Supervised Classification and Unsupervised Clustering for Multi-label Toxic Comment Detection

Any Das (922710) & Aarohi Rakeshkumar Mistry (925352)
CdLM Data Science, Università degli Studi di Milano-Bicocca

Abstract
This study addresses the challenge of multi-label toxic comment detection by evaluating three computational paradigms: Statistical (TF-IDF), Deep Learning (Text CNN), and Transformer-based (DistilBERT) models. Using the Jigsaw dataset, we demonstrate that DistilBERT, optimized with Focal Loss to mitigate severe class imbalance, achieves state-of-the-art performance (Macro ROC-AUC: 0.9858, Macro F1: 0.62), significantly outperforming static baselines in identifying context-dependent toxicity. Furthermore, our unsupervised analysis reveals that Semantic Clustering (using SBERT and UMAP-reduced embeddings with HDBSCAN) effectively isolates outliers and uncovers high-purity toxic micro-communities (e.g., targeted identity hate groups), offering critical insights missed by traditional K-Means clustering. We conclude that a hybrid approach combining contextual classification for precision and density-based clustering for pattern discovery provides a robust solution for modern content moderation.

Keywords
Multi-label Classification, Toxicity Detection, Class Imbalance, Focal Loss, DistilBERT, Text CNN, TF-IDF, Semantic Clustering, SBERT, HDBSCAN, UMAP.

TABLE OF CONTENTS

1. INTRODUCTION

The rapid expansion of global digital discourse faces a severe threat from the exponential growth of toxic commentary, including hate speech, harassment, and personal attacks. Given the massive volume of user-generated content, reliance on manual moderation has become impractical. Furthermore, contemporary toxic language is often cloaked in irony, evolving slang, and implicit contextual references, rendering outdated keyword-based filters ineffective [1]. Consequently, there is a critical need for analytical systems capable of detecting nuanced forms of abuse that evade traditional detection methods.

This project addresses this challenge by implementing a comprehensive, multi-faceted machine learning framework focused on multi-label toxic comment detection. The analysis employs a novel dual-strategy approach that integrates two primary analytical goals: Supervised Classification for accurate, immediate categorization, and Unsupervised Clustering for discovering emerging, policy-evading toxicity trends. This strategic combination ensures both the precision required for real-time moderation and the adaptability needed to proactively combat evolving forms of digital abuse.

Our experimental design involves a systematic evaluation of three distinct computational paradigms:

1. **A Traditional Statistical Model:** TF-IDF combined with LinearSVC and Ridge Classifier.

2. **A Deep Learning Baseline:** Static Word Embeddings via FastText-TextCNN.
3. **A State-of-the-Art Benchmark:** Dynamic Contextual Embeddings via DistilBERT and Sentence-BERT (SBERT).

The central objective is to quantify the performance benefits achieved by transitioning to Transformer-based models, specifically identifying the "contextual premium" in detection accuracy. All experiments are conducted using the Jigsaw Toxic Comments Classification dataset, ensuring robust real-world applicability and comparative validation with existing literature [2].

2. DATASET DESCRIPTION

The study utilizes the Jigsaw Toxic Comment Classification Dataset, a widely recognized benchmark in the field of automated toxicity detection, originally released during a public Kaggle competition [2]. This corpus is particularly valuable for modeling real-world digital discourse as it comprises a large-scale collection of Wikipedia talk page edits, providing a realistic representation of online user interactions.

The dataset consists of 159,571 English comments in the training set and 153,164 comments in the evaluation set. A defining characteristic of this dataset is its multi-label structure, where each text entry is annotated with binary indicators across six distinct toxicity categories: *toxic*, *severe_toxic*, *obscene*, *threat*, *insult*, and *identity_hate*. Unlike simple binary classification tasks, this schema allows a single comment to be classified into multiple categories simultaneously (e.g., a comment can be labeled as both toxic and insult), necessitating models capable of capturing complex, overlapping semantic dependencies.

The initial data structure includes the raw *comment_text* alongside the six target labels, as illustrated in the data sample below. This configuration serves as the foundational input for both the supervised learning and unsupervised clustering pipelines developed in this research.

```
Training Data Head:
                 id                                       comment_text  toxic  \
0  0000997932d777bf  Explanation\nWhy the edits made under my usern...      0
1  000103f0d9cfb60f  D'aww! He matches this background colour I'm s...      0
2  000113f07ec002fd  Hey man, I'm really not trying to edit war. It...      0
3  0001b41b1c6bb37e  "\nMore\nI can't make any real suggestions on ...      0
4  0001d958c54c6e35  You, sir, are my hero. Any chance you remember...      0

   severe_toxic  obscene  threat  insult  identity_hate
0             0        0       0       0              0
1             0        0       0       0              0
2             0        0       0       0              0
3             0        0       0       0              0
4             0        0       0       0              0
```

Figure 1: Dataset Structure and Initial Data Head

3. TEXT PRE-PROCESSING

Before applying pipeline-specific transformations, a standardized data integrity phase was executed to ensure consistency across all experimental setups. The evaluation dataset was constructed by merging the provided test.csv with test_labels.csv. Crucially, rows containing -1 in any toxicity label (indicating unscored entries) were removed. This filtration reduced the raw test set from 153,164 to 63,978 valid samples, establishing a reliable baseline for performance evaluation.

Given the distinct architectural requirements of statistical versus neural models, we implemented a bifurcated pre-processing strategy tailored to each paradigm.

3.1 TF-IDF Pre-processing

For the statistical baseline, we employed the Term Frequency-Inverse Document Frequency (TF-IDF) weighting scheme [3]. After initial cleaning, the dataset underwent specific quality-control steps to enhance feature density. This included the elimination of comments with fewer than two words (4,720 from training; 3,008 from test) as they lack sufficient semantic context, and the removal of duplicate cleaned comments (1,150 from training; 275 from test) to prevent model bias. The resulting dataset dimensions for this pipeline were 153,701 training samples and 60,695 test samples.

The consistent application of this pipeline resulted in a high-quality, non-redundant corpus as summarized in Table 1 below:

Table 1: TF-IDF Dataset Overview Before and After Cleaning

Dataset Phase	Original Count	Final Count	Data Loss
Training Set	159,571	153,701	5,870 (3.68%)
Test Set	63,978	60,695	3,283 (5.13%)

Using the NLTK (Natural Language Toolkit) and Scikit-learn, we executed a sequential normalization pipeline designed for the TF-IDF framework. First, non-textual symbols (emojis) were eliminated, followed by lowercasing and the removal of punctuation, special characters, and URLs to reduce noise. The text was then tokenized into individual word units. To filter out low-information terms, common English stop-words were removed. Finally, Part-of-Speech (POS) aware lemmatization was applied to reduce tokens to their base forms, improving lexical consistency while preserving important distinctions between word forms.

A preliminary analysis of basic text features, including *comment_length* and *word_count*, revealed substantial variation typical of user-generated text. Comment lengths ranged from as short as 1-6 characters to a maximum of 5,000 characters. Such variability

underscores the critical importance of robust preprocessing and feature extraction to ensure reliable model performance across diverse input lengths.

Analysis of target labels revealed substantial class imbalance, with several toxicity categories representing less than 1% of the data, as detailed in Table 2.

