both at 0.71), suggesting that LSTM's sequential modeling capability provides an edge when capturing the dependencies in the OLID dataset. However, the differences are not very large, and each model has strengths depending on the specific class.

**Class-Specific Analysis:**

**Precision and Recall for Class 0 (Non-Offensive):** In the in-domain setup, CNN had the highest precision (0.85), but LSTM had the highest recall (0.82), indicating that the LSTM was better at identifying true positives for

Class 0. BiLSTM also performed well, balancing between CNN and LSTM in terms of precision (0.82) and recall (0.76). **Precision and Recall for Class 1 (Offensive):** For the offensive class, CNN achieved the highest recall (0.66), while LSTM and BiLSTM achieved higher precision values (0.50 and 0.48, respectively). This indicates that CNN is more aggressive in identifying offensive content but may produce more false positives. Overall, LSTM and BiLSTM exhibited better performance when it came to accurately identifying offensive content, as seen from their balanced F1-scores for Class 1 (0.48 for LSTM and 0.53 for BiLSTM). The BiLSTM's bidirectional nature allows it to capture contextual information from both past and future tokens, which helps in understanding subtle nuances that are often present in offensive content.

#### 4.2.3 Why Do LSTM and BiLSTM Perform Well in the In-Domain Setup?

LSTM and BiLSTM models perform well in the in-domain setup because of their ability to model long-range dependencies in text. Tweets often have specific word order patterns that convey offensive meaning, and these models are particularly good at learning and remembering such patterns:

- **Long-Term Dependencies:** LSTM and BiLSTM models are capable of learning long-term dependencies in sequences, which is crucial in detecting offensive language that may depend on the context provided by multiple preceding words.
- **Bidirectional Context:** BiLSTM's use of both forward and backward contexts allows it to better interpret the relationships between words in a sentence, thus making it effective in detecting offensive content where the meaning is determined by a combination of words before and after the target word.
- **Reduced Variability in the Dataset:** Since both models were trained and evaluated on the same dataset (OLID), they could effectively learn the specific characteristics of offensive language used in tweets, thus achieving better precision and recall.

#### 4.2.4 Does the Best Model in the In-Domain Setup Also Provide the Best Results in the Cross-Domain Setup?

No, the model that performs best in the in-domain setup does not necessarily provide the best results in the cross-domain setup. When moving to a cross-domain scenario (training on HASOC and testing on OLID), we see performance drops across all models. This suggests that while these models are able to generalize within a single dataset, they struggle to adapt to different datasets with variations in language, platform, and content.

In the cross-domain setup: **CNN:** The performance of CNN drops in terms of both precision and recall, especially for Class 1. The F1-score for offensive content dropped from 0.56 (in-domain) to 0.44 (cross-domain). This suggests that CNN's reliance on local word patterns makes it less capable of handling different types of offensive language present in HASOC versus OLID. **LSTM:** LSTM performed slightly better than CNN in the cross-domain setup, achieving a macro-averaged F1-score of 0.62. This indicates that its sequential modeling provides some generalization, but not enough to overcome the domain differences fully. **BiLSTM:** BiLSTM showed comparable performance to LSTM but slightly lower accuracy (0.66 vs. 0.68). The bidirectional approach may have helped capture some nuanced features of offensive language, but the overall generalization remained limited when applied across domains.

#### 4.2.5 Is There a Drop in Performance in the Cross-Domain Setup?

Yes, there is a noticeable drop in performance for all three models when comparing in-domain to cross-domain setups:

- **Macro-Averaged F1-Score:** All models saw a drop in macro-averaged F1-score, with CNN dropping from 0.67 to 0.61, LSTM from 0.65 to 0.62, and BiLSTM from 0.66 to 0.61.
- **Accuracy:** The accuracy also dropped for all models, with the LSTM seeing a decrease from 0.72 (in-domain) to 0.68 (cross-domain). BiLSTM had the largest drop, from 0.71 to 0.66.
- **F1-Score for Class 1 (Offensive):** The drop was particularly noticeable for the offensive class across all models, reflecting the difficulty in accurately identifying offensive content in a different domain.

