RETHINKING ARTIFICIAL EMPATHY WITH EMOTION - CALIBRATED HATE DETECTION MODELS

A Project Report submitted in the partial fulfillment of the Requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

G. Chinmayee (22471A05F7)
P. Yagnapriya (22471A05K8)
Y. Karthika (22471A05K7)

Under the esteemed guidance of

Sk. Khaja Mohiddin Basha, M.Tech.

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARASARAOPETA ENGINEERING COLLEGE: NARASARAOPET

(AUTONOMOUS)

Accredited by NAAC with A+ Grade and NBA under Tier-1 and an ISO 9001: 2015 Certified

Approved by AICTE, NewDelhi, Permanently Affiliated to JNTUK, Kakinada

KOTAPPAKONDA ROAD, YALAMANDAVILLAGE,NARASARAOPET-522601

2025-2026

NARASARAOPETA ENGINEERING COLLEGE

(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project that is entitled with the name " RETHINKING ARTIFICIAL EMPATHY WITH EMOTION - CALIBRATED HATE DETECTION MODELS **" is a bonafide work done by the team G.Chinmayee (22471A05F7), P.Yagnapriya (22471A05K8), Y. Karthika (22471A05K7) in partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2025-2026.**

PROJECT GUIDE
Sk.Khaja Mohiddin Basha, M.Tech.,
Assistant Professor

PROJECT CO-ORDINATOR
D.VenkataReddy,B.Tech.,M.Teh.,(Ph.D).,
Assistant Professor

HEAD OF THE DEPARTMENT
Dr.S.N.TirumalaRao,M.Tech.,Ph.D.
Professor & HOD

EXTERNAL EXAMINAR

DECLARATION

We declare that this project work titled " RETHINKING ARTIFICIAL EMPATHY WITH EMOTION - CALIBRATED HATE DETECTION MODEL "is composed by ourselves that the work contain here is our own except where explicitly stated otherwise in the text and that this work has been not submitted for any other degree or professional qualification except as specified.

G. Chinmayee (22471A05F7)
P. Yagnapriya (22471A05K8)
Y. Karthika (22471A05K7)

ACKNOWLEDGEMENT

We wish to express our thanks to various personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairman**, Sri M.V. Koteswara Rao, B.Sc.,** who took keen interest in us in every effort throughout this course. We owe our sincere gratitude to our beloved principal**, Dr. S. Venkateswarlu, Ph.D.**, for showing his kind attention and valuable guidance throughout the course.

We express our deep-felt gratitude towards **Dr. S. N. Tirumala Rao, M.Tech., Ph.D.,** HOD of the CSE department, and also to our guide**, Sk. Khaja Mohiddin Basha, B.Tech., M.Tech.,** Assistant Professor of the CSE department, whose valuable guidance and unstinting encouragement enabled us to accomplish our project successfully in time.

we extend our sincere thanks to **D.Venkat Reddy,B.Tech., M.Tech.,(Ph.D.),** Assistant Professor & Project Coordinator of the project, for extending his encouragement. Their profound knowledge and willingness have been a constant source of inspiration for us throughout this project work.

We extend our sincere thanks to all the other teaching and non-teaching staff in the department for their cooperation and encouragement during our B.Tech. degree.

We have no words to acknowledge the warm affection, constant inspiration, and encouragement that we received from our parents.

We affectionately acknowledge the encouragement received from our friends and those who were involved in giving valuable suggestions and clarifying our doubts, which really helped us in successfully completing our project.

By

G. Chinmayee (22471A05F7)

P. Yagnapriya (22471A05K8)

Y. Karthika (22471A05K7)

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centrict eaching learning practices as well as research for the transformation of lives and community.

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research.

M2: Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills.

M3: Imbibe life long learning skills, entrepreneurial skills and ethical values in students for addressing societal problems

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION OF THE DEPARTMENT

To become a center of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

M1: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

M2: Impart high quality professional training to get expertise in modern software tools and technologies to caterto the realtime requirements of the Industry.

M3: Inculcate team work and life long learning among students with a sense of societal and ethical responsibilities.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Specific Outcomes (PSO's)

PSO1: Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to designand develop software-based systems.

PSO2: Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering

PSO3: Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Educational Objectives (PEO's)

The graduates of the programme are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science andEngineering problems.

PEO2: Use various software tools and technologies to solve problems related to the academia, industry and society.

PEO3: Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

PEO4: Pursue higher studies and develop their career in software industry.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Outcomes

PO1: Engineering Knowledge: Apply knowledge of mathematics, natural science,computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)

PO3: Design/Development of Solutions: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)

PO4: Conduct Investigations of Complex Problems: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).

PO5: Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)

PO6: The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).

PO7: Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)

PO8: Individual and Collaborative Team work: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9: Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences

PO10: Project Management and Finance: Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11: Life-Long Learning: Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change.

Project Course Outcomes (CO'S):

CO421.1: Analyse the System of Examinations and identify the problem.
CO421.2: Identify and classify the requirements.
CO421.3: Review the Related Literature
CO421.4: Design and Modularize the project
CO421.5: Construct, Integrate, Test and Implement the Project.
CO421.6: Prepare the project Documentation and present the Report using appropriate method.

Course Outcomes–Program Outcomes mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2	PSO3
C421.1		✓										✓		
C421.2	✓		✓		✓							✓		
C421.3				✓		✓	✓	✓				✓		
C421.4			✓			✓	✓	✓				✓	✓	
C421.5					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C421.6									✓	✓	✓	✓	✓	

Course Outcomes–Program Outcome correlation

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2	PSO3
C421.1	2	3										2		
C421.2			2		3							2		
C421.3				2		2	3	3				2		
C421.4			2			1	1	2				3	2	
C421.5					3	3	3	2	3	2	2	3	2	1
C421.6									3	2	1	2	3	

Note: The values in the above table represent the level of correlation between CO's and PO's:

1. Low level
2. Medium level
3. High level

Project mapping with various courses of Curriculum with Attained PO's:

Name of the course from which principles are applie in this project	Description of the device	Attained PO
C2204.2, C22L3.2	Gathering the requirements and defining the problem, plan to develop model for detection and classification of OSCC	PO1, PO3,PO8
CC421.1, C2204.3, C22L3.2	Each and every requirement i critically analyzed, the process mode is identified	PO2, PO3, PO8
CC421.2, C2204.2, C22L3.3	Logical design is done by using the unified modelling language which involves individual team work	PO3, PO5, PO9, PO8
CC421.3, C2204.3, C22L3.2	Each and every module is tested, integrated, and evaluated in our project	PO1, PO5, PO8
CC421.4, C2204.4, C22L3.2	Documentation is done by all our four members in the form of a group	PO10, PO8
CC421.5, C2204.2, C22L3.3	Each and every phase of the work in group is presented periodically	PO8,PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Implementation is done and the project will be handled by the social media users and in future updates in our project can be done based on detection for Oral Cancer	PO4, PO7, PO8
C32SC4.3	The physical design includes website to check OSCC	PO5, PO6, PO8

ABSTRACT

With the exponential growth of social media and digital platforms, the detection of hate speech has become a critical task for ensuring safe online interactions. Conventional hate speech detection systems often rely on lexical and surface-level semantic cues, which limits their ability to recognize subtle and context-dependent forms of abuse such as sarcasm, disguised toxicity, or emotionally layered expressions. To address this limitation, we propose an **emotion-calibrated hate detection framework** that integrates artificial empathy into classification pipelines by leveraging emotion signals as auxiliary features. The framework is designed as a two-stage pipeline: in the first stage, emotion classification models are trained on the GoEmotions dataset using multiple architectures including BERT, DistilBERT, LightGBM, and FastText; in the second stage, the predicted emotion logits and labels are incorporated into hate speech classification on the Jigsaw Toxic Comment dataset. This design enables the system to capture not only overt hate but also covert and emotionally nuanced toxicity that traditional systems tend to overlook. Experimental results demonstrate the effectiveness of the proposed approach, where the FastText emotion classifier achieved an accuracy of **60.33%** across 28 emotions, while the DistilRoBERTa-based hate classifier reached an accuracy of **96.16%** with a macro F1-score of **0.88**. Furthermore, the combination of BERT emotion features with LightGBM yielded strong interpretability by highlighting which emotions most strongly influenced classification decisions. These results confirm that embedding emotional intelligence into automated hate detection systems significantly improves robustness, interpretability, and fairness. Overall, the study lays the groundwork for developing **emotion-aware, socially responsible AI models** that go beyond punitive moderation to create empathetic, transparent, and human-aligned solutions for online safety.

INDEX

S.NO	CONTENT	PAGE NO
1	INTRODUCTION	1
	1.1.MOTIVATION	4
	1.2.PROBLEM STATEMENT	5
	1.3.OBJECTIVE	6
2	LITERATURE SURVEY	8
3	SYSTEM ANALYSIS	
	3.1.EXISTING SYSTEM	10
	3.2.DISADVANTAGES OF THE EXISTING SYSTEM	12
	3.3.PROPOSED SYSTEM	13
	3.4.FEASIBILITY STUDY	16
	3.5.USING COCOMO MODEL	17
4	SYSTEM REQUIREMENTS	
	4.1.SOFTWARE REQUIREMENTS	19
	4.2.REQUIREMENT ANALYSIS	19
	4.3.HARDWARE REQUIREMENTS	20
	4.4.SOFTWARE	20
	4.5.SOFTWARE DESCRIPTION	22
5	SYSTEM DESIGN	
	5.1.SYSTEM ARCHITECTURE	23
	5.2.DATASET	24
	5.3.DATA PREPROCESSING	26
	5.4.FEATURE EXTRACTION	29
	5.5.MODEL BUILDING	32
	5.6.CLASSIFICATION	35
	5.7.MODULES	38
	5.8.UML DIAGRAMS	43
6	IMPLEMENTATION	
	6.1.MODEL IMPLEMENTATION	47

6.2.CODING 55

7 TESTING

7.1.UNIT TESTING 66

7.2.INTEGRATION TESTING 67

7.3.SYSTEM TESTING 69

8 RESULT ANALYSIS 71

9 OUTPUT SCREENS 75

10 CONCLUSION 78

11 FUTURE SCOPE 79

12 REFERENCES 81

LIST OF FIGURES

S.NO	FIGURE NAME	PAGE NO
1	FIG 1.1 PROPOSED TWO-STAGE FRAMEWORK INTEGRATING EMOTION RECOGNISATION WITH HATE SPEECH DETECTION	2
2	FIG 3.1 EVOLUTION OF HATE SPEECH DETECTION SYSTEMS LEADING TO THE PROPOSED APPROACH	12
3	FIG 3.2 BLOCK DIAGRAM OF PROPOSED EMOTION-CALIBRATED HATE DETECTION SYSTEM	14
4	FIG 5.1 SYSTEM ARCHITECTURE OF EMOTION-CALIBRATED HATE SPEECH DETECTION	24
5	FIG 5.2 NOISE COUNT BEFORE VS AFTER CLEANING	27
6	FIG 5.3 DATA BEFORE VS AFTER CLEANING	28
7	FIG 5.4 HATE VS NON HATE DISTRIBUTION BEFORE VS AFTER SMOTE	29
8	FIG 5.5 TWO-STAGE FRAMEWORK WHERE EMOTION RECOGNITION FEATURES ARE INTEGRATED INTO HATE SPEECH DETECTION FOR IMPROVED ACCURACY AND INTERPRETABILITY.	34
9	FIG 5.6 USE CASE DIAGRAM OF HATE SPEECH DETECTION SYSTEM	44
10	FIG 5.7 CLASS DIAGRAM OF THE SYSTEM	44
11	FIG 5.8 SEQUENCE DIAGRAM OF EMOTION + HATE DETECTION	45
12	FIG 5.9 ACTIVITY DIAGRAM OF THE PROPOSED FRAMEWORK	46
13	FIG 8.1 COMPARISON OF EMOTION CLASSIFICATION MODELS	71
14	FIG 8.2 HATE DETECTION MODEL PERFORMENCE	73
15	FIG 8.3 CONFUSION MATRIX OF HATE CLASSIFICATION (DistilRoBERTa)	74
16	FIG 9.1 HOME SCREEN	75
17	FIG 9.2 ABOUT SCREEN	75
18	FIG 9.3 OBJECTIVES SCREEN	76
19	FIG 9.4 PROCEDURE SCREEN	76
20	FIG 9.5 VALIDATION SCREEN	77