Table 2: Toxicity Label Distribution in Training Data

Toxicity Label	Count	Percentage (%)
toxic	14,631	9.52
obscene	8,065	5.25
insult	7,536	4.90
severe_toxic	1,524	0.99
identity_hate	1,345	0.88
threat	452	0.29

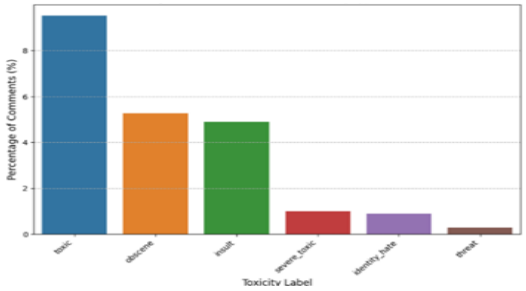


Figure 2: Percentage of Comments Labeled per Toxicity Type

(a) (b)

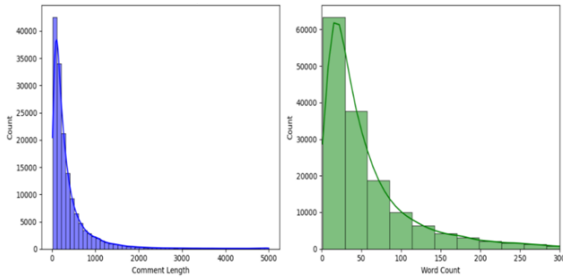


Figure 3: Distribution of Comment Length (a) and Word Count (b)

3.2 Neural Embedding Pre-processing

The pre-processing pipeline for neural embedding models was designed to preserve semantic richness and syntactic structure. This strategy supports static embeddings like FastText [4] used in TextCNN architectures [5], as well as contextual transformers like DistilBERT [6] and Sentence-BERT (SBERT) [7]. This "minimal intervention" approach contrasts with the aggressive normalization used in statistical NLP, ensuring that the linguistic fidelity required for deep learning is maintained.

To prepare the text for embedding layers without stripping context, we applied a focused cleaning strategy. All text was converted to lowercase for uniform representation. A custom bert_minimal_clean utility was then used to remove specific non-linguistic elements that degrade embedding quality, such as URLs, HTML tags, emojis, user mentions, and redundant whitespace. This ensures stable input for sub-word tokenizers without stripping meaningful content. Any comment collapsing into an empty string after cleaning was discarded. Regarding duplicates, we applied a label-aware filtering strategy: duplicates were preserved only if they carried different label combinations to capture annotation variations.

Crucially, for neural embedding models, we retained the full linguistic structure of each comment. Unlike the TF-IDF based Bag-of-Words approach, we did not remove stop-words or apply stemming/lemmatization. Stop-words act as grammatical anchors that help transformer models capture relationships between words, while keeping original word forms ensures compatibility with both FastText's vector lookup and BERT-style sub-word tokenization.

The consistent application of this pipeline resulted in minimal data loss, ensuring a high-quality, non-redundant corpus as shown in Table 3 below:

Table 3: Neural Embedding Dataset Overview Before and After Cleaning

Dataset Phase	Original Count	Final Count	Data Loss
Training Set	159,571	159,330	241 (0.15%)
Test Set	63,978	63,927	51 (0.08%)

A 90:10 training-validation split was subsequently applied, resulting in 143,397 training and 15,933 validation samples.

4. TEXT REPRESENTATION

After pre-processing, the textual data was converted into numerical vectors tailored to the specific requirements of each modeling paradigm. We employed three distinct approaches: Statistical, Static Neural, and Contextual Neural paradigms.

4.1 Bag-of-Words Representation (TF-IDF)

We utilized the Term Frequency-Inverse Document Frequency (TF-IDF) weighting scheme as our traditional baseline [3]. This approach transforms text into a sparse matrix where term weights reflect their importance relative to the entire corpus. Specifically, the vectorizer assigns high weights to terms frequent in a specific comment but rare across the dataset (e.g., specific slurs), while penalizing common words.

To capture the broad vocabulary of user-generated content while managing memory, we limited the maximum features to 100,000. Crucially, we employed an N-gram range of (1, 2). The inclusion of bigrams was essential for capturing toxic phrases (e.g., "shut up") that

simple unigrams might miss. Stop-words were explicitly excluded as they were already removed during pre-processing. The TF-IDF approach was selected to provide a strong, interpretable baseline that effectively balances classification performance with computational speed, making it highly compatible with traditional algorithms like LinearSVC.

To ensure rigorous experimental integrity and prevent data leakage, vectorization was executed in two strict phases. First, the vectorizer was fitted only on the training data (fit_transform) to learn the vocabulary and weight distributions. Second, this pre-fitted vectorizer was applied to the test data (transform) to ensure feature consistency across datasets without exposing the model to unseen test information.

4.2 Static Word Embeddings (FastText)

To overcome the sparsity limitations of TF-IDF, we employed FastText static embeddings [4]. Unlike word-level models (e.g., Word2Vec), FastText represents words as a bag of character n-grams. This allows the model to construct vectors for unseen words by aggregating the vectors of their constituent n-grams, providing a dense, continuous vector space that captures deeper morphological relationships.

Each word was represented as a dense 300-dimensional vector. We standardized the sequence length to 100 tokens (MAX_LEN_CNN = 100). During the training of our CNN architecture, the embedding layer was set to 'trainable', allowing the pre-trained vectors to be fine-tuned specifically for the toxic domain. FastText is uniquely suited for toxic comment analysis because of its robust handling of Out-of-Vocabulary (OOV) terms. It effectively generates embeddings for misspellings, slang, and intentional obfuscations (e.g., "f*ck") commonly found in abusive online behavior, which traditional models often miss.

4.3 Contextual Embeddings (Transformers)

For nuanced understanding, we utilized Transformer-based models that generate contextual embeddings. Unlike static models where a word's vector is fixed, Transformers dynamically adjust the embedding based on surrounding words (e.g., distinguishing "kill the process" from "kill you"). We implemented a dual-model strategy optimized for different tasks:

Classification: DistilBERT For the classification task, we employed the DistilBERT architecture. Using WordPiece tokenization, it generates 768-dimensional contextual vectors. We utilized the [CLS] token embedding as the aggregate semantic representation of the entire comment [6]. The maximum sequence length was standardized to 256 tokens to cover the majority of comments without excessive padding. We selected distilbert-base-uncased for its optimal balance of accuracy and efficiency; it retains 97% of BERT's performance while being 40% smaller and 60% faster, making it ideal for fine-tuning on large-scale datasets.

Clustering: Sentence-BERT (SBERT) & UMAP For unsupervised clustering, standard BERT embeddings often yield poor performance due to the anisotropy of the vector space. Therefore, we utilized Sentence-BERT (SBERT), specifically the all-MiniLM-L6-v2 model [7]. This model uses mean-pooling to create fixed-length, 384-dimensional sentence vectors optimized for cosine similarity, ensuring that comments are grouped based on semantic intent rather than just keyword overlap.

To prepare these dense vectors for downstream clustering, we applied UMAP (Uniform Manifold Approximation and Projection) [8]. This step reduced the 384-dimensional embeddings to 5 dimensions. UMAP was chosen to preserve the non-linear semantic relationships of the high-dimensional space while ensuring that algorithms like K-Means and HDBSCAN can identify meaningful, well-separated groups.

Table 4 summarizes the technical specifications of the vectorization techniques employed in this study.

Table 4: Comparison of Text Representation Techniques

Model (Type)	Dim.	Primary Advantage
TF-IDF (Sparse)	100k	Computational Speed & Interpretability
FastText (Static)	300	Robust handling of OOV & Slang
DistilBERT (Context)	768	Deep Contextual Awareness
SBERT (Sentence)	384	Semantic Similarity for Clustering

5. CLASSIFICATION IMPLEMENTATION

The text classification phase involved training and evaluating models across three paradigms to identify the most effective approach for multi-label toxicity detection. This section details the implementation of the statistical, deep learning, and transformer-based models.