**Quantifying the Drop:**
The largest drop in F1-score for Class 1 occurred for CNN, from 0.56 to 0.44 (a drop of 0.12 or 12LSTM's F1-score for Class 1 dropped from 0.48 to 0.46, and BiLSTM from 0.53 to 0.47, indicating that LSTM-based models were slightly more stable but still struggled with generalization.

#### 4.2.6 Potential Explanations for the Drop in Cross-Domain Performance

The drop in performance across domains can be attributed to several factors:

- **Differences in the Data:** HASOC includes data from both Twitter and Facebook, leading to variations in language styles, vocabulary, and content types that may not be well captured by models trained solely on OLID's Twitter data.
- **Context and Vocabulary Shifts:** The LSTM and BiLSTM models, while capable of learning context, may not be able to fully adapt to different linguistic styles or platforms. Tweets in OLID have unique vocabulary and context cues that differ from the mixed content in HASOC, contributing to the performance drop.

- **CNN's Limitations:** CNN, which relies heavily on local word patterns, struggles more with cross-domain tasks as the local n-grams it learns are likely to be different across datasets. This explains the larger drop in performance compared to LSTM and BiLSTM.
- **Generalization Capabilities:** LSTM and BiLSTM models are designed to capture sequential dependencies, which might provide some advantage over CNN in terms of understanding context. However, the bidirectional approach of BiLSTM did not result in significantly better cross-domain performance, suggesting that other factors such as dataset-specific biases played a bigger role.
- **Label Distribution Variance:** The difference in label distributions between HASOC and OLID datasets could have also impacted the ability of these models to generalize. A mismatch in the proportion of offensive vs. non-offensive content would lead to suboptimal decision thresholds when applied across domains.

**Summary:** Overall, LSTM and BiLSTM models showed slight advantages over CNN in terms of stability and performance across both in-domain and cross-domain setups, primarily due to their ability to learn sequential relationships. However, the domain-specific nature of the datasets used in this study limits their generalization abilities, and further fine-tuning or adaptation techniques may be required to mitigate this drop in performance.

## 4.3 Transformer versus Other Models

To evaluate the relative effectiveness of transformer-based models compared to traditional neural network architectures for hate speech detection, we selected the best-performing models from each group. The best transformer-based model was **HateBERT**, while the best traditional model was **LSTM**. Their performances were compared in both the in-domain and cross-domain setups.

### 4.3.1 In-Domain Setup

In the in-domain setup, where models were trained and evaluated on the OLID dataset, HateBERT showed a clear advantage over LSTM in terms of both accuracy and F1-score:

- **Accuracy:** - HateBERT: 84.57% - LSTM: 72%
- **Macro-Averaged F1-Score:** - HateBERT: 0.8023% - LSTM: 0.65%
- **Class-Specific Performance:** - HateBERT achieved a higher F1-score for both offensive (0.7424%) and non-offensive content (0.8254%), indicating better overall precision and recall in detecting offensive language.

The superior performance of HateBERT can be attributed to its pre-training on offensive language datasets, allowing it to better capture the nuances of hate speech and offensive content. In contrast, LSTM, while capable of modeling sequential information, lacks the contextual understanding that transformers achieve through self-attention mechanisms.

### 4.3.2 Cross-Domain Setup

In the cross-domain setup, where models were trained on the HASOC dataset and evaluated on OLID, HateBERT also outperformed LSTM:

- **Accuracy:** - HateBERT: 78.37% - LSTM: 68%
- **Macro-Averaged F1-Score:** - HateBERT: 0.7263% - LSTM: 0.62%
- **Class-Specific Performance:** - HateBERT had a higher F1-score for both offensive and non-offensive classes compared to LSTM, suggesting better adaptability to new domains.

Despite the performance drop seen in both models when transitioning from in-domain to cross-domain evaluation, HateBERT maintained a significant edge over LSTM. This resilience can be linked to the transformer model's ability to generalize across different domains, owing to its pre-training on large-scale diverse corpora. LSTM struggled more with the domain shift, as evidenced by its lower precision and recall for both classes.