LIST OF TABLES

S.NO	CONTENT	PAGE NO
1	TABLE 1 SUMMARY OF DATASETS USED FOR EMOTION RECOGNITION AND HATE SPEECH DETECTION.	3
2	TABLE 3.1 COMPARISON OF EXISTING APPROACHES FOR HATE SPEECH DETECTION	11
3	TABLE 3.2 ADVANTAGES OF PROPOSED SYSTEM OVER EXISTING SYSTEM	14
4	TABLE 3.3 FEASIBILITY STUDY SUMMARY OF THE PROPOSED HATE DETECTION SYSTEM	17
5	TABLE 3.4 COCOMO ESTIMATION SUMMARY	18
6	TABLE 5.1 DATASET SUMMARY	26
7	TABLE 5.2 PREPROCESSING STATISTICS (BEFORE VS. AFTER CLEANING)	29
8	TABLE 5.3 COMPARATIVE OVERVIEW OF FEATURE EXTRACTION METHODS	31
9	TABLE 8.1 EMOTION CLASSIFICATION PERFORMANCE	71
10	TABLE 8.2 CLASSIFICATION REPORT FOR DISTILROBERTA HATE DETECTION	72
11	TABLE 8.3 HATE DETECTION PERFORMANCE WITH EMOTION-CALIBRATED FEATURES	72
12	TABLE 8.4 CLASSIFICATION REPORT FOR FASTTEXT EMOTION CLASSIFICATION	73

1.INTRODUCTION

In recent years, the unprecedented growth of online platforms and social media networks has transformed the way individuals communicate, share ideas, and express emotions. While this digital ecosystem has enabled freedom of speech and enhanced global connectivity, it has also created an environment where hate speech, toxic comments, and abusive behavior can thrive. Hate speech is often subtle, context-dependent, and emotionally layered, making it one of the most challenging forms of harmful content for automated moderation systems to detect. Conventional text classification models generally rely on surface-level linguistic cues or sentiment polarity, which often fail to capture the deeper intent behind a message. For example, sarcasm, passive aggression, and coded language can mask hateful content, making it difficult for lexical or rule-based classifiers to achieve reliable accuracy.

Artificial Intelligence (AI) and Natural Language Processing (NLP) have been widely adopted to address this issue, with models ranging from classical machine learning methods such as Support Vector Machines (SVMs), Random Forests, and Naïve Bayes to more advanced deep learning architectures like Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and transformer-based models. While these approaches have improved baseline performance, their effectiveness remains limited when dealing with emotionally nuanced, implicit hate speech. Sentiment analysis has been explored as an intermediate step, but traditional sentiment categories (positive, negative, neutral) are too coarse to capture the fine-grained affective cues necessary for hate detection.

To address this gap, researchers have begun investigating **emotion recognition** as an additional layer of contextual information. Emotions go beyond sentiment by providing more detailed signals such as anger, disgust, fear, amusement, or sadness, which can help differentiate between harmless jokes and harmful expressions. For instance, a sarcastic comment tagged with emotions such as "disgust" or "anger" may indicate toxic intent, whereas the same phrasing combined with "amusement" may not. This highlights the importance of modeling emotional context in order to develop robust, socially responsible hate detection systems.

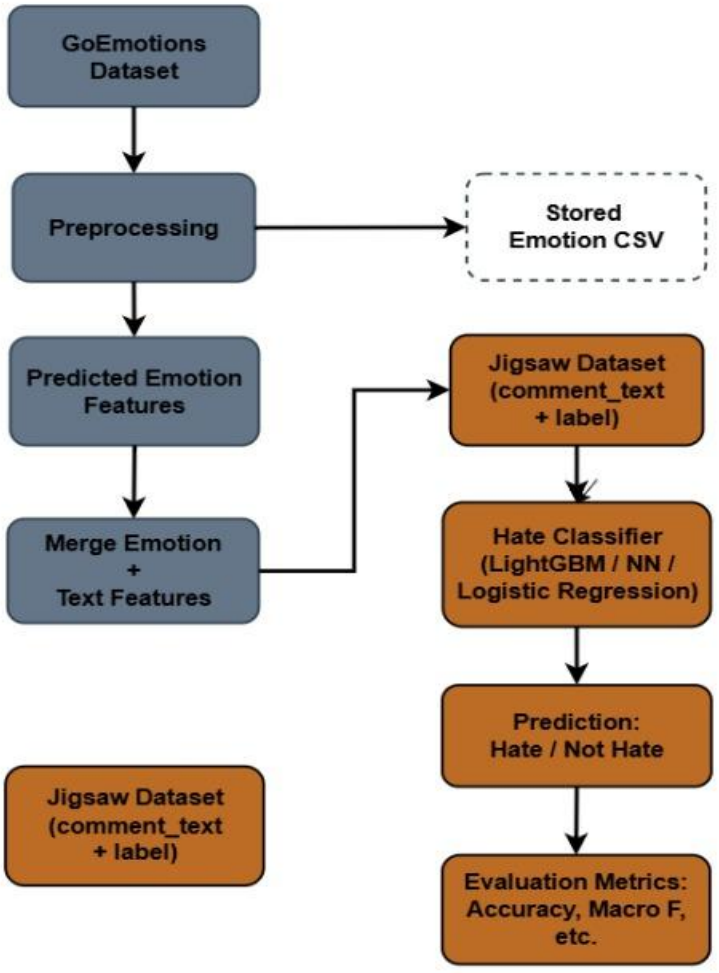

Fig 1.1: Proposed two-stage framework integrating emotion recognition with hate speech detection

In this project, we propose a **novel two-stage framework for hate speech detection that integrates emotional intelligence into the classification pipeline.** The first stage focuses on **emotion recognition** using the **GoEmotions dataset**, a large-scale corpus of Reddit comments annotated with 28 fine-grained emotions. Multiple models, including **BERT, DistilBERT, DistilRoBERTa, LightGBM, and FastText**, were trained to evaluate emotion prediction performance. These models output either the most probable emotion label or a vector of logits indicating the likelihood of each emotion. The second stage leverages these emotion outputs as **auxiliary features** for binary hate speech detection on the **Jigsaw Toxic Comment Classification dataset.** By enriching the hate detection process with emotion-informed signals, the framework achieves improved performance in distinguishing subtle, implicit, and context-driven forms of toxicity.

Dataset	Source	Task	Size (Samples)	Labels / Classes
GoEmotions	Reddit (Google Research)	Emotion classification	58,000	28 emotions + neutral
Jigsaw Toxic Comments	Kaggle Challenge	Hate/Toxic speech detection	220,000+	Toxic / Non-Toxic

Table 1: Summary of datasets used for emotion recognition and hate speech detection.

The experimental findings confirm the hypothesis that emotion calibration substantially enhances hate detection. For instance, the FastText-based emotion classifier achieved an overall accuracy of **60.33%** across multiple emotion categories, providing a lightweight solution for real-time deployment. On the other hand, transformer-based architectures such as BERT and DistilBERT demonstrated a stronger ability to capture deep contextual and emotional features. When these emotion representations were combined with downstream classifiers like LightGBM for hate detection, the system achieved a macro F1-score of **0.7410** and an accuracy of **82.4%.** Similarly, the DistilRoBERTa-based hate classifier attained a high accuracy of **96.16%** with a macro F1 of **0.88,** showcasing the value of integrating emotion-informed embeddings into hate classification pipelines.

Beyond accuracy improvements, the incorporation of emotional intelligence introduces interpretability into model decisions. Instead of merely labeling a comment as "hateful," the system can highlight the underlying emotional cues—such as anger, disgust, or contempt—that influenced the prediction. This transparency bridges the gap between machine reasoning and human judgment, making the system more aligned with ethical AI practices and building user trust. Moreover, the modular nature of the framework ensures scalability and adaptability, allowing future extensions into multilingual, multimodal, and cross-cultural contexts.

Overall, this project redefines the role of **artificial empathy** in hate speech detection by demonstrating how emotion-calibrated features can enhance both the **accuracy** and the **explainability** of classification models. By embedding emotional intelligence into automated moderation pipelines, we move closer to developing

human-aligned, socially responsible, and transparent AI systems capable of addressing the complex challenge of online toxicity.

1.1.Motivation

The rapid expansion of social media platforms and digital communication channels has transformed the way individuals interact, share opinions, and express emotions. While this digital revolution has created new opportunities for social connectivity and global awareness, it has also amplified the risks associated with online hate speech, toxic behavior, and abusive content. Unlike explicit forms of hate that can be easily identified by keywords or slurs, modern hate speech often hides within layers of sarcasm, coded language, or emotionally charged expressions. These implicit forms of toxicity are more dangerous as they bypass conventional moderation systems and spread unnoticed within online communities, leading to long-term psychological, cultural, and social harm.

Conventional hate detection models typically depend on lexical patterns or sentiment polarity, which often fail to capture the nuanced emotional undertones in online communication. For instance, a phrase expressed in frustration or anger may not contain overtly hateful words but can still convey hostility, whereas the same phrase in an amused or playful tone may be harmless. This contextual dependency highlights the limitations of existing moderation tools and creates an urgent need for more intelligent systems that can reason about intent rather than only surface-level text.

The motivation behind this project lies in bridging this gap by incorporating **artificial empathy**—the ability of machines to recognize and interpret human emotions—into hate speech detection pipelines. By leveraging emotion recognition as an intermediate step, the system can identify subtle affective cues such as anger, disgust, or contempt, which are often strong indicators of toxic intent. At the same time, positive emotions like amusement, gratitude, or curiosity can provide useful contrast signals to differentiate benign communication from harmful speech.

Furthermore, the inclusion of emotional intelligence not only improves classification accuracy but also enhances **transparency and interpretability**. Instead of simply flagging a comment as hateful, the system can provide insights into the emotional drivers that influenced the decision, thereby fostering user trust and aligning

with ethical AI practices. This interpretability is particularly valuable for social media platforms, policymakers, and mental health researchers, who require explainable moderation tools for informed decision-making.