5.1 TF-IDF Workflow (Baseline)

The baseline models utilize traditional machine learning algorithms trained on the sparse TF-IDF vectors described in Section 4.1. These models serve as a benchmark for computational efficiency and interpretability. To handle the multi-label nature of the dataset, we employed a One-vs-Rest strategy, where a separate binary classifier is trained for each toxicity category. All implementations utilized the Scikit-learn framework [9].

Ridge Classifier (One-vs-Rest). The Ridge Classifier was selected for its ability to handle multicollinearity in high-dimensional text data through L_2 regularization. The model was initialized with a regularization strength of $\alpha = 1.0$. We utilized the LSQR solver (*solver='lsqr'*), which is specifically optimized for large sparse matrices. To counteract the dominance of non-toxic comments, the *class_weight='balanced'* parameter was set, automatically adjusting weights inversely proportional to class frequencies.

Instead of relying on standard binary predictions, we extracted raw confidence scores using the *decision_function*. These scores were then passed through an adaptive thresholding loop to maximize the F1-score for each label independently.

Linear Support Vector Classifier (LinearSVC). The LinearSVC model utilizes a linear kernel Support Vector Machine, widely regarded as a robust baseline for text classification tasks [10]. The regularization parameter was tuned to *C=0.5* to prevent overfitting on the noisy user-generated text. Crucially, we set *dual=False* to optimize the algorithm for our dataset where the number of samples exceeds the number of features (n_samples > n_features), ensuring faster convergence. Similar to the Ridge model, *class_weight='balanced'* was applied, and the model was trained for a maximum of 1,000 iterations (*max_iter=1000*) to guarantee stability.

Predictions were generated by optimizing decision thresholds (ranging from -5 to 5 in the decision space) using the Nelder-Mead optimization algorithm. This ensured that the final binary labels were calibrated to prioritize the F1-score over simple accuracy.

5.2 Text CNN Workflow (Deep Learning)

We established a deep learning baseline using a Text CNN (Convolutional Neural Network for Text) architecture, enhanced with FastText static embeddings [4]. This approach was selected to leverage dense, semantic word representations while maintaining computational efficiency compared to recurrent models.

To optimize the input for the CNN, a meticulous data preparation pipeline was executed. First, a custom vocabulary of 50,001 tokens was built from the training corpus, reserving special tokens for padding and unknown words. We then loaded 300-dimensional FastText vectors (crawl-300d-2M.vec). To minimize information loss from Out-of-Vocabulary (OOV) terms, a case-insensitive lookup strategy was implemented. This process successfully matched 32,362 words directly and recovered an additional 605 terms through casing variations, achieving an overall vocabulary coverage of 64.7%. The remaining 17,638 OOV words were initialized with random vectors to be learned during training. Finally, all comments were padded or truncated to a fixed length of 100 tokens *(MAX_LEN_CNN = 100)* to ensure uniform tensor dimensions for batch processing [5].

The model architecture was specifically designed to capture local n-gram patterns indicative of toxicity. The core consists of three parallel 1D convolutional layers with kernel sizes of 3, 4, and 5, corresponding to trigrams, 4-grams, and 5-grams, respectively. Each kernel utilized 128 filters, allowing the model to learn a diverse set of features for each n-gram size. The output of each layer was passed through a Global Max Pooling layer to extract the most salient features (highest activation) across the sentence, rendering the model robust to the specific position of toxic keywords. These pooled features were concatenated into a single 384-dimensional vector, followed by a Dropout layer (p=0.5) to mitigate overfitting before the final fully connected classification layer.

To address the severe class imbalance inherent in the dataset, we utilized Focal Loss *(α=0.25, γ=2)* of standard Cross-Entropy [11]. This loss function dynamically down-weights easy examples (non-toxic comments) and forces the model to focus on hard-to-classify, minority toxic samples. The model was trained using the Adam optimizer [12] with a learning rate of 0.001 for 2 epochs. Post-training, we implemented a fine-grained threshold optimization strategy on the validation set. Unlike standard coarse tuning, we evaluated thresholds ranging from 0.01 to 0.95 with a step size of 0.01 to precisely calibrate the decision boundary for each of the six toxicity labels independently. The final model comprises approximately 15.5 million parameters.

5.3 DistilBERT Workflow (State-of-the-Art)

For the state-of-the-art benchmark, we fine-tuned DistilBERT, a distilled and lighter version of the BERT transformer. Unlike the Text CNN baseline which relies on local n-gram features, DistilBERT employs a multi-head self-attention mechanism. This allows the model to understand the full context of a word based on its surroundings, capturing bidirectional dependencies and subtle nuances essential for detecting implicit toxicity.

To optimize the input for the transformer, we utilized the DistilBertTokenizerFast, which is based on the WordPiece algorithm. This method effectively resolves the Out-of-Vocabulary (OOV) issue found in static embeddings by decomposing unknown words into subword units (e.g., "toxic" → "tox", "##ic"). Consequently, we achieved 100% vocabulary coverage without the need for random vector initialization. Input

sequences were standardized to a maximum length of 256 tokens (MAX_LEN = 256). This threshold was empirically selected to cover the majority of comment lengths while optimizing memory usage and computational speed during the fine-tuning process.

The backbone of the architecture consists of the pre-trained distilbert-base-uncased model [6], designed to extract deep contextualized features. For classification, we extracted the embedding of the special [CLS] token (index 0) from the final hidden state. This 768-dimensional vector represents the aggregate semantic meaning of the entire sequence. It is passed through a Dropout layer (p=0.1) for regularization and finally into a fully connected Linear layer to map the hidden features to the six toxicity output classes.

To address the class imbalance, we adapted the Focal Loss function with an increased focusing parameter *(α=0.25, γ=3)* [11]. The higher gamma value was chosen to further suppress the contribution of well-classified examples, compelling the model to prioritize difficult minority classes such as *'threat'* and *'identity_hate'* which static models often miss. The fine-tuning was performed using the AdamW optimizer with a learning rate of 2e-5 and a weight decay of 0.01 to prevent overfitting. A linear learning rate scheduler was applied to ensure stability. The model was trained for 3 epochs with a batch size of 32, utilizing DataParallel for multi-GPU acceleration. Post-training, we applied the same rigorous fine-grained threshold optimization (0.01 to 0.95, step 0.01) used in the CNN baseline to maximize the F1-score for each category independently.

6. CLASSIFICATION RESULTS

This section presents a detailed evaluation of the three modeling paradigms. Performance is assessed using Macro-Averaged F1-Score (to account for class imbalance) and Macro ROC-AUC (to measure ranking quality).

6.1 TF-IDF

Two statistical machine learning models, the Ridge Classifier and the Linear Support Vector Classifier (LinearSVC), were evaluated on the refined TF-IDF feature space.

Ridge Classifier. While the Ridge Classifier exhibited strong discrimination capability with a Macro ROC-AUC of 0.9553, its practical classification performance was moderate, yielding a tuned Macro F1-score of 0.4875. As detailed in Table 5, the Ridge model performed adequately on high-frequency classes like toxic and obscene but struggled significantly with sparsely represented labels such as *threat* (F1: 0.3511) and *severe_toxic* (F1: 0.3981).

LinearSVC. In contrast, the LinearSVC model demonstrated superior performance, outperforming the Ridge Classifier across all aggregate metrics. It achieved a Macro ROC-AUC of 0.9692 and a tuned Macro F1-score of 0.5483. Notably, LinearSVC showed substantial improvements in identifying key toxicity categories, including obscene (0.6926), *toxic* (0.6717), and *insult* (0.6315). Furthermore, it offered better generalization for minority classes, improving the F1-scores for *identity_hate* to 0.5015 and *threat* to 0.3896.