### 4.3.3 Summary of Comparison

**HateBERT** demonstrated consistently superior performance over **LSTM** in both in-domain and cross-domain setups. The key reasons for this advantage include:

- **Contextual Understanding:** HateBERT's use of self-attention mechanisms allows it to capture nuanced relationships between words, which is critical for detecting offensive content that relies on context.
- **Pre-Training on Offensive Language:** HateBERT was specifically fine-tuned on offensive language datasets, making it more sensitive to the characteristics of hate speech.
- **Generalization Ability:** HateBERT demonstrated better cross-domain generalization, which is essential for real-world applications where the data might differ from the training distribution.

In contrast, **LSTM**, while effective at capturing sequential dependencies, lacks the sophisticated attention mechanism of transformers, which limits its performance in capturing the broader context required for accurate offensive language detection. Furthermore, LSTM's performance drop in the cross-domain setup suggests limited adaptability to new and diverse datasets.

In conclusion, transformer-based models, specifically HateBERT, provide a significant performance advantage over traditional neural networks such as LSTM for the task of hate speech and offensive language detection, particularly in terms of both accuracy and robustness across domains.

## References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*, 2019.

[2] Yinhan Liu, Myle Ott, Naman Goyal, et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint arXiv:1907.11692, 2019.

[3] Marcos Zampieri, et al. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of SemEval-2019*, 2019.

[4] Thomas Mandl, et al. Overview of the HASOC Track at FIRE 2019: Hate Speech and Offensive Content Identification in Indo-European Languages. In *Proceedings of FIRE 2019*, 2019.

[5] Tommaso Caselli, et al. HateBERT: Retraining BERT for Abusive Language Detection in English. arXiv preprint arXiv:2010.12472, 2020.

[6] Victor Sanh, Lysandre Debut, Julien Chaumond, Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108, 2019.

[7] Zhilin Yang, et al. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Proceedings of NeurIPS*, 2019.

[8] Ashish Vaswani, et al. Attention is All You Need. In *Proceedings of NeurIPS*, 2017.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. In *Neural Computation*, 1997.

[10] Alex Graves and Jürgen Schmidhuber. Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. In *Proceedings of IJCNN*, 2005.

## A Division of Work

This section outlines the contributions of each group member for this assignment.

- **Paul Constantinescu:** Conducted literature review, implemented and trained the Transformer models (BERT, RoBERTa, etc.).
- **Liza Darwesh:** Implemented and trained the LSTM and BiLSTM models.
- **Andres Garcia Rincon:** Focused on data exploration and analysis for LSTM and BiLSTM.
- **Mithat Can Ozgun:** Implemented and trained the CNN models.

All members contributed to the style and structure of the document.

## B Results

This section presents the performance results of the models trained and evaluated on different datasets.

### B.1 Results for Models Trained and Evaluated on OLID

This section provides the results of various transformer models trained and evaluated on the OLID dataset.