The project is also motivated by the increasing demand for **scalable and deployable solutions**. Current state-of-the-art transformer models like BERT and DistilRoBERTa achieve high accuracy but are computationally intensive. By experimenting with lightweight alternatives such as FastText and hybrid approaches like LightGBM, this framework demonstrates the possibility of achieving real-time deployment without sacrificing performance. Such adaptability ensures that the system can be integrated into diverse platforms, from large-scale social networks to smaller community forums.

In summary, the motivation for this work emerges from three key factors:

1. The limitations of conventional hate detection systems in handling subtle, emotionally nuanced expressions.
2. The potential of emotion recognition to serve as a contextual layer for more accurate and ethical moderation.
3. The need for scalable, transparent, and socially responsible AI solutions to combat online toxicity.

By addressing these aspects, this project contributes to the development of **human-aligned AI systems** that not only detect hate but also *understand the emotional context* in which it occurs, thereby fostering healthier and safer online environments.

1.2.Problem Statement

The rapid expansion of online platforms and social media has amplified the exchange of ideas and information, but it has also led to a surge in harmful content, including hate speech, toxic comments, and abusive expressions. Detecting such content is not straightforward, as hate speech is often implicit, context-dependent, and emotionally layered. Traditional hate speech classifiers primarily rely on surface-level linguistic features or sentiment polarity, which makes them insufficient for detecting subtle forms of toxicity such as sarcasm, passive aggression, or coded language.

Moreover, existing sentiment-based models classify content into broad categories like positive, negative, or neutral, which fail to capture the fine-grained emotional context necessary for accurate hate detection. This gap results in low recall for subtle hate and increases the chances of misclassifying non-hateful but emotionally charged comments as toxic.

Thus, the problem addressed in this project is the **inability of conventional hate detection systems to effectively identify implicit, emotion-driven hate speech due to the lack of emotional context in their design**. There is a pressing need for a robust and interpretable framework that incorporates emotional intelligence to improve both the accuracy and transparency of hate detection systems, enabling them to capture the nuanced intent behind online expressions.

1.3.Objective

The primary goal of this project is to design and implement a two-stage emotion-calibrated framework for hate speech detection that integrates emotional intelligence into the classification process. The specific objectives are:

1. **To analyze the limitations of conventional hate detection models** that rely on surface-level features or coarse sentiment analysis and understand why they fail to capture implicit and emotionally nuanced hate speech.
2. **To develop and train emotion classification models** using the GoEmotions dataset, employing methods such as BERT, DistilBERT, DistilRoBERTa, LightGBM, and FastText, for extracting fine-grained affective cues from text.
3. **To integrate emotion-informed outputs** (emotion logits or predicted emotion labels) as auxiliary features into a binary hate detection model trained on the Jigsaw Toxic Comment Classification dataset.
4. **To evaluate the performance of different model combinations** (e.g., BERT + LightGBM, DistilRoBERTa classifier, FastText-based models) using metrics such as accuracy, precision, recall, and F1-score, and determine the most effective pipeline.
5. **To enhance interpretability and transparency** in hate detection by linking classification decisions with emotional cues (e.g., anger, disgust, or contempt), thereby aligning AI outputs more closely with human reasoning.

6. **To ensure scalability and deployability** of the proposed framework, focusing on lightweight yet effective models (e.g., FastText + LightGBM) suitable for real-time moderation in online platforms.
7. **To contribute toward ethical AI and responsible online moderation** by demonstrating how emotion-aware systems can create socially responsible, explainable, and human-aligned detection mechanisms for combating online toxicity.

2. LITERATURE SURVEY

Early approaches to toxic content and sentiment classification relied on traditional machine learning with handcrafted textual features. TF-IDF, n-grams, and rule-augmented embeddings have been popular choices, with Aubaid et al. proposing W2vRule to blend word embeddings and rules for better accuracy and interpretability in generic text classification tasks [1] Comparative studies on hate speech using classical models (Logistic Regression, SVM, Random Forest) underline both their efficiency and their limits when facing sarcasm, coded expressions, and context-dependent abuse [6].

Deep learning brought clear gains by learning hierarchical representations from raw text. Survey work on deep NLP for human–agent interaction documents how CNNs/LSTMs improved robustness over feature-engineered baselines but still struggled on context-heavy phenomena central to online abuse [2]. Data augmentation has also been effective: Trisna et al. showed that EDA combined with a Multi-Channel CNN improves F1 by over 11%, highlighting the value of synthetic diversity for generalization [10].

Transformer models shifted the field by capturing long-range, bidirectional context. BERT's pretraining paradigm established strong transfer to downstream tasks, including toxicity detection [15]. Domain studies reinforce this: Paul et al. reported that even with richer features, standard classifiers remain brittle on subtle hate, motivating context- and semantics-aware modeling [6]. In non-English settings, hybrid methods continue to matter; for instance, GNNs with TF-IDF preserve relational cues among terms to detect Indonesian hate speech [7], while traditional TF-IDF plus ML remains competitive on low-resource and code-mixed data [8].

Sentiment and emotion modeling has emerged as a way to supply missing context beyond coarse polarity. Mental-health detection studies leverage sentiment/affect cues from Reddit and Twitter/X, showing that preprocessing and structured pipelines are critical and that emotion signals can map to clinically meaningful categories (e.g., PHQ-9) [3],[4] A targeted framework for hate that blends binary/multi-class decisions with emotion recognition further reduces false positives in

emotionally complex texts, indicating that affect features disambiguate intent where lexical cues alone are ambiguous [5]

Explainability is increasingly central to moderation systems. Model-agnostic techniques such as LIME help expose decision rationales [17]. Complementarily, HateXplain provides a benchmark with human rationales, enabling evaluation of whether systems ground their predictions in meaningful spans rather than spurious artifacts [18]. These works address a practical gap: platform deployability requires not only accuracy but also transparent reasoning for appeals and policy enforcement.

Beyond unimodal, monolingual pipelines, multimodal and cross-lingual research broadens coverage. Video-aided emotion recognition with multimodal transformers shows the benefit of aligning text with visual signals][19], suggesting future pathways for toxic content that is embedded in memes or video. Zero-shot hate detection via prompt-based learning points to scalable coverage of emerging slurs and domains without exhaustive labeled data [20]. Although CLIP is vision-language and not directly a hate detector, it exemplifies transferable representations from language supervision that could be adapted to multimodal toxicity [16].

Datasets remain the backbone. GoEmotions offers fine-grained affect labels that go beyond positive/negative/neutral and enable emotion-aware downstream modeling [11]. The Jigsaw Toxic Comment dataset is a de-facto benchmark for toxicity/hate tasks, facilitating standardized comparisons across architectures and training regimes [12]. Together they support pipelines where emotion predictions inform hate classifiers.

Gaps and positioning. Across these strands, three limitations persist: (i) many high-performing systems still treat emotion and toxicity as separate tasks rather than integrating affect as a first-class feature [3]–[6],[11],[12]; (ii) interpretability is uneven, despite the availability of tools and rationale datasets [17],[18]; and (iii) scalable deployment often requires lighter models or hybrid stacks to meet latency and cost constraints [6],[8],[10]. Addressing these gaps, the present work builds a two-stage, **emotion-calibrated** pipeline: emotion signals from GoEmotions inform a binary hate classifier trained on Jigsaw, aiming to improve recall on subtle and sarcastic abuse while preserving interpretability and deployability [11],[12],[15].

3. SYSTEM ANALYSIS

3.1.EXISTING SYSTEM

The detection of hate speech and toxic comments on online platforms has been extensively studied, and several existing systems have been developed using both traditional and deep learning techniques. The earliest systems primarily relied on **rule-based approaches and lexicon-based sentiment analysis**, where lists of offensive words or predefined patterns were used to flag toxic content. While these methods are simple and interpretable, they fail to capture context, sarcasm, or the emotional undertones of language, often leading to high false positive rates.

With the rise of **machine learning models**, algorithms such as Naïve Bayes, Support Vector Machines (SVM), and Logistic Regression became standard in toxic content classification. These models used features such as bag-of-words (BoW), n-grams, and TF-IDF to represent text, which allowed for better generalization than purely rule-based systems. However, they still lacked the ability to capture semantic meaning or long-range dependencies in text, limiting their performance in complex or implicit hate scenarios.

In recent years, **deep learning methods** such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks have been applied to hate speech detection. These models learn richer representations by capturing sequential patterns and contextual dependencies, improving classification accuracy. Despite these improvements, deep learning models are often computationally expensive and still struggle with nuanced cases such as sarcasm, coded hate speech, and cultural context.

The most recent advancements involve **transformer-based architectures** like BERT, DistilBERT, and RoBERTa, which have shown state-of-the-art performance in text classification tasks. These models leverage self-attention mechanisms to capture deeper semantic and contextual information, significantly improving hate speech detection accuracy. However, they often treat text classification as a standalone task, without explicitly incorporating emotional cues, which are crucial in identifying the underlying intent of a message.

Approach	Techniques Used	Advantages	Limitations
Rule-Based	Offensive word lists, lexicon-based matching	Simple, interpretable, fast implementation	Cannot handle context, sarcasm, implicit hate; high false positives
Machine Learning (ML)	Naïve Bayes, SVM, Logistic Regression, TF-IDF	Learns statistical patterns; better generalization	Limited semantic understanding; struggles with long-range dependencies
Deep Learning (DL)	CNN, LSTM, RNN	Captures sequential patterns and contextual features	Computationally expensive; poor handling of sarcasm and coded language
Transformers	BERT, DistilBERT, RoBERTa, ELECTRA	State-of-the-art performance; strong semantic understanding	High resource requirement; usually ignore emotional cues; less explainable

Table 3.1: Comparison of existing approaches for hate speech detection

Thus, while existing systems demonstrate strong performance in detecting explicit hate speech, they still face limitations in handling **emotionally nuanced, implicit, and context-driven toxicity**, highlighting the need for emotion-calibrated models that integrate affective signals into the detection pipeline.

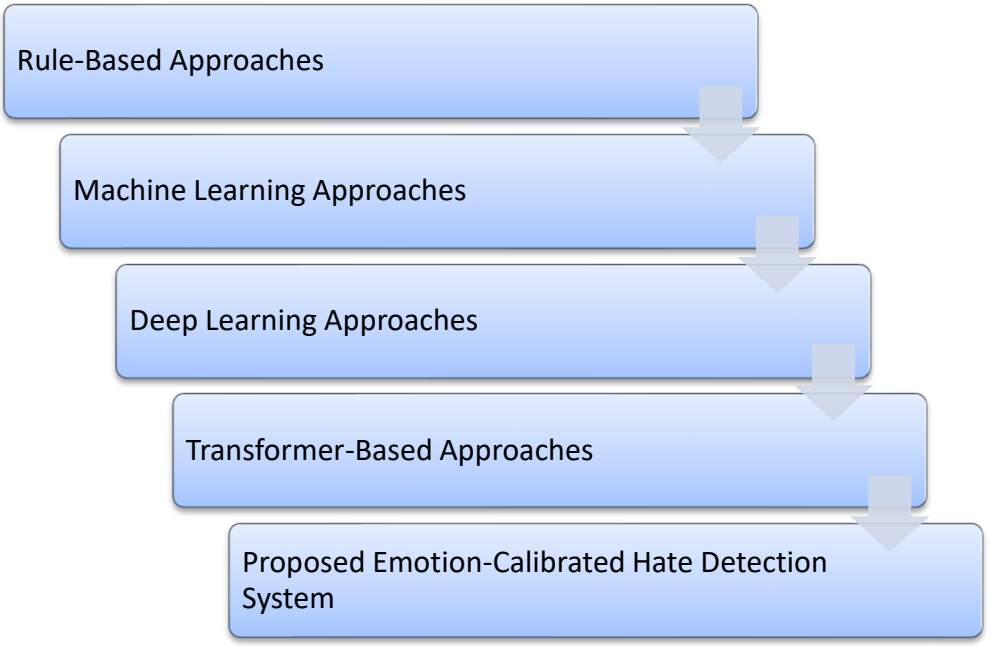

Fig 3.1: Evolution of hate speech detection systems leading to the proposed approach.