Table 5: Per-Label Performance of Statistical Baseline Models (TF-IDF)

Label / Metric	Ridge F1-Score	LinearSVC F1-Score
toxic	0.5855	0.6717
severe_toxic	0.3981	0.4032
obscene	0.6005	0.6926
threat	0.3511	0.3896
insult	0.5715	0.6315
identity_hate	0.4185	0.5015
Macro ROC-AUC	0.9553	0.9692
Tuned Macro F1-Score	0.4875	0.5483

Visual analysis of the decision boundaries further confirms the superiority of the LinearSVC. The confusion matrices (Figure 4 and Figure 5) demonstrate that while both models maintain high True Negative rates, LinearSVC exhibits tighter decision boundaries with fewer False Negatives for ambiguous comments.

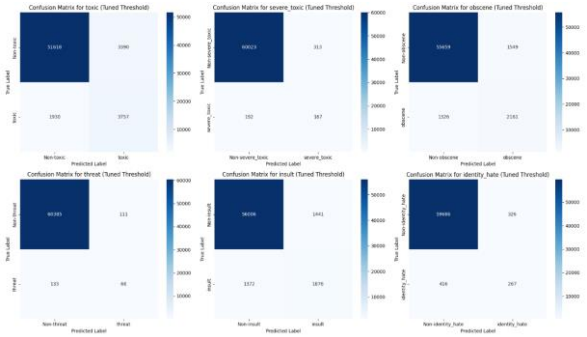


Figure 4: Confusion Matrix for Ridge Classifier

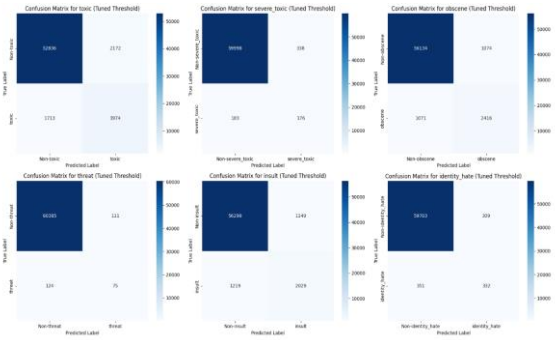
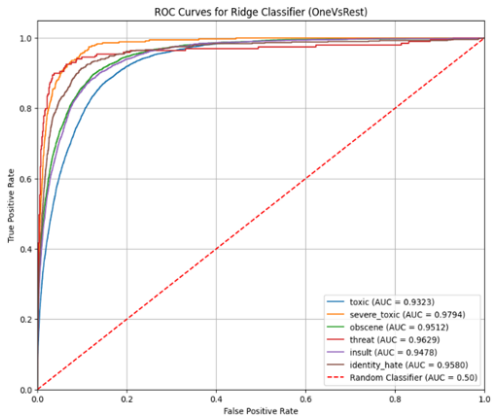


Figure 5: Confusion Matrix for LinearSVC

The ROC Curves presented in Figure 6 illustrate that separation is strongest for high-frequency labels such as *toxic* and *obscene*, where the curves push towards the top-left corner. However, rare labels like *threat* and *identity_hate* exhibit flatter curves, highlighting the inherent limitation of statistical models in capturing semantic context from sparse examples.

(a)



(b)

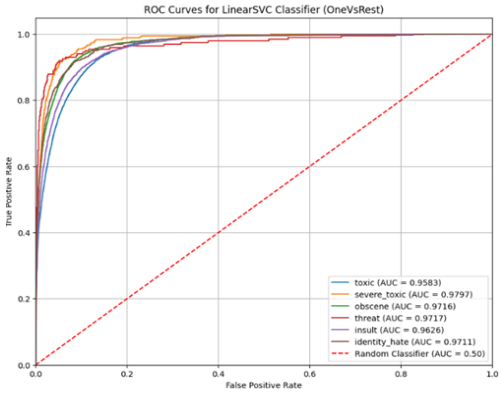


Figure 6: Multi-label ROC AUC curves for different classifiers (a) Ridge Classifier and (b) Linear SVC

Based on the combined assessment of macro-ROC-AUC, F1-scores, and per-label stability, the LinearSVC classifier is established as the robust statistical baseline for this study, effectively handling the high-dimensional feature spaces of the TF-IDF framework.

6.2 Text CNN

The Text CNN model, enhanced with FastText static word embeddings (approx. 15.5 million parameters), was evaluated on the full test set of 63,927 samples to benchmark the efficacy of static dense representations against statistical and contextual approaches.

The model demonstrated strong ranking capabilities, achieving a Macro ROC-AUC of 0.9555 and a Micro ROC-AUC of 0.9634. These scores indicate that the model effectively differentiates between toxic and non-toxic comments in a probabilistic ranking scenario. However, converting these probabilities into hard classifications proved challenging. Even after comprehensive threshold optimization on the validation set, the Macro F1-score reached 0.4764, illustrating the difficulty of classifying rare labels with static embeddings.

To address the severe class imbalance, decision thresholds were aggressively fine-tuned using a fine-grained optimization strategy (step size 0.01). As detailed in Table 6, the model required significantly lower confidence thresholds (e.g., 0.30 for identity_hate and 0.23 for threat) to trigger positive predictions for minority classes. This reflects a strategy prioritized to capture rare toxic instances at the cost of precision.

Table 6: Threshold Optimization Results on Validation Set

Toxicity Category	Optimal Threshold	Validation F1	Precision	Recall
toxic	0.35	0.7681	0.7858	0.7511
severe_toxic	0.31	0.4883	0.4052	0.6144
obscene	0.30	0.7613	0.7617	0.7608
threat	0.23	0.3333	0.3091	0.3617
insult	0.33	0.7065	0.6747	0.7414
identity_hate	0.30	0.4981	0.6132	0.4194

Table 7 presents the final performance on the test set using these optimized thresholds. The model showed improved stability across categories compared to previous iterations. Notably, performance on the most challenging minority classes saw a recovery, with *severe_toxic* reaching an F1-score of 0.3001 and *identity_hate* achieving 0.4124.

Table 7: Text CNN Performance Metrics on Test Set

Category	AUC-ROC	F1-Score	Precision	Recall	Support
toxic	0.9435	0.6190	0.5047	0.8004	6,087
severe_toxic	0.9724	0.3001	0.2021	0.5831	367
obscene	0.9499	0.6120	0.5182	0.7470	3,688
threat	0.9671	0.3233	0.2230	0.5877	211
insult	0.9510	0.5916	0.5063	0.7115	3,425
identity_hate	0.9491	0.4124	0.4221	0.4031	712
Macro Avg	0.9555	0.4764	0.3961	0.6388	-
Micro Avg	0.9634	0.5846	0.4841	0.7377	14,490

Visual analysis provides further insight into these limitations. The discrepancy between high ROC-AUC scores and moderate F1-scores highlights a distinct bias towards Recall (Macro: 0.6388) over Precision (Macro: 0.3961). This behavior is a direct result of using Focal Loss (gamma=2), which penalizes the model heavily for missing minority class examples. While this ensures that

more toxic comments are flagged (e.g., 58.31% recall for *severe_toxic*), it inevitably introduces a higher rate of false positives.