Table 6: Results for Models Trained and Evaluated on OLID

| Model | MCC | Accuracy | F1 Score | Eval Loss | AUROC | AUPRC |
|---|---|---|---|---|---|---|
| **BERT (2e-05)** | 0.5712 | 0.8244 | 0.7853 | 0.8455 | 0.8548 | 0.7235 |
| **BERT (3e-05)** | 0.5213 | 0.8035 | 0.7603 | 0.5153 | 0.8595 | 0.7298 |
| **BERT (1e-05)** | 0.5722 | 0.8267 | 0.7861 | 0.5299 | 0.8738 | 0.7466 |
| **RoBERTa (2e-05)** | 0.5768 | 0.8267 | 0.7882 | 0.6368 | 0.8783 | 0.7541 |
| **RoBERTa (3e-05)** | 0.5794 | 0.8198 | 0.7871 | 0.4490 | 0.8770 | 0.7607 |
| **RoBERTa (1e-05)** | 0.5746 | 0.8256 | 0.7870 | 0.4663 | 0.8834 | 0.7810 |
| **HateBERT (2e-05)** | 0.5814 | 0.8291 | 0.7905 | 0.7021 | 0.8469 | 0.7463 |
| **HateBERT (3e-05)** | 0.5809 | 0.8372 | 0.7887 | 0.4874 | 0.8479 | 0.7558 |
| **HateBERT (1e-05)** | 0.6046 | 0.8457 | 0.8023 | 0.4341 | 0.8674 | 0.7668 |
| **DistilBERT (2e-05)** | 0.5383 | 0.8105 | 0.7688 | 0.7169 | 0.8521 | 0.7319 |
| **DistilBERT (3e-05)** | 0.4886 | 0.7895 | 0.7439 | 0.5082 | 0.8427 | 0.6932 |
| **DistilBERT (1e-05)** | 0.5401 | 0.8174 | 0.7699 | 0.4554 | 0.8543 | 0.7349 |
| **XLNet (2e-05)** | 0.5483 | 0.8233 | 0.7732 | 0.6354 | 0.8740 | 0.7753 |
| **XLNet (3e-05)** | 0.5630 | 0.7988 | 0.7720 | 0.5041 | 0.8803 | 0.7692 |
| **XLNet (1e-05)** | 0.5988 | 0.8384 | 0.7994 | 0.3996 | 0.8895 | 0.7950 |

## B.2 Cross-Domain Results for Models Trained on HASOC and Evaluated on OLID

This section provides the cross-domain evaluation of models trained on HASOC and evaluated on OLID.

Table 7: Results for Models Trained on HASOC and Evaluated on OLID (Cross-Domain)

| Model | MCC | Accuracy | F1 Score | Eval Loss | AUROC | AUPRC |
|---|---|---|---|---|---|---|
| **BERT (2e-05)** | 0.4386 | 0.7791 | 0.7189 | 0.7044 | 0.7801 | 0.6073 |
| **BERT (3e-05)** | 0.4216 | 0.7895 | 0.6890 | 0.5298 | 0.7681 | 0.6221 |
| **BERT (1e-05)** | 0.4370 | 0.7849 | 0.7154 | 0.4991 | 0.7910 | 0.6306 |
| **RoBERTa (2e-05)** | 0.4526 | 0.7744 | 0.7258 | 0.5881 | 0.7965 | 0.6537 |
| **RoBERTa (3e-05)** | 0.3922 | 0.7814 | 0.6320 | 0.4703 | 0.8195 | 0.6819 |
| **RoBERTa (1e-05)** | 0.4969 | 0.7988 | 0.7484 | 0.4822 | 0.8185 | 0.6668 |
| **HateBERT (2e-05)** | 0.4552 | 0.7791 | 0.7275 | 0.6126 | 0.7841 | 0.6639 |
| **HateBERT (3e-05)** | 0.3771 | 0.7674 | 0.6819 | 0.5810 | 0.7245 | 0.5966 |
| **HateBERT (1e-05)** | 0.4531 | 0.7837 | 0.7263 | 0.4812 | 0.7913 | 0.6730 |
| **DistilBERT (2e-05)** | 0.4351 | 0.7721 | 0.7175 | 0.6177 | 0.7841 | 0.6354 |
| **DistilBERT (3e-05)** | 0.3606 | 0.7070 | 0.6705 | 0.6050 | 0.7653 | 0.6248 |
| **DistilBERT (1e-05)** | 0.4619 | 0.7965 | 0.7257 | 0.4773 | 0.7861 | 0.6674 |
| **XLNet (2e-05)** | 0.4629 | 0.7860 | 0.7314 | 0.5478 | 0.8035 | 0.6774 |
| **XLNet (3e-05)** | 0.4874 | 0.8105 | 0.7242 | 0.4497 | 0.8242 | 0.6817 |
| **XLNet (1e-05)** | 0.5168 | 0.8058 | 0.7584 | 0.4671 | 0.8258 | 0.6895 |