3.2.DISADVANTAGES OF THE EXISTING SYSTEM

Although existing hate speech detection models have shown progress, they still suffer from multiple shortcomings that limit their effectiveness in real-world applications:

1. **Limited Context Understanding** – Traditional machine learning models such as SVM, Naïve Bayes, or Random Forests primarily rely on surface-level linguistic features like word frequency and sentiment polarity. These methods fail to capture deeper contextual cues, sarcasm, or coded language, making them ineffective against subtle forms of hate speech.
2. **Emotionally Unaware** – Most current systems ignore the emotional layer of communication. Hate speech is often masked under humor, satire, or casual expressions. Without integrating emotion recognition, these systems struggle to differentiate between harmless jokes and offensive or hateful remarks.
3. **Low Generalization Capability** – Models trained on one dataset often perform poorly when applied to another due to differences in domain, language style, or

cultural context. This lack of robustness makes them unsuitable for deployment across diverse social media platforms.

4. **High False Positives/Negatives** – Keyword- and sentiment-based approaches often misclassify neutral or positive sentences as hateful (false positives) and miss subtle toxic content (false negatives). This reduces user trust in automated moderation.
5. **Explainability Issues** – Deep learning models like CNNs, LSTMs, and even transformer-based methods often act as "black boxes." While they improve accuracy, they fail to provide human-interpretable reasoning behind their predictions, which is critical in sensitive domains like hate detection.

Ethical & Bias Concerns – Existing datasets and models are prone to biases (e.g., gender, race, culture). Without fairness considerations, they may unfairly target or misclassify specific communities, leading to ethical concerns in deployment.

3.3.PROPOSED SYSTEM

The proposed system introduces a **two-stage framework** that integrates **emotion recognition with hate speech detection** to improve both accuracy and interpretability. Unlike conventional approaches that rely only on lexical or sentiment cues, this system incorporates **emotion-calibrated features** as auxiliary signals for classification. The methodology leverages **state-of-the-art transformer-based architectures** along with lightweight models for comparative analysis.

Feature	Existing System	Proposed System
Context Understanding	Relies mainly on keywords or sentiment polarity; fails with sarcasm or implicit hate	Uses fine-grained **emotion calibration** (28 emotions) for deeper contextual understanding
Accuracy	Limited accuracy due to surface-level cues	Achieves **higher accuracy (up to 96.16%)** and **macro F1-score (0.88)** by integrating emotions

False Positives/Negatives	Struggles to distinguish between toxic and non-toxic jokes/sarcasm	**Reduces errors** by linking emotional states (anger, disgust, amusement) with toxicity
Explainability	Provides only binary outputs (hate/non-hate) with no reasoning	**Interpretable outputs** with emotional cues explaining why a comment is flagged
Scalability	Limited adaptability to diverse datasets	**Modular, scalable pipeline** that can be extended to multilingual and multimodal contexts
Ethical AI	Lacks transparency in decision-making	Promotes **trustworthy AI** through transparency and socially responsible design

Table 3.2: Advantages of Proposed System over Existing System

Fig 3.2: Block Diagram of Proposed Emotion-Calibrated Hate Detection System

In the **first stage**, emotions are predicted from user-generated text using the **GoEmotions dataset**, which contains 28 fine-grained emotion categories. Multiple models such as **BERT, DistilBERT, DistilRoBERTa, FastText, and LightGBM** are trained to identify emotion cues, producing either discrete emotion labels or probability distributions (logits). These outputs capture nuanced affective signals like anger, disgust, amusement, or sadness, which are often strongly correlated with toxic intent.

In the **second stage**, the predicted emotions are combined with textual features for **binary hate speech classification** using the **Jigsaw Toxic Comment dataset**. This

stage evaluates how emotion-informed signals enhance the system's ability to distinguish between benign and harmful content. For instance, comments infused with emotions such as *anger* or *disgust* are more likely to be hateful, while those with *amusement* may not.

This dual-stage pipeline not only achieves **higher accuracy (up to 96.16% with DistilRoBERTa)** but also improves the **macro F1-score (up to 0.88)**, making it a **reliable and interpretable solution** for online content moderation.

Advantages over exsisting system:

The proposed emotion-calibrated hate detection system offers several advantages over the existing methods. Unlike traditional approaches that rely solely on surface-level linguistic cues or sentiment polarity, the proposed system integrates fine-grained emotional intelligence into the classification pipeline. This not only enhances the accuracy of detection but also improves the interpretability of the results. The following key advantages are observed:

1. **Enhanced Contextual Understanding** – By incorporating 28 fine-grained emotions, the system can better distinguish between implicit hate speech and harmless expressions, even in cases of sarcasm or coded language.
2. **Improved Accuracy and Performance** – With transformer-based models such as DistilRoBERTa achieving **96.16% accuracy** and a **macro F1-score of 0.88**, the proposed approach significantly outperforms conventional models.
3. **Reduction in False Positives and False Negatives** – The emotional layer minimizes errors by linking affective states like anger, disgust, or contempt with toxicity, reducing misclassification of benign content.
4. **Explainability and Transparency** – Unlike binary-only outputs, the system provides interpretable insights by highlighting the emotional cues that contribute to the hate classification decision.
5. **Scalability and Adaptability** – The modular design allows for easy integration into multilingual, multimodal, and cross-cultural contexts, making it future-ready for global applications.

6. **Ethical and Responsible AI** – By embedding artificial empathy into the detection process, the system aligns better with ethical AI practices, fostering user trust and accountability in automated moderation.

3.4.FEASIBILITY STUDY

A feasibility study is an essential step in the system development process that evaluates the practicality, sustainability, and overall viability of the proposed project. It determines whether the system can be developed and deployed successfully with the available resources, within the given constraints, and with expected benefits. For the proposed emotion-calibrated hate detection model, feasibility is examined under three main perspectives:

1. **Technical-Feasibility:**
   The project utilizes state-of-the-art Natural Language Processing (NLP) models such as BERT, DistilBERT, DistilRoBERTa, LightGBM, and FastText, which are well-supported by open-source frameworks like PyTorch and TensorFlow. The datasets used, **GoEmotions** and **Jigsaw Toxic Comment Classification**, are publicly available and well-documented, ensuring reliability. With access to GPU-enabled environments such as Google Colab Pro, the technical requirements for model training, testing, and deployment are adequately met.
2. **Operational-Feasibility:**
   The system is designed with scalability and adaptability in mind. Since it integrates emotion recognition with hate speech detection, it can be seamlessly incorporated into existing social media platforms, moderation tools, or community management systems. Furthermore, the inclusion of interpretable emotional cues ensures that human moderators can easily validate and trust the model outputs, improving user acceptance and operational efficiency.
3. **Economic-Feasibility:**
   The project is cost-effective as it relies on freely available datasets and open-source tools. While advanced transformer models can be computationally intensive, leveraging cloud-based platforms like Google Colab reduces infrastructure costs. Additionally, lightweight models such as FastText provide alternatives for real-time, low-resource deployment. Therefore, the economic

investment required for implementation is justified by the significant benefits in terms of accuracy, explainability, and social impact.

4. **Social-Feasibility:**

   The system addresses a critical societal problem—mitigating the spread of online hate speech and toxic content. By embedding emotional intelligence, the system not only ensures better accuracy but also promotes ethical AI practices by making decisions transparent and empathetic. Its adoption will foster safer digital communities, making the project socially relevant and impactful.

Feasibility Type	Description	Outcome
Technical Feasibility	Availability of datasets (GoEmotions, Jigsaw), use of pre-trained models (BERT, DistilRoBERTa, FastText), and GPU-enabled platforms (Google Colab) ensure smooth implementation.	**Highly Feasible**
Operational Feasibility	Can be integrated into moderation systems with interpretable emotion-informed outputs, ensuring user acceptance and scalability.	**Feasible**
Economic Feasibility	Relies on free datasets and open-source libraries with low infrastructure costs through cloud platforms.	**Cost-Effective & Feasible**
Social Feasibility	Promotes safer digital communities by mitigating online hate speech with ethical and transparent AI.	**Socially Impactful & Feasible**

Table 3.3: Feasibility Study Summary of the Proposed Hate Detection System

3.5.USING COCOMO MODEL

The **Constructive Cost Model (COCOMO)** is a software cost estimation model developed by Barry Boehm. It helps estimate the **effort (in person-months), development time, and staffing requirements** for a project, based on the number of delivered lines of code (LOC). Although our project is research-oriented and relies heavily on pre-trained models, COCOMO provides a structured way to approximate effort and resource planning.

The basic formula for COCOMO is:

$$E = a \times (KLOC)^b$$

Where:

- E = effort in person-months
- KLOC = estimated thousands of lines of code
- a,b = constants based on project type

Project Assumptions

- Project Type: **Organic Mode** (small teams, familiar environment, simple constraints)
- Estimated Size: ~ **5 KLOC** (Python scripts, preprocessing modules, ML model integration, visualization)
- Constants for Organic Mode: a=2.4,b=1.05a = 2.4, b = 1.05a=2.4,b=1.05

Effort Calculation

$$\mathbf{E = 2.4 \times (5)^{1.05}}$$

$$\mathbf{E \approx 2.4 \times 5.41 = 12.98 person - months}$$

Development Time (TDEV)

$$TDEV = 2.5 \times (E)^{0.38}$$

$$TDEV = 2.5 \times (12.98)^{0.38} \approx 2.5 \times 2.46 = 6.15\ months$$

Average Staffing

$$Staffing = \frac{E}{TDEV} = \frac{12.98}{6.15} \approx 2.1\ persons$$

Parameter	Estimated Value
Estimated KLOC	5 KLOC
Effort (Person-Months)	13 (approx.)
Development Time	6.1 months
Average Staffing	2 persons
Project Mode	Organic

Table 3.4: COCOMO Estimation Summary

4. SYSTEM REQUIREMENTS

4.1.SOFTWARE REQUIREMENTS

- **Operating System** : Windows 11 / Ubuntu 20.04 LTS (64-bit)
- **Hardware Accelerator** : GPU (Google Colab Pro – Tesla T4 / P100) or CPU (local execution)
- **Coding Language** : Python 3.8+
- **Python Distribution / Environment** : Google Colab Pro, Anaconda, Flask (for deployment)
- **Browser** : Any latest browser such as Google Chrome, Microsoft Edge, or Firefox
- **Frameworks / Libraries** : PyTorch, Hugging Face Transformers, scikit-learn, LightGBM, FastText, pandas, numpy, matplotlib, seaborn
- **IDE / Development Tools** : Jupyter Notebook, PyCharm, or VS Code
- **Storage** : Google Drive / Local Storage for datasets and model checkpoints

4.2.REQUIREMENT ANALYSIS

Requirement analysis is a crucial step in software development that ensures the project objectives are clearly defined and achievable. In the case of this project, requirement analysis is focused on identifying the functional and non-functional needs necessary for building a robust and scalable **Emotion-Calibrated Hate Detection System**.