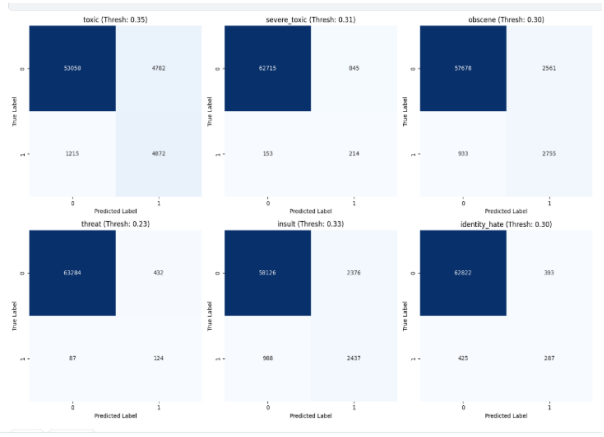


Figure 7: Confusion Matrices for Text CNN Model

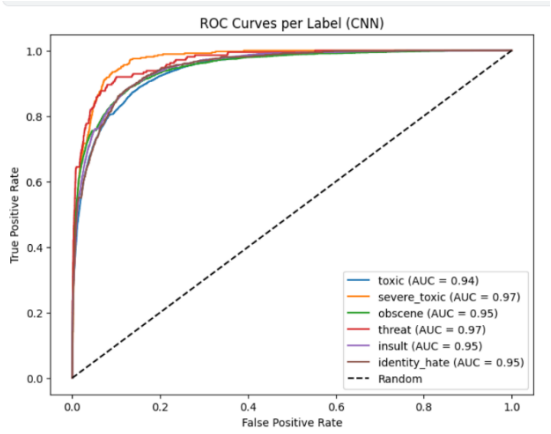


Figure 8: ROC Curves for Text CNN Model

Furthermore, the results suggest that the static nature of FastText embeddings limits the model's ability to resolve semantic ambiguity. Since static embeddings assign a fixed vector to each word regardless of context, the CNN architecture struggles to differentiate between benign and toxic uses of the same word (e.g., "killing it" in a game vs. a violent threat) without the aid of dynamic contextualization.

6.3 DistilBERT

The fine-tuned DistilBERT model represents the contextual paradigm in our study. Evaluated on the same test set of 63,927 samples, this model leveraged the Transformer architecture to generate dynamic contextual embeddings and capture deep semantic dependencies.

DistilBERT achieved state-of-the-art results, setting a new benchmark for this project. It attained a Macro ROC-AUC of 0.9858 and a Macro F1-Score of 0.62, demonstrating an exceptional balance between Precision and Recall. Threshold optimization on the validation set (Table 8) revealed stable confidence calibration, with optimal thresholds generally ranging between 0.35 and 0.46, indicating high model confidence in positive predictions.

Table 8: Threshold Optimization Results on Validation Set

Toxicity Category	Optimal Threshold	Validation F1
toxic	0.45	0.8524
severe_toxic	0.39	0.5489
obscene	0.46	0.8481
threat	0.37	0.6531
insult	0.43	0.7801
identity_hate	0.44	0.6259

Table 9 illustrates the model's robust performance across all categories on the test set. The model maintained high F1-scores even for the most challenging minority classes, effectively overcoming the typical limitations associated with imbalanced datasets.

Table 9: DistilBERT Performance Metrics on Test Set

Category	AUC-ROC	F1-Score	Precision	Recall	Support
toxic	0.9737	0.69	0.56	0.90	6,087
severe_toxic	0.9901	0.42	0.31	0.66	367
obscene	0.9814	0.71	0.65	0.78	3,688
threat	0.9966	0.56	0.47	0.70	211
insult	0.9810	0.70	0.66	0.75	3,425
identity_hate	0.9918	0.63	0.67	0.60	712
Macro Avg	0.9858	0.62	0.55	0.73	-
Micro Avg	-	0.68	0.59	0.81	14,490

Detailed analysis of the results highlights the model's mastery over minority classes. The most remarkable performance was observed in the *identity_hate* category, where DistilBERT achieved an F1-score of 0.63 (Precision: 0.67, Recall: 0.60). This demonstrates that contextual embeddings effectively capture the subtle nuances of hate speech. Additionally, the model exhibited exceptional sensitivity for the majority class, achieving a Recall of 0.90 for toxic comments.

The ROC Curves (Figure 9) for threat and severe_toxic show near-perfect separation (AUC > 0.99). The curves hug the top-left corner tightly, indicating that the model ranks the most severe toxic comments correctly with extremely high reliability. This capability is crucial for prioritizing the review of high-risk content in real-world applications.

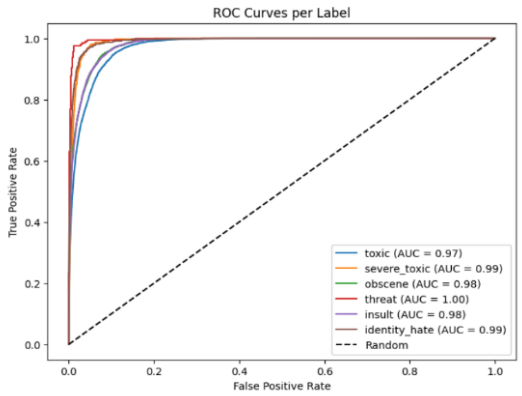


Figure 9: ROC Curves for DistilBERT Model

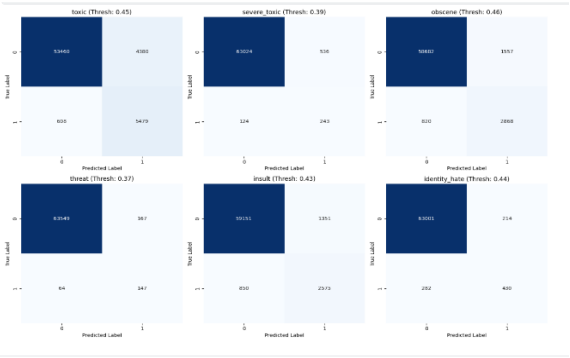


Figure 10: Confusion Matrices for DistilBERT Model

Furthermore, the Confusion Matrices (Figure 10) indicate a strong ability to filter False Positives. The self-attention mechanism allows DistilBERT to distinguish between toxic and non-toxic uses of ambiguous words, leading to a higher overall Precision (0.55 Macro Avg) while maintaining a strong Recall (0.73 Macro Avg). This performance validates the hypothesis that contextual understanding is critical for solving the multi-label toxicity classification problem.

6.4 Comparative Analysis

This study systematically evaluated three distinct computational paradigms to identify the optimal strategy for multi-label toxicity detection. Table 10 provides a summary comparison of the representative models across key performance metrics.

Table 10: Summary Comparison of Classification Paradigms

Metric / Model	LinearSVC (Statistical)	Text CNN (Deep Learning)	DistilBERT (Transformer)
Representation	Sparse (TF-IDF)	Static (Fast Text)	Contextual Dense
Macro ROC-AUC	0.9692	0.9555	0.9858
Macro F1-Score	0.5483	0.4764	0.6200

Critical Analysis & Key Findings

The comparative results highlight the "Contextual Premium" offered by dynamic embeddings. The most significant finding is the performance gap between DistilBERT and TextCNN. While both utilized dense embeddings, DistilBERT achieved a +30.1% higher F1-score compared to the TextCNN baseline (0.6200 vs 0.4764). This disparity underscores the limitation of static embeddings like FastText, which assign a fixed vector to polysemous words (e.g., "kill" or "attack") regardless of context. In contrast, DistilBERT's dynamic attention mechanism adapts the embedding based on the surrounding sentence structure. This allowed the model to accurately distinguish between benign gaming jargon and actual threats, resulting in near-perfect classification for the threat category.

Another notable observation is the trade-off between Statistical Robustness and Deep Learning Complexity. Unexpectedly, the statistical baseline (LinearSVC) outperformed the Text CNN deep learning baseline in terms of F1-score (0.5483 vs 0.4764). Toxic comments often rely on explicit profanity or specific keywords. TF-IDF's high-dimensional sparse representation preserves the distinct identity of these keywords perfectly. Text CNN, by compressing these into lower-dimensional static vectors, likely lost some of this discriminative signal. This suggests that for toxicity detection, a simple statistical model is preferable to a static-embedding deep learning model if computational resources are limited, whereas Transformers are necessary for state-of-the-art performance.

Finally, the efficacy of Focal Loss yielded distinct behaviors in the neural models. For Text CNN ($\gamma=2$), the loss function successfully forced the model to prioritize rare examples, but this aggressive focus led to "over-flagging" and lower Precision (0.3961). However, DistilBERT, aided by a higher focusing parameter *($\gamma=3$)* and superior semantic understanding, leveraged Focal Loss effectively. It captured minority classes with high Recall (e.g., Identity Hate) without sacrificing Precision, confirming that contextual transformers are indispensable for robust, balanced toxicity detection.

7. CLUSTERING IMPLEMENTATION

The clustering phase aimed to discover latent semantic structures within the toxic comments without relying on pre-defined labels. This unsupervised approach serves as a complementary strategy to classification, enabling the detection of emerging or uncatalogued toxicity patterns that supervised models might overlook.

7.1 Statistical Clustering (TF-IDF)

The first pipeline utilized the sparse TF-IDF representations derived in Section 4.1. Given the high dimensionality of the vocabulary (100,000 features), a direct application of distance-based algorithms would suffer from the "curse of dimensionality."

To address this, we implemented Dimensionality Reduction using Truncated SVD (Latent Semantic Analysis) [13]. The 100,000-dimensional TF-IDF vectors were projected down to 50 components (*n_components=50*). This step preserves the most significant variance in the data while filtering out noise and sparsity, rendering the dense vectors suitable for geometric clustering.

We applied two distinct algorithms on these reduced vectors.

Partition-Based Clustering: To identify broad thematic groups, we utilized the K-Means algorithm [14]. The optimal number of clusters (k) was determined using the Elbow Method, plotting the Inertia against a range of k values (2 to 9). As shown in Figure 11, the "elbow" point indicated the optimal tradeoff between compactness and complexity.

Density-Based Clustering: To capture complex, non-spherical structures and handle outliers, we employed HDBSCAN [15]. The algorithm was configured with a minimum cluster size of 100 (*min_cluster_size=100)* to avoid fragmented micro-clusters. A key advantage of this approach is its ability to classify points as "noise" (label -1), allowing us to distinguish between coherent toxic patterns and irrelevant random chatter.

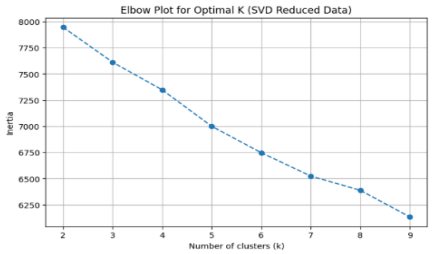


Figure 11: Elbow Method Plot for Optimal K Selection

7.2 Semantic Clustering (SBERT)

To overcome the limitations of keyword-based statistical clustering, we implemented a semantic pipeline that groups comments based on contextual intent rather than surface-level lexical overlap.

The raw text was first transformed into dense, semantically rich vectors using the pre-trained Sentence-BERT model *(all-MiniLM-L6-v2)* [7]. This generated 384-dimensional vectors where semantically similar sentences map close to each other. To further enhance clustering density and visualize the manifold, we applied UMAP (Uniform Manifold Approximation and Projection) [8] to reduce the embeddings to 5 dimensions. We configured UMAP with n_neighbors=15 and metric='cosine' to preserve the local neighborhood structure while maintaining global topological relationships.

We employed a dual strategy to capture both broad themes and fine-grained communities:

- **Macro-Clustering (K-Means):** As a baseline for broad thematic partitioning, we applied K-Means on the UMAP-reduced vectors. Re-applying the Elbow Method (Figure 12) on the semantic space indicated a distinct elbow at *k=5*. The model was initialized using *k-means++* with 10 random restarts to ensure convergence.
- **Micro-Clustering (HDBSCAN):** To detect arbitrarily shaped clusters, HDBSCAN was tuned with *min_cluster_size=100* and *min_samples=10*. This configuration balances cluster stability with sensitivity. We employed the Excess of Mass (EOM) selection method to persist larger, significant clusters while treating ambiguous points as noise.

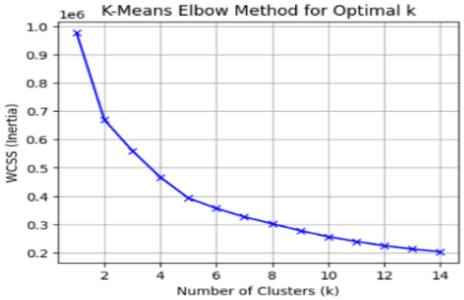


Figure 12: Elbow Method Plot for SBERT Clustering

To interpret the latent semantics of each group in the absence of ground-truth labels, we developed a robust profiling protocol. We applied a class-based TF-IDF (c-TF-IDF) approach, treating all comments within a cluster as a single document, to extract the top-10 distinctive keywords representing the group's theme. Complementing this, we calculated a Toxic Purity score for each cluster, defined as the percentage of comments explicitly labeled as toxic. This metric serves as a crucial risk indicator, enabling the identification of "high-risk" micro-communities that may require prioritized moderation.

8. CLUSTERING RESULTS

This section presents the findings from the unsupervised learning phase. We evaluated two distinct feature representation paradigms: Statistical (TF-IDF) and Semantic (SBERT) to discover latent structures within the comment dataset without relying on pre-defined toxicity labels.

8.1 Statistical Results

Clustering was performed on the dimensionality-reduced TF-IDF vectors (50 components via Truncated SVD). We compared the performance of Partition-Based (K-

Means) and Density-Based (HDBSCAN) algorithms to assess their ability to isolate toxic patterns based purely on lexical features.

As summarized in Table 11, the clustering quality varied significantly between the two algorithms. The K-Means algorithm *(k=4)* produced a structure with moderate separation but significant overlapping, evidenced by a low Silhouette Score of 0.2328. The external metrics (NMI: 0.0145, ARI: -0.0665) indicate a weak alignment with ground-truth toxicity labels. Although the Purity score appears high (0.9048), this is misleading; it primarily reflects the dataset's inherent class imbalance rather than the algorithm's success in isolating toxicity. In contrast, HDBSCAN achieved a substantially higher Silhouette Score (0.5826), demonstrating its superiority in identifying dense, coherent structures within the sparse feature space.

Table 11: Comparative Metrics for Statistical Clustering (TF-IDF)

Metric	K-Means (k=4)	HDBSCAN
Silhouette Score	0.2328	0.5826
NMI Score	0.0145	0.0125
ARI Score	-0.0665	-0.0490
Purity	0.9048	0.9056
Noise Points	-	26,905 (17.5%)

The visualization of the latent space is presented in Figure 13, which highlights the structural differences between the two algorithms.