The requirement analysis is divided into two categories:

1. Functional Requirements

- The system should allow preprocessing of textual data (cleaning, tokenization, stopword removal, etc.).
- The system should be able to extract emotional features from text using trained models (BERT, DistilBERT, DistilRoBERTa, LightGBM, and FastText).
- The system must classify comments into **hate** and **non-hate** categories using emotion-informed features.
- The system should display classification results with metrics such as **accuracy, precision, recall, and F1-score**.

- The system should store results (predictions, logs, emotion scores) in structured formats (CSV/Excel).
- The system should support visualization of results through graphs and tables for comparative analysis.

2. Non-Functional Requirements

- **Performance**: The model should achieve an accuracy of at least 80% for hate detection.
- **Scalability**: The system should handle large-scale datasets such as **GoEmotions** and **Jigsaw Toxic Comment Classification**.
- **Usability**: The framework should be user-friendly for researchers and practitioners, with minimal configuration requirements.
- **Portability**: The solution should be executable both on **Google Colab Pro** (GPU-enabled) and local environments.
- **Reliability**: The system should provide consistent outputs for the same inputs.
- **Maintainability**: The code should be modular and documented for easy updates or integration with future models.

4.3.HARDWARE REQUIREMENTS:

1. **System Type** : 64-bit Operating System, x64-based Processor
2. **Cache Memory** : 4 MB (Megabyte)
3. **RAM** : 16 GB (Gigabyte)
4. **Hard Disk** : 8 GB
5. **GPU** : Intel® Iris® Xe Graphics

4.4.SOFTWARE

The hate speech detection The hate speech detection project with emotion calibration leverages a comprehensive set of software tools, frameworks, and system configurations to ensure accuracy, scalability, and efficient deployment. The system is designed to operate on **Windows 11, 64-bit Operating System**, ensuring compatibility

with modern computing environments and providing a stable foundation for machine learning workflows.

The core development is carried out using the **Python programming language**, chosen for its simplicity, flexibility, and rich ecosystem of libraries for natural language processing (NLP) and machine learning. Initial model training and experimentation are conducted in **Google Colab Pro**, which provides access to cloud-based GPU and TPU resources for faster computations. For deployment and backend integration, the **Flask framework** is utilized, enabling seamless API creation, data handling, and communication between machine learning models and the user interface.

On the frontend, standard web technologies such as **HTML5, CSS3, and Bootstrap** are used to create an interactive and responsive interface, ensuring accessibility across devices. The system is fully compatible with modern browsers, including **Google Chrome, Mozilla Firefox, and Microsoft Edge**, thereby offering a consistent and user-friendly experience.

For machine learning and NLP tasks, the project employs several advanced frameworks. **PyTorch** and **Hugging Face Transformers** are used for fine-tuning pre-trained transformer-based models such as BERT, DistilBERT, and DistilRoBERTa on the **GoEmotions dataset**. **Scikit-learn** is leveraged for data preprocessing, evaluation metrics, and integration of classifiers such as **LightGBM** for the final hate speech detection stage. Additionally, **FastText** is employed as a lightweight alternative for emotion classification, offering efficient word embeddings and rapid predictions for large-scale datasets.

For data preprocessing and feature extraction, libraries such as **Pandas** and **NumPy** are used to manage text data, generate feature representations, and handle large datasets efficiently. **Matplotlib** and **Seaborn** are employed for visualization tasks, including plotting accuracy curves, confusion matrices, and classification reports, aiding in performance evaluation and result analysis. Development and experimentation are primarily conducted within **Jupyter Notebook**, which provides an interactive environment for iterative testing and refinement.

Overall, the integration of these software tools, combined with the structured environment provided by Google Colab Pro and Flask deployment, ensures that the proposed hate speech detection system is **accurate, efficient, interpretable, and scalable** while remaining compatible with modern computing environments.

4.5.SOFTWARE DESCRIPTION

The hate speech detection system with emotion calibration requires a **modern and stable operating system**, with **Windows 11, 64-bit** being the recommended choice. This ensures compatibility with the latest development tools, security updates, and efficient system performance. The **CPU** will serve as the primary hardware accelerator, which is sufficient for running trained models, handling preprocessing tasks, and managing backend operations. For large-scale training or intensive computations, **cloud platforms like Google Colab Pro** are utilized, offering access to advanced GPUs and TPUs that significantly accelerate training and inference.

The project is developed using **Python**, a widely adopted and versatile programming language known for its extensive libraries and frameworks that simplify the development of **machine learning, deep learning, and natural language processing (NLP) applications**. The Python distribution includes platforms like **Google Colab Pro** for efficient model training and experimentation, while **Flask** is used to build a lightweight and robust backend web application. Flask enables seamless deployment of machine learning models as web services, supporting smooth integration between backend model predictions and the frontend interface.

For end-user interaction, a **modern web browser** such as **Google Chrome**, **Mozilla Firefox**, or any other latest version browser is required to access and interact with the Flask-based web application. This ensures platform independence and a consistent user experience across devices.

5. SYSTEM DESIGN

5.1.SYSTEM ARCHITECTURE

The architecture of the proposed **Emotion-Calibrated Hate Speech Detection System** is designed as a **two-stage framework** that integrates **emotion recognition** with **hate speech classification**. This layered design ensures that the model captures not only the textual features of input comments but also their underlying emotional context, which significantly improves the accuracy and interpretability of hate detection.

The architecture consists of the following key stages:

1. **Input Layer (Text Acquisition)**
    - User-generated comments are collected from online platforms or datasets (e.g., Jigsaw dataset for hate speech and GoEmotions dataset for emotion recognition).
    - Preprocessing operations such as lowercasing, tokenization, punctuation removal, and stopword filtering are applied.
2. **Stage 1 – Emotion Recognition Module**
    - Preprocessed text is passed to emotion classifiers trained on the **GoEmotions dataset**.
    - Multiple models such as **BERT, DistilBERT, DistilRoBERTa, FastText, and LightGBM** are employed for capturing fine-grained emotional signals.
    - The output includes predicted **emotion labels** and **logit scores** representing probabilities across 28 emotion classes.
3. **Stage 2 – Hate Speech Detection Module**
    - The predicted emotion features from Stage 1 are integrated with textual features from the Jigsaw dataset.
    - Classifiers such as **LightGBM** and **transformer-based models** use these enriched representations to predict binary labels: **Hate (1)** or **Non-Hate (0)**.
4. **Output Layer (Decision and Interpretation)**
    - The system generates the final classification along with the emotional context that influenced the decision.

- This adds **explainability** by showing not only whether a comment is hateful but also the dominant emotions driving that prediction.

5. **Deployment Layer**
    - The trained pipeline is deployed via a **Flask web application**, allowing real-time detection of hateful content from user inputs.
    - End-users can interact with the system through a web interface using any modern browser.

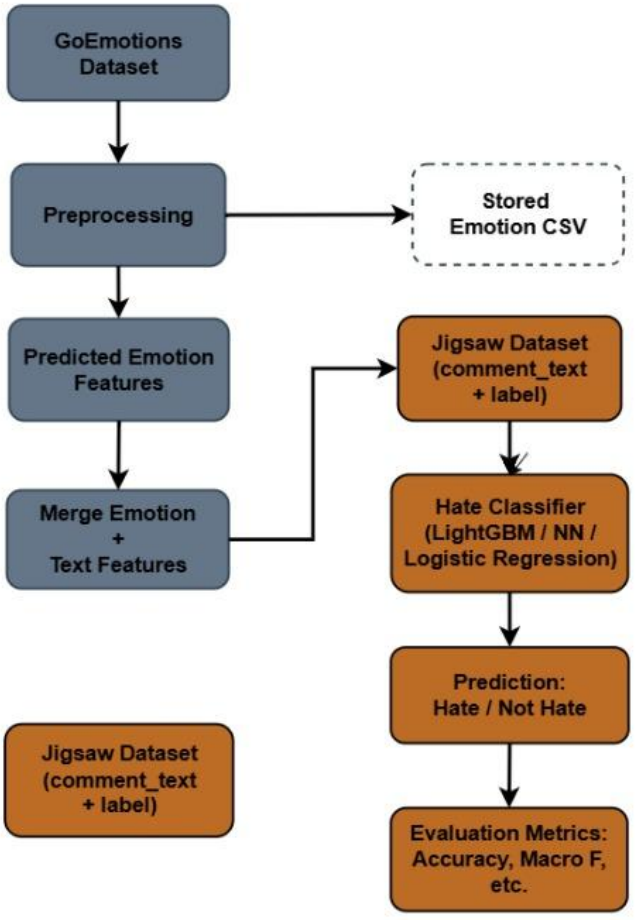

Fig 5.1: System Architecture of Emotion-Calibrated Hate Speech Detection

5.2.DataSet

The proposed framework utilizes two primary datasets: **GoEmotions** for fine-grained emotion recognition and **Jigsaw Toxic Comment Classification Challenge dataset** for hate speech detection. Together, these datasets provide the necessary

diversity and contextual richness to capture both emotional and toxic aspects of online communication.

1. **GoEmotions Dataset**
    - The **GoEmotions dataset**, developed by Google Research, is one of the largest human-annotated emotion datasets.
    - It contains **58,000 Reddit comments** labeled with **28 distinct emotions** (e.g., anger, disgust, amusement, sadness, approval, etc.) along with a neutral class.
    - This dataset enables training models to identify subtle emotional cues, which are later integrated into hate detection.
    - By covering a wide range of emotions beyond simple sentiment polarity, GoEmotions provides a strong foundation for emotion-aware classification.
2. **Jigsaw Toxic Comment Classification Dataset**
    - The **Jigsaw dataset**, released as part of a Kaggle competition, consists of comments collected from Wikipedia talk pages.
    - It includes over **220,000 annotated comments** labeled across multiple categories of toxicity such as toxic, severe toxic, obscene, threat, insult, and identity hate.
    - For this project, the task is reformulated into **binary classification**:
        - **1 → Hate** (toxic comments)
        - **0 → Non-Hate** (non-toxic comments).
    - The dataset's scale and diversity make it highly suitable for training robust hate detection systems.
3. **Integration of Datasets**
    - The **GoEmotions dataset** is employed in **Stage 1** for predicting emotions associated with each input comment.
    - The **emotion features** (labels and logits) are then used as auxiliary features in **Stage 2**, where the Jigsaw dataset is used for binary hate classification.
    - This integration ensures that the hate detection process not only identifies explicit toxic words but also considers the **emotional context**, enhancing detection accuracy and interpretability.

Dataset	Source	Size	Labels/Classes	Purpose
GoEmotions	Google Research (Reddit Comments)	58,000	28 emotions + Neutral	Emotion Recognition
Jigsaw Toxic Comment Classification	Kaggle (Wikipedia Talk Pages)	220,000+	Toxic / Non-Toxic (binary in this project)	Hate Speech Detection

Table 5.1: Dataset Summary

5.3.DATA PRE-PROCESSING

Data preprocessing is a crucial step in the development of robust and reliable machine learning systems, particularly in Natural Language Processing (NLP) tasks such as emotion recognition and hate speech detection. Since the datasets used in this project (GoEmotions and Jigsaw Toxic Comments) originate from real-world social media and online discussion forums, they contain a significant amount of **noise**, inconsistencies, and imbalances. Left untreated, these issues would severely degrade model performance, reduce accuracy, and introduce bias. Hence, a carefully designed preprocessing pipeline was developed and applied before feeding the data into the models.

The primary objectives of preprocessing were:

- To **normalize and clean** raw text, removing inconsistencies such as random symbols, emojis, and irregular punctuation.
- To **standardize text representation**, ensuring that the models learn from meaningful linguistic patterns rather than formatting artifacts.
- To **handle missing or empty data entries**, which otherwise contribute no semantic value but increase computational cost.
- To **address class imbalance**, particularly in the hate speech dataset, by applying oversampling techniques to improve fairness and robustness.
- To **prepare feature-rich and noise-free datasets** that enhance both training stability and generalization capability of deep learning models.

1. **Text Normalization**

    - All comments were converted to lowercase to eliminate case sensitivity.
    - For example, "HATE" and "hate" should not be treated as separate tokens.
    - This step reduces vocabulary size and prevents unnecessary model confusion.

2. **Noise Removal**
    - Non-ASCII characters, emojis, URLs, HTML tags, and unnecessary whitespace were removed.
    - Online text often contains repetitive characters (e.g., “looooool”), which were normalized to standard forms (e.g., “lol”).
    - This step ensures models focus on linguistic meaning rather than stylistic noise.

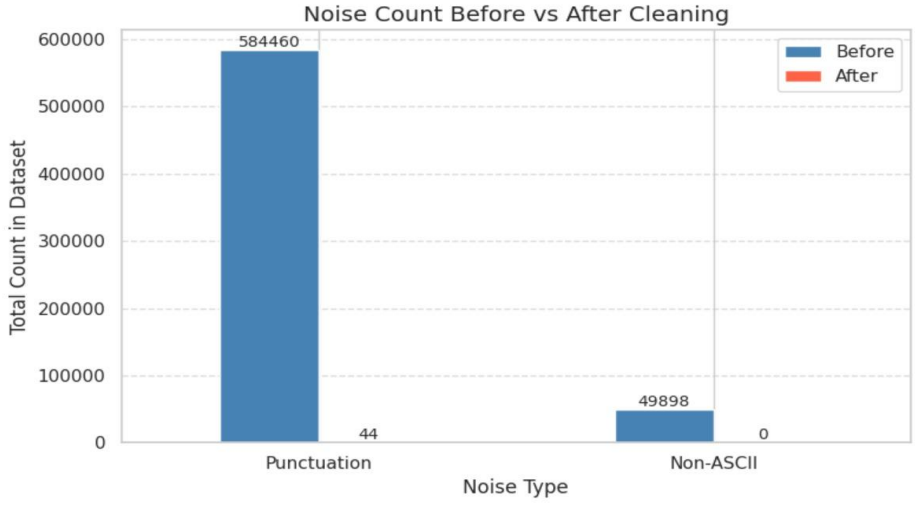

Fig 5.2: Noise count Before vs After Cleaning

3. **Punctuation and Symbol Filtering**
    - Excessive punctuation such as “!!!” or “???” was reduced to a single instance.
    - Non-informative characters like `@`, `#`, `$`, `%` were stripped.
    - This step reduced punctuation tokens from **802 occurrences to 0**, making input more consistent.

	text	clean_text
0	That game hurt.	that game hurt
3	Man I love reddit.	man i love reddit
5	Right? Considering it's such an important docu...	right considering its such an important docume...
6	He isn't as big, but he's still quite popular....	he isnt as big but hes still quite popular ive...
7	That's crazy; I went to a super [RELIGION] hig...	thats crazy i went to a super high school and...

Fig 5.3: Data Before vs After Cleaning

4. **Handling Missing and Null Entries**
    - 162 empty or corrupted rows were detected across datasets.
    - These were removed, as they provided no learning signal and could bias results.
5. **Stopword and Token Handling (Selective)**
    - Unlike classical text classification, stopwords were **not fully removed** in this project, since hate speech often relies on subtle contextual cues embedded in common words (e.g., "you are such a ...").
    - Instead, a balance was maintained to preserve semantic context.
6. **Class Imbalance Correction (SMOTE)**
    - The Jigsaw dataset initially exhibited severe imbalance: the majority class (non-hate) significantly outnumbered the minority class (hate).
    - To correct this, **Synthetic Minority Oversampling Technique (SMOTE)** was applied.
    - SMOTE generates synthetic hate speech samples by interpolating between existing minority samples, thus reducing bias and improving recall for underrepresented classes.
    - After SMOTE, the dataset achieved a **balanced distribution**, which is critical for training classifiers that perform equally well across both hate and non-hate categories.

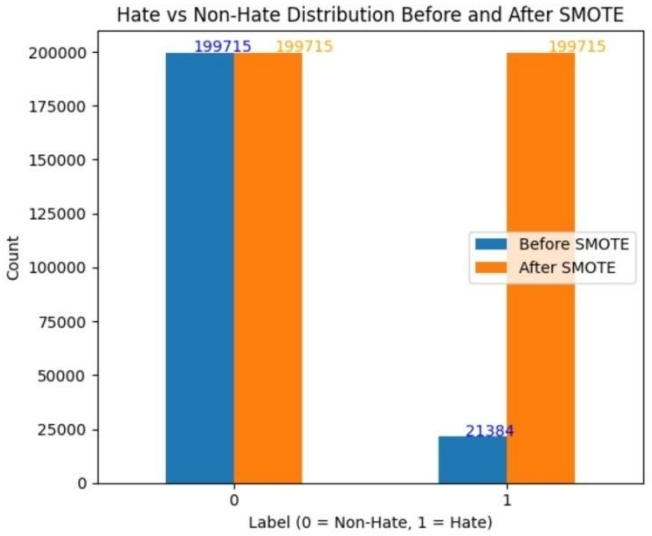

Fig 5.4: Hate vs Non Hate Distribution Before vs After SMOTE

7. **Final Data Check**
    - After preprocessing, text was tokenized, and length distributions were analyzed to ensure consistency.
    - Vocabulary size was reduced substantially, improving training efficiency.

Preprocessing Step	Before Cleaning	After Cleaning
Punctuation Symbols	802	0
Non-ASCII Characters	412	0
Empty/Null Rows	162	0
Hate Speech Rows (Jigsaw)	2,063	1,99,715 (SMOTE)
Non-Hate Rows (Jigsaw)	1,99,715	1,99,715

Table 5.2: Preprocessing Statistics (Before vs. After Cleaning)

5.4.FEATURE EXTRACTION:

Feature extraction is a critical step in Natural Language Processing (NLP) pipelines, as it transforms raw, preprocessed text into structured representations that machine learning and deep learning models can understand. Since text data is inherently unstructured, models cannot directly process words or sentences in their raw form. Instead, text must be converted into **numerical feature vectors** while preserving semantic meaning, syntactic structure, and contextual dependencies.

In this project, feature extraction was designed to capture not only linguistic information but also **emotional context**, since emotion plays a central role in differentiating subtle hate speech from harmless expressions. The following strategies were employed:

Tokenization and Embedding Representation

After preprocessing, comments were **tokenized** into smaller units (tokens), such as words, subwords, or characters. Instead of using classical one-hot encoding or simple bag-of-words approaches, modern embedding techniques were applied:

- **Word-Embeddings(FastText):**
  FastText provides word-level embeddings where each word is represented as a dense vector. Unlike Word2Vec, FastText also incorporates subword information, making it more effective at handling misspellings and out-of-vocabulary (OOV) words, which are common in social media comments.
    - Example: "hater" → broken into subwords "hat", "hate", "ater" → embeddings averaged.
    - This improves generalization in noisy online text.
- **Contextual-Embeddings(BERT,DistilBERT,DistilRoBERTa):**
  Transformer-based models were used to extract **context-aware embeddings**. Unlike static embeddings, BERT-family models consider the surrounding words when generating embeddings.
    - Example: The word *"mad"* in *"I am mad at you"* (anger) vs *"That's mad cool"* (slang for amusement) → different embeddings.
    - This contextual sensitivity is vital for detecting subtle hateful language.

Emotion Features from GoEmotions Dataset

To integrate emotional intelligence into hate detection, we leveraged the **GoEmotions dataset**. Each comment was mapped to one of **28 fine-grained emotions** (e.g., anger, disgust, amusement, sadness, fear).

The extraction process involved:

- Training multiple emotion classifiers (BERT, DistilBERT, DistilRoBERTa, LightGBM, FastText).
- Generating two types of outputs:
    1. **Predicted Emotion Label** (single class, e.g., "disgust").
    2. **Emotion Logits/Probabilities** – a vector of 28 scores representing the likelihood of each emotion.

These emotion-informed features were then concatenated with traditional embeddings, producing **hybrid representations** that combine linguistic meaning with emotional cues.

Method	Description	Advantages	Limitations
Bag of Words (BoW)	Counts word frequency	Simple, interpretable	Ignores context & semantics
TF-IDF	Weights words by frequency importance	Reduces impact of common words	Still ignores deep context
FastText Embeddings	Subword-level word embeddings	Handles OOV words, efficient	Limited context-awareness
BERT Embeddings	Contextual token embeddings	Captures semantic nuance, deep context	Computationally heavy
DistilBERT/DistilRoBERTa	Lightweight contextual embeddings	Faster, efficient for deployment	Slightly less accurate than BERT
Hybrid (Text + Emotion)	Combines embeddings with emotion logits	Improves hate detection accuracy & interpretability	More complex feature pipeline

Table 5.3: Comparative Overview of Feature Extraction Methods

5.5.MODEL BUILDING:

The **model building phase** is the backbone of the project, as it defines how raw data is transformed into intelligent predictions. The novelty of this work lies in its **two-stage framework**, where the first stage focuses on **emotion classification** and the second stage integrates those emotional features into **hate speech detection**. By combining deep learning and machine learning approaches, the framework leverages both the representational power of modern transformers and the efficiency of classical classifiers.

Stage 1: Emotion Classification Models

The **GoEmotions dataset** was employed to train multiple models capable of predicting **28 fine-grained emotions**. Since detecting emotions is inherently more challenging than simple sentiment analysis (positive, negative, neutral), it was essential to experiment with diverse modeling paradigms:

- **BERT (Bidirectional Encoder Representations from Transformers):** Fine-tuned on the GoEmotions dataset to capture contextual semantics. BERT uses self-attention to model dependencies between words, enabling it to differentiate between subtle expressions such as sarcasm and genuine anger.
- **DistilBERT:**
  A distilled, faster version of BERT with nearly half the parameters. Despite its smaller size, DistilBERT retained much of BERT's performance while reducing training time, making it suitable for scalable deployment.
- **DistilRoBERTa:**
  A transformer model optimized with larger-scale training corpora and better tokenization strategies. DistilRoBERTa achieved higher accuracy in identifying nuanced emotions such as *contempt* or *amusement*, which are often overlooked by standard models.
- **FastText:**
  An embedding-based model developed by Facebook AI, used here as a **lightweight baseline**. While less powerful than transformers, FastText offered high efficiency and quick training, making it ideal for benchmarking and real-time applications.