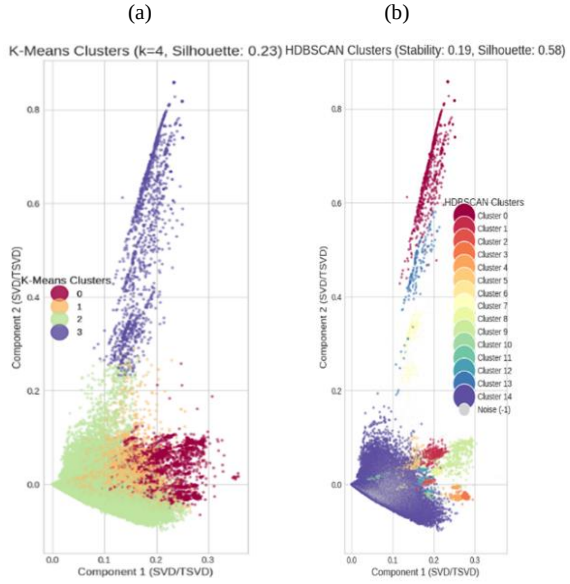


Figure 13: Comparison of K-Means (a) and HDBSCAN (b) Clusters

This visualization reveals the limitations of the TF-IDF approach. K-Means forces data into broad partitions where Clusters 0, 1, and 2 overlap substantially in the lower region, failing to form distinct toxic groups. Conversely, HDBSCAN leverages density to resolve complex, non-spherical structures. A major advantage of this approach is its explicit identification of 26,905 noise points (17.50% of the dataset), effectively filtering out outliers consisting of low-information terms (e.g., "please," "thanks," "edit") that lack sufficient density.

To understand the thematic content of the generated clusters, we analyzed the distinct vocabulary and sample distribution for both algorithms.

K-Means Characterization: This revealed that the algorithm identified broad, topic-based groupings rather than toxicity-based groups. As shown in Table 12, the largest group (Cluster 2) dominates with over 134,000 samples and is characterized by generic terms like "article," "wiki," and "page." This dominance dilutes any specific toxic signals, resulting in a low Toxic Purity of 1.12%. Interestingly, Cluster 3 captured specific moderation terms like "blocked" and "vandalize," yet achieved 0.00% purity, indicating that it grouped procedural notifications rather than actual toxic behavior.

Table 12: Semantic Characterization of K-Means

Cluster	Sample Count	Top Keywords	Toxic Purity
2	134,284	article, wiki, page, one	1.12%
1	13,933	talk, page, redirect, user	0.18%
0	4,098	page, deletion, image	0.00%
3	1,386	blocked, vandalize, edit	0.00%

Figure 14: K-Means Topic Profiling: Toxic vs. Non-Toxic Keyword Comparison

HDBSCAN Characterization: Conversely, this successfully uncovered smaller, highly coherent micro-clusters. For instance, Cluster 14 (122,025 samples) captures the bulk of general discussion, while smaller clusters isolate specific procedural interactions. As detailed in Table 13, clusters like Cluster 11 focus on "redirects" or "lists," while Cluster 0 isolates specific moderation events like "vandalism warnings" and "blocked" users.

Table 13: Semantic Characterization of HDBSCAN

Cluster	Sample Count	Top Keywords	Toxic Purity
14	122,025	article, page, talk, would	0.93%
11	1,233	redirect, list, film, album	0.08%
0	673	vandalize, blocked, edit	0.00%
1	399	test, sandbox, welcome	0.00%

Figure 15: HDBSCAN Topic Profiling: Top 2 Most Toxic Clusters Comparison

The analysis reveals distinct operational characteristics for each model. K-Means captures broad topic clusters with low toxic purity (0.00–1.12%), reflecting general discussion patterns such as Wikipedia editing and article management. This is useful for macro-level moderation, providing context on high-volume discussion areas. In contrast, HDBSCAN identifies semantically coherent clusters and effectively isolates noise. While its overall toxic purity remains low (0.00–0.93%), the high coherence of its clusters supports fine-grained moderation and the early detection of emerging subgroups (e.g., vandalism patterns) that do not fit the general conversation.

8.2 Semantic Results

This section presents a detailed analysis of unsupervised semantic clustering using contextual embeddings. Unlike surface-level methods, semantic clustering leverages advanced natural language understanding to uncover meaningful groupings in toxic comments.

The Comparative Cluster Visualization via UMAP 2D projection (Figure 16) highlights the differing philosophies of the two algorithms: K-Means partitions the semantic space into five balanced macro-clusters with clear boundaries, whereas HDBSCAN identifies 174 natural semantic micro-communities, labeling 56.01% of samples as noise to isolate dense regions.

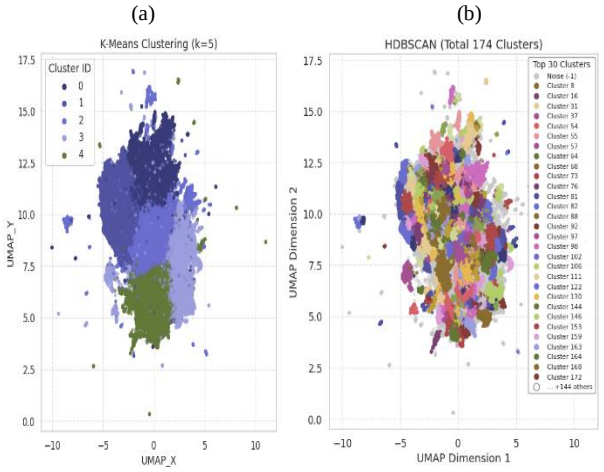


Figure 16: UMAP 2D Projection Showing K-Means (a) and HDBSCAN (b) Cluster Distributions

Table 14 presents a quantitative comparison of the two algorithms, identifying distinct performance metrics. HDBSCAN demonstrates superior cluster quality with a higher Silhouette Score (0.5177 vs. 0.2783) and a significantly better Davies-Bouldin Index (0.63 vs. 1.25), indicating that density-based clusters are naturally defined and well-separated. Conversely, K-Means yielded a higher Calinski-Harabasz Index (59,332 vs. 26,977), reflecting its tendency to create large, spherical macro-clusters that maximize inter-cluster variance. Notably, both methods exhibit low alignment with manual toxicity labels (NMI and ARI near 0), highlighting a key finding that semantic structure is driven by topic and intent, which does not perfectly overlap with human-annotated labels. However, HDBSCAN achieved a higher Cluster Purity (0.9415), effectively capturing homogeneous groups by filtering out ambiguous outliers.

Table 14: Semantic Clustering Performance Metrics Comparison

Metric	K-Means	HDBSCAN
Total Clusters	5	174
Silhouette Score	0.2783	0.5177
Calinski-Harabasz Index	59,332.44	26,977.18
Davies-Bouldin Index	1.25	0.63
NMI Score	0.0276	0.0376
ARI Score	0.0096	0.0010
Cluster Purity	0.8942	0.9415

The centroid-based K-Means approach revealed five distinct semantic macro-groupings. While useful for broad categorization, the toxic purity within these clusters is relatively diluted. As detailed in Table 15, Cluster 0 appears to be the primary repository for toxicity (22.95% purity), dominated by explicit profanity like "fuck," "bitch," and "shit." The remaining clusters (Clusters 1-4) are largely benign, focusing on Wikipedia-specific terminology such as "page" and "article," with low toxicity levels ranging from 3.59% to 14.63%.

Table 15: K-Means Semantic Cluster Profile

Cluster	Size	Toxic Purity (%)	Top Keywords
0	28,480	22.95%	fuck, bitch, shit, stupid
4	31,823	14.63%	page, wikipedia, blocked, edit
3	31,446	7.61%	article, people, think, know
2	37,646	4.12%	article, wikipedia, page, deletion
1	29,935	3.59%	image, article, talk, page

Figure 17: K-Means Cluster Topic Distribution

In contrast, **the density-based HDBSCAN approach** uncovered specialized high-toxicity clusters with exceptional semantic coherence. Unlike K-Means, it identified micro-communities with greater than 90% toxicity, revealing distinct abusive subcultures. Table 16 highlights these findings, showing that Cluster 145 (97.81% purity) specifically targets LGBTQ+ individuals, while Cluster 146 (90.77% purity) focuses on gender-based attacks. This granularity allows for the distinction between general extremist rhetoric found in Cluster 80 and targeted sexual orientation harassment in Cluster 140.