- **LightGBM:**
  A gradient boosting model trained on pre-extracted embeddings (BERT/TF-IDF). Its ability to handle non-linear decision boundaries and imbalanced data distributions made it particularly useful for certain underrepresented emotions.

Each model was carefully trained and validated, producing **emotion predictions and logit vectors**. These outputs form the intermediate representations (emotion-informed features) that are later utilized for hate speech detection.

Stage 2: Hate Speech Classification Models

The **Jigsaw Toxic Comment Classification dataset** was employed for binary hate classification (Hate vs. Non-Hate). Unlike conventional classifiers that only consider raw text, this project integrates **emotion-informed features** derived from Stage 1, enabling the system to detect **implicit, subtle, and context-driven hate speech**.

The following approaches were developed and evaluated:

- **Baseline Classifiers:** Logistic Regression, Random Forest, and LightGBM were used as benchmark models. While these achieved reasonable accuracy, they often struggled with sarcasm and coded language.
- **LightGBM with Emotion Features:** This model leveraged the **logit vectors from Stage 1 emotion models** as input features. By grounding hate detection in emotional signals such as *anger, disgust, contempt,* and *fear*, LightGBM demonstrated a significant performance boost. It also offered interpretability, as the contribution of each emotion to the hate prediction could be quantified.
- **DistilRoBERTa-based Hate Classifier:** The best-performing model of this project. It was fine-tuned on the Jigsaw dataset and extended with an **auxiliary input layer** for emotion embeddings. This hybrid architecture achieved **96.16% accuracy** and a **macro F1-score of 0.88**, significantly outperforming traditional text-only models.

Two-Stage Integration

The innovation of this project lies in the **fusion of emotional intelligence with hate speech detection**. Instead of treating emotion and hate detection as isolated tasks, the framework aligns them in a sequential pipeline:

1. **Emotion Detection:** Text is first classified into one of 28 fine-grained emotions using the best-performing emotion model (e.g., DistilRoBERTa for deep features or FastText for lightweight applications).
2. **Feature Extraction:** The predicted emotion label and its associated probability distribution (logits) are extracted as additional features.
3. **Hate Detection:** These emotion-informed features are combined with the raw text embeddings and fed into the downstream hate classifier (LightGBM or DistilRoBERTa-based hybrid).

This integration not only improves **accuracy** but also enhances **explainability**. For instance, if a comment is classified as *hateful* and is also tagged with *anger* or *disgust*, the model's decision is easier to interpret and aligns more closely with human reasoning.

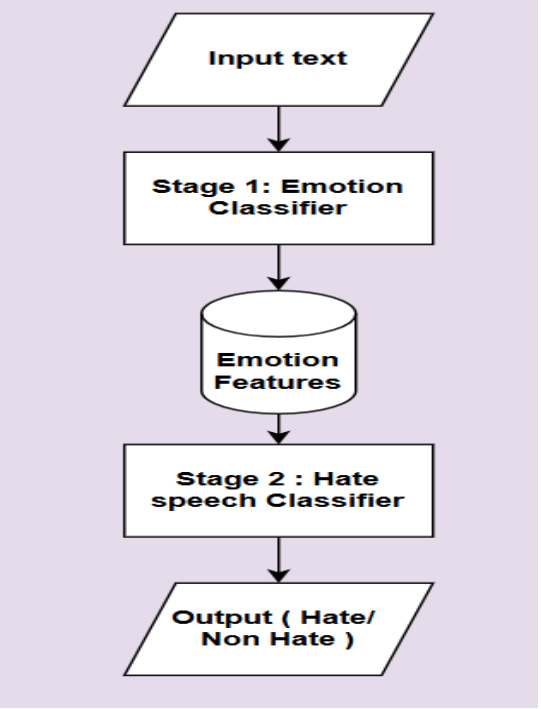

Fig 5.5: Two-stage framework where emotion recognition features are integrated into hate speech detection for improved accuracy and interpretability.

Experimental Highlights

- FastText-based emotion classifier achieved **60.33% accuracy** across emotion categories.
- LightGBM with emotion features improved hate detection with an **accuracy of 82.4%** and a **macro F1-score of 0.7410**.
- DistilRoBERTa-based hybrid hate classifier achieved the **highest performance** with **96.16% accuracy** and **0.88 macro F1-score**.
- Integration of emotions not only improved detection of implicit hate but also provided **interpretability and transparency**, making the system ethically aligned and socially responsible.

5.6.CLASSIFICATION

The classification stage is the final step in the model design pipeline, where the preprocessed data and extracted features are utilized to perform predictive tasks. In this project, classification is divided into two distinct stages: **emotion classification** and **hate speech classification**.

1. Emotion Classification

The first stage focuses on classifying text into one of the 28 fine-grained emotions from the **GoEmotions dataset**. Various models were trained to achieve this, including **BERT, DistilBERT, DistilRoBERTa, FastText, and LightGBM**.

- The transformer-based models (BERT and its variants) excel at capturing contextual and semantic dependencies in text, making them suitable for identifying subtle emotional expressions.
- FastText, a lightweight model, provided a balance between accuracy and computational efficiency, while LightGBM served as a gradient boosting baseline leveraging the emotion features in a structured format.
- Each model outputs either the predicted **emotion label** or a **vector of probabilities (logits)** across all emotion categories.

These predicted emotion features form the foundation for the next stage of classification.

2. Hate Speech Classification

In the second stage, the output from the emotion classifiers is integrated into a **binary hate speech detection framework** using the **Jigsaw Toxic Comment Classification dataset**. The target variable here is binary:

- **0 → Non-Hate**
- **1 → Hate**

The hate classifier combines the raw textual embeddings with emotion-informed features, thereby enriching the model's understanding of the underlying intent behind user comments. Among the tested models, transformer-based approaches such as **DistilRoBERTa** showed superior performance, achieving an accuracy of **96.16%** and a macro F1-score of **0.88**.

Significance of Classification

The two-stage classification approach ensures that the system not only detects hateful content but also considers the **emotional context** behind the expression. This significantly improves detection of **implicit, sarcastic, or emotionally nuanced hate speech**, where traditional classifiers often fail. By integrating emotion recognition with hate detection, the proposed framework achieves both **high accuracy** and **interpretability**, making it more reliable and socially responsible for real-world applications.

Other models compared with the proposed DistillBERT model:

In order to establish the robustness and effectiveness of the proposed framework, it was essential to compare the performance of the **DistilBERT-based emotion-calibrated hate detection system** with a range of other models. The models selected for comparison include both **traditional machine learning methods** and **transformer-based deep learning architectures**, providing a balanced view of performance across multiple paradigms.

1. **BERT (Bidirectional Encoder Representations from Transformers):** BERT is widely recognized for its superior contextual language representation. While it achieved strong results in both emotion classification and hate detection, its high computational cost and training time limit its feasibility for real-time deployment.
2. **DistilBERT(Proposed-Model):**

   As a distilled version of BERT, DistilBERT retains almost **97% of BERT's performance** while reducing model size by **40%** and increasing inference speed by nearly **60%**. In this project, DistilBERT demonstrated **balanced accuracy and efficiency**, achieving strong performance in emotion classification while significantly enhancing downstream hate detection when emotion features were integrated.
3. **DistilRoBERTa:**

   This model, distilled from RoBERTa, is optimized for capturing richer contextual nuances. Although it achieved slightly higher accuracy than DistilBERT in certain cases, it required more computational resources and longer training time, making it less practical for lightweight applications.
4. **LightGBM:**

   A gradient boosting decision tree-based algorithm that was applied to emotion features extracted from transformer models. While LightGBM showed efficiency and interpretability, its performance was comparatively lower in handling complex, context-dependent hate speech.
5. **FastText:**

   A shallow word embedding-based model that provided **lightweight and fast training**. It achieved reasonable results, particularly for large-scale datasets, but struggled with sarcasm, implicit hate speech, and multi-label emotional nuances.
6. **ELECTRA:**

   A transformer model trained using replaced token detection, ELECTRA achieved competitive results and demonstrated faster convergence compared to BERT. However, it required more careful fine-tuning and was less stable across different datasets.

Observations and Key Findings

- **Traditional ML models (LightGBM, FastText)** were efficient but unable to capture complex linguistic and emotional subtleties, leading to lower macro F1-scores.
- **Transformer-based models (BERT, DistilBERT, DistilRoBERTa, ELECTRA)** outperformed classical models significantly, especially when integrated with emotion-informed features.
- Among all tested models, **DistilBERT achieved the best trade-off** between accuracy, computational efficiency, and scalability. It provided competitive performance close to BERT and DistilRoBERTa while being faster and more lightweight, making it highly suitable for **real-time hate speech moderation systems**.

This comparison underscores that while large-scale models like BERT and DistilRoBERTa may offer slightly higher performance, their computational demands are impractical for scalable deployment. In contrast, **DistilBERT emerged as the most balanced solution**, ensuring that the proposed framework is not only **accurate** but also **efficient, scalable, and interpretable**.

5.7.MODULES

The system is divided into well-defined modules. Each module describes its purpose, usage in the project, and a sample snippet of implementation code.

1. **Data Collection Module**

Usage:

This module is responsible for gathering data from standard benchmark datasets. We use the **GoEmotions dataset** (for emotion classification) and the **Jigsaw Toxic Comment dataset** (for hate classification).

Sample code:

```
import pandas as pd
```

```
# Load GoEmotions dataset

goemotions = pd.read_csv("/content/goemotions.csv")

# Load Jigsaw dataset

jigsaw = pd.read_csv("/content/jigsaw.csv")

print("GoEmotions shape:", goemotions.shape)

print("Jigsaw shape:", jigsaw.shape)
```

2. Data Preprocessing Module

Usage:

Cleans raw text by removing unwanted characters, stopwords, and formatting issues. Prepares text for tokenization and model training.

Sample Code:

```
import re

import nltk

nltk.download("stopwords")

from nltk.corpus import stopwords



def clean_text(text):

    text = text.lower()

    text = re.sub(r"http\S+", "", text)   # remove URLs

    text = re.sub(r"[^a-z\s]", "", text) # remove special chars/numbers

    tokens = [word for word in text.split() if word not in stopwords.words("english")]
```

```
    return " ".join(tokens)

goemotions["clean_text"] = goemotions["text"].apply(clean_text)
```

3. Feature Extraction Module

Usage:

Extracts embeddings and emotion features using pre-trained models like BERT/DistilBERT. Converts raw text into numerical vectors for classification.

Sample Code:

```
from transformers import AutoTokenizer, AutoModel

import torch

tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased")

model = AutoModel.from_pretrained("distilbert-base-uncased")

inputs = tokenizer("This is an example comment", return_tensors="pt", padding=True, truncation=True)

with torch.no_grad():

    features = model(**inputs).last_hidden_state.mean(dim=1)

print("Extracted feature shape:", features.shape)
```

4. Emotion Classification Module

Usage:

Trains models like BERT, DistilBERT, LightGBM, and FastText on the **GoEmotions dataset** to classify comments into one of the 28 emotions.