Table 16: HDBSCAN High-Toxicity Semantic Clusters

Cluster	Size	Toxic Purity (%)	Toxicity Specialization
145	137	97.81%	Targeted LGBTQ+ harassment
146	1,495	90.77%	Gender-based attacks
140	171	86.55%	Sexual orientation harassment
132	214	85.05%	Broad-spectrum abuse
80	309	41.10%	Extremist rhetoric

Figure 18: HDBSCAN Toxic Semantic Clusters: Top Two High-Purity Clusters

The analysis demonstrates that semantic clustering provides superior capability in uncovering the complex ecosystem of online toxicity through contextual understanding. HDBSCAN excels at discovering high-purity micro-clusters, successfully distinguishing between different toxicity types, such as homophobic harassment versus general profanity. This supports a tiered content moderation strategy utilizing K-Means for broad categorization and HDBSCAN for the deep investigation of high-risk patterns. This method complements supervised classification by enabling the discoverability of novel toxicity patterns and providing deeper linguistic insights into coded language and emerging abusive trends.

8.3 Comparative Analysis

This section synthesizes the findings from both statistical and semantic clustering experiments to identify the most effective unsupervised strategy for toxicity mining. Table 17 summarizes the quantitative performance across the two feature representations, highlighting the trade-offs between computational efficiency, cluster coherence, and toxicity purity.

Table 17: Performance Comparison of Clustering Approaches

Feature Space	Algorithm	Silhouette Score	Purity
Statistical (TF-IDF)	K-Means	0.2328	90.48%
Statistical (TF-IDF)	HDBSCAN	0.5826	90.56%
Semantic (SBERT)	K-Means	0.2783	89.42%
Semantic (SBERT)	HDBSCAN	0.5177	94.15%

A comparative assessment reveals distinct operational characteristics for each approach. The Statistical K-Means model proved to be computationally efficient for large datasets but suffered from poor cluster separation (Silhouette: 0.23), failing to isolate toxicity effectively due to lexical overlap. While Statistical HDBSCAN offered improved geometric separation and noise handling, it lacked the semantic context required to distinguish subtle vocabulary groups. Moving to the semantic domain, Semantic K-Means captured broad thematic structures but diluted toxicity (Purity: 89.42%) by forcing outliers into large clusters. Ultimately, Semantic HDBSCAN emerged as the optimal strategy, achieving the highest purity (94.15%) by identifying dense, high-risk micro-clusters.

The comparative success of the semantic density-based approach can be attributed to two fundamental mechanisms:

- **Representation Quality (TF-IDF vs. SBERT):** The transition from sparse TF-IDF vectors to dense SBERT embeddings fundamentally changed the clustering nature. TF-IDF was driven by exact keyword matches, grouping comments discussing "Wikipedia" together regardless of whether the sentiment was helpful or abusive. In contrast, SBERT was driven by intent and context. It successfully grouped varied expressions of hate, such as distinct clusters for homophobia vs. misogyny, even when they did not share the exact same vocabulary. This "Contextual Premium" allows for the discovery of implicit toxicity.
- **Algorithm Effectiveness (Centroid vs. Density):** Across both representations, HDBSCAN consistently outperformed K-Means due to two critical factors. The first is noise robustness. HDBSCAN's ability to designate ambiguous data (approx. 17-56%) as noise was a decisive advantage. By refusing to cluster noisy data, it allowed the remaining clusters to achieve high coherence. In contrast, K-Means forced every outlier into a cluster, which degraded the overall quality and purity of the groups. The second factor is cluster resolution. While K-Means found only 4-5 broad clusters, HDBSCAN identified between

15 (Statistical) to 174 (Semantic) distinct micro-communities. This granularity is essential for actionable moderation, as it isolates specific "attack vectors" rather than just general topics.

For the objective of exploratory toxicity mining, the combination of SBERT + UMAP + HDBSCAN emerges as the superior methodology. While computationally heavier, it provides the necessary semantic depth and granularity to uncover distinct, high-purity subcultures of abuse that statistical methods overlook.

9. CONCLUSION

This project addressed the critical challenge of online toxicity by implementing a dual-strategy framework that integrates supervised classification for immediate moderation and unsupervised clustering for exploratory analysis. By systematically evaluating statistical, deep learning, and transformer-based paradigms, we derived profound insights into the mechanics of detecting abusive content in a highly imbalanced dataset.

The transition from static to contextual embeddings proved to be the most transformative factor. DistilBERT emerged as the superior classification model, achieving a Macro ROC-AUC of 0.9858 and Macro F1-score of 0.62, significantly outperforming both the Text CNN (0.4764) and LinearSVC (0.5483) baselines. Its self-attention mechanism allowed it to master nuanced minority classes like *identity_hate* and *threat*, which static models consistently failed to interpret correctly due to their inability to resolve semantic ambiguity.

A key aspect of this study was the strategic implementation of Focal Loss to combat severe class imbalance. We observed distinct behaviors depending on the model architecture. For Text CNN ($\gamma=2$), the loss function successfully prioritized rare toxicity types but introduced a critical trade-off: the aggressive focus led to "over-flagging," resulting in a high rate of False Positives and lower Precision. However, DistilBERT, utilizing a higher focusing parameter ($\gamma=3$), demonstrated superior resilience. Its deep semantic understanding allowed it to leverage Focal Loss to maintain high Recall without sacrificing Precision as drastically as the CNN model, offering the most reliable balance for automated systems.

Furthermore, the unsupervised analysis revealed that standard clustering (K-Means on TF-IDF) is limited to grouping comments by general topics rather than toxicity. In contrast, the novel combination of SBERT + UMAP + HDBSCAN successfully uncovered high-purity toxic micro-communities (94.15% purity). This method acts as a powerful "threat intelligence" tool, capable of identifying specific attack vectors (e.g., coordinated harassment campaigns) that predefined supervised labels might not capture.

For a robust content moderation system, we recommend a hybrid deployment strategy. DistilBERT should be utilized for real-time, high-accuracy filtering of known toxicity types. This should be augmented by periodic Semantic Clustering (SBERT+HDBSCAN) to proactively identify and adapt to evolving trends in online abuse, ensuring the system remains effective against new forms of toxicity.

10. REFERENCES

[1] N. Van Royen, B. Poels, and H. J. Dais, "Automatic monitoring of cyberbullying on social networking sites: From keyword-based to context-based approaches," in *Proc. IEEE Int. Conf. on Web Intelligence*, 2015, pp. 445–452.

[2] Kaggle, "Jigsaw Toxic Comment Classification Challenge," 2018. [Online]. Available: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

[3] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.

[4] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 135–146, 2017.

[5] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1746–1751.

[6] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT: A distilled version of BERT: Smaller, faster, cheaper and lighter," *arXiv:1910.01108*, 2019.

[7] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. Conf. on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 3982–3992.

[8] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," *arXiv:1802.03426*, 2018.

[9] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

[10] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. 3rd Int. Conf. on Learning Representations (ICLR)*, 2015.

[11] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[12] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “LIBLINEAR: A library for large linear classification,” *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, 2008.

[13] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, “Indexing by latent semantic analysis,” *J. Amer. Soc. Inf. Sci.*, vol. 41, no. 6, pp. 391–407, 1990.

[14] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proc. 5th Berkeley Symp. on Math. Statist. and Prob.*, 1967, pp. 281–297.

[15] R. J. G. B. Campello, D. Moulavi, and J. Sander, “Density-based clustering based on hierarchical density estimates,” in *Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD)*, 2013, pp. 160–172.