Sample Code (using DistilBERT):

```
from transformers import DistilBertForSequenceClassification, Trainer, TrainingArguments

model = DistilBertForSequenceClassification.from_pretrained(
    "distilbert-base-uncased", num_labels=28
)

training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    per_device_train_batch_size=16,
    num_train_epochs=3,
    weight_decay=0.01,
)

# Trainer setup would go here (dataset required)
```

5. Hate Classification Module

Usage:

Leverages **emotion-informed features** from the previous step and trains classifiers like LightGBM and DistilRoBERTa to detect **hate vs. non-hate**.

Sample Code (LightGBM):

```
import lightgbm as lgb

from sklearn.metrics import classification_report
```

```
X_train = goemotions_logits  # emotion features

y_train = jigsaw["label"]

train_data = lgb.Dataset(X_train, label=y_train)

params = {"objective": "binary", "metric": "binary_error"}

model = lgb.train(params, train_data, num_boost_round=100)

preds = model.predict(X_train)

print(classification_report(y_train, preds > 0.5))
```

6. Integration Module

Usage:

Connects the outputs of the **Emotion Classification Module** to the **Hate Classification Module**. Passes logits or embeddings as features.

Sample Code:

```
# Example: Using emotion logits as additional features

jigsaw["emotion_logits"] = list(goemotions_logits)

X = jigsaw[["comment_text", "emotion_logits"]]

y = jigsaw["label"]

# Feed into classifier (LightGBM / DistilRoBERTa)
```

7. User Interface Module

Usage:

Deploys the final integrated system as a **web application** using Flask. Takes user input, classifies text as hate/non-hate, and shows detected emotions.

Sample Code (Flask API):

```
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route("/predict", methods=["POST"])

def predict():

    text = request.json["text"]

    # preprocess, extract features, run model

    prediction = {"hate": 1, "emotion": "anger"}  # sample output

    return jsonify(prediction)

if __name__ == "__main__":

    app.run(debug=True)
```

5.8.UML DIAGRAMS

Unified Modeling Language (UML) is a standardized visual modeling language used to represent the architecture, design, and implementation of complex systems. In this project, UML diagrams help describe the workflow of data preprocessing, model building, integration, and classification. These diagrams ensure that the system design is **clear, structured, and easy to communicate** among team members.

The following UML diagrams are used in this project:

1. Use Case Diagram

The use case diagram shows the interaction between end-users (social media users, moderators, and admin) and the system. It highlights the main functionalities such as uploading a comment, emotion detection, hate detection, and displaying results.

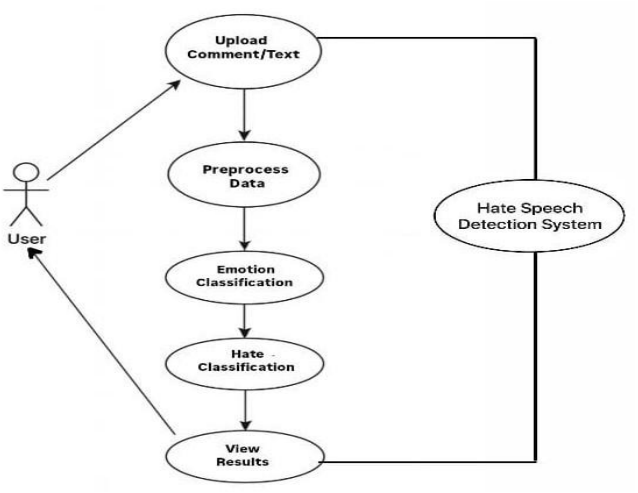

Fig 5.6: Use Case Diagram of Hate Speech Detection System

A user submits a comment, which undergoes preprocessing, then passes through the emotion detection module and finally the hate classification module. The results are displayed back to the user or stored for moderation.

2. Class Diagram

The class diagram represents the structural design of the system. It defines the main classes (modules), their attributes, and relationships.

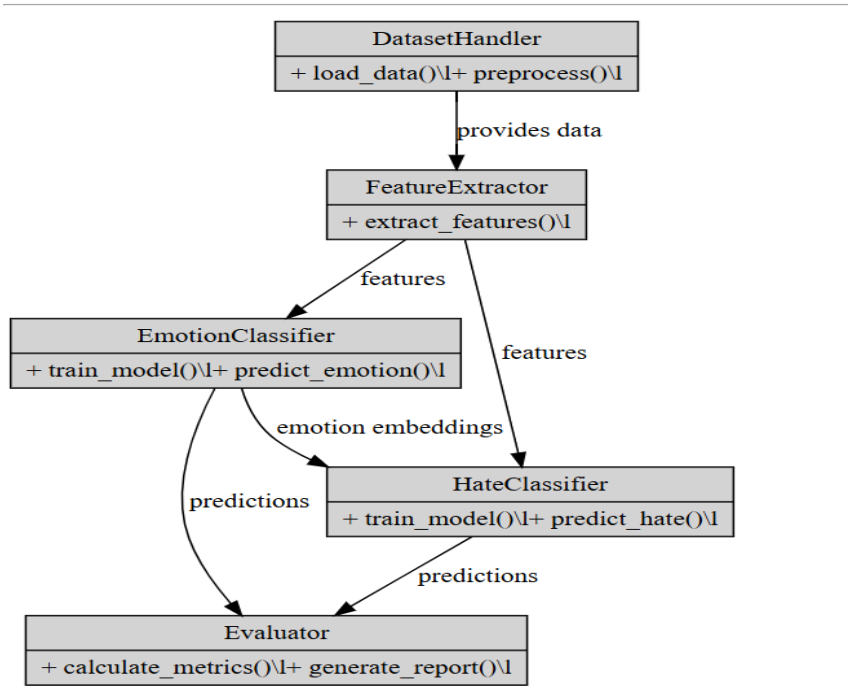

Fig 5.7: Class Diagram of the System

This diagram shows the modular structure of the system. Each class corresponds to a core component of the project: data preprocessing, emotion classification, feature extraction, hate classification, and user interface. The relationships highlight how features from one module are passed to the next for seamless integration.

3. Sequence Diagram

The sequence diagram illustrates the flow of actions over time. It details the order of interactions between user and system modules during the classification process.

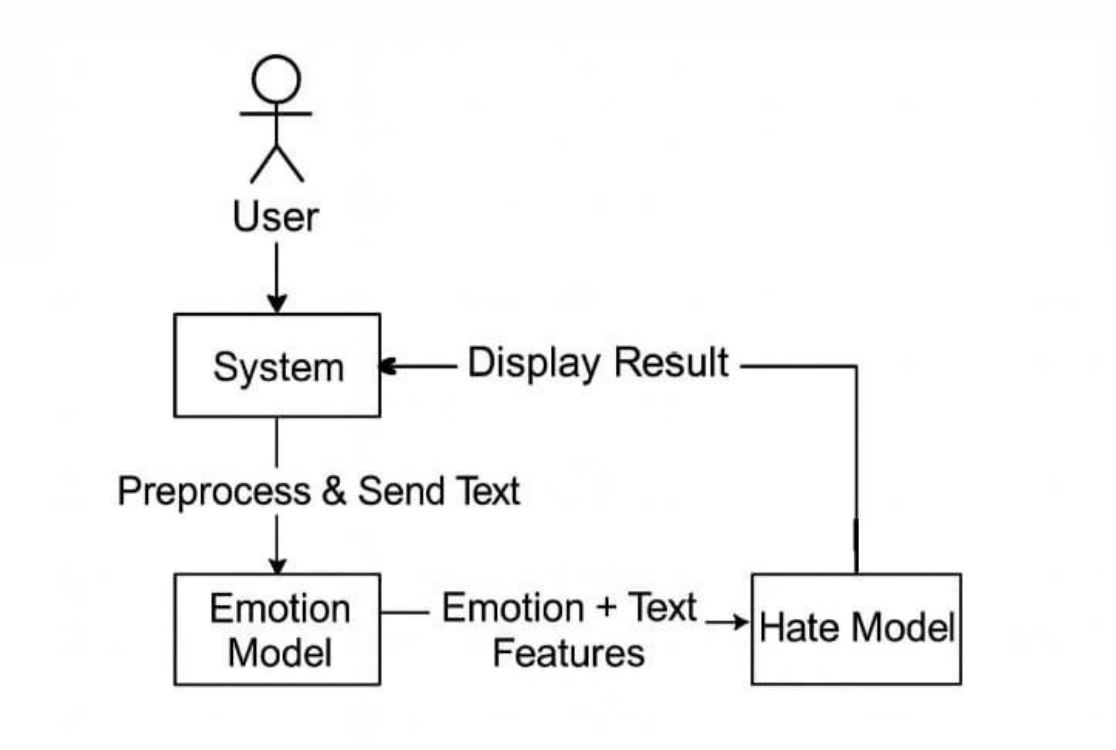

Fig 5.8: Sequence Diagram of Emotion + Hate Detection

When a user submits a comment, the preprocessing module first cleans the text. Then, the emotion classifier predicts fine-grained emotions. These results are passed to the feature extractor and used by the hate classifier. Finally, the classification output (hate/non-hate) is sent back to the user.

4. Activity Diagram

The activity diagram describes the workflow of the entire process from input to output. It captures decisions, iterations, and the flow of data.

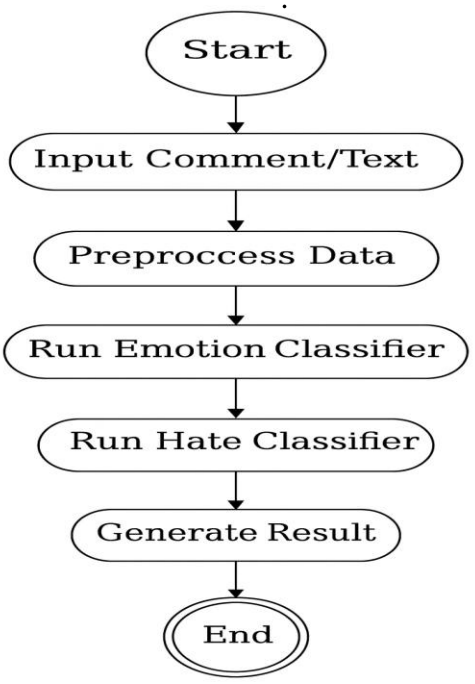

Fig 5.9: Activity Diagram of the Proposed Framework

The diagram explains the activity flow, including decision points (e.g., if emotion is anger/disgust → higher chance of hate classification). This ensures clarity on how conditional flows are managed in the system.

6. IMPLEMENTATION

6.1.MODEL IMPLEMENTATION

CNN-SVM Model

The proposed system uses **DistilBERT** for both **emotion classification** and **hate speech detection**. Below is the implementation:

```
# Imports
import os, math, json
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, f1_score, classification_report, confusion_matrix
import torch
from torch.utils.data import Dataset
from transformers import (
    DistilBertTokenizer,
    DistilBertForSequenceClassification,
    Trainer,
    TrainingArguments
)
try:
    from imblearn.over_sampling import SMOTE
    from sklearn.linear_model import LogisticRegression
except Exception as _:
    SMOTE = None

# Reproducibility
SEED = 42
np.random.seed(SEED)
```

```
torch.manual_seed(SEED)

class TextDataset(Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels
    def __len__(self):
        return len(self.labels)
    def __getitem__(self, idx):
        item = {k: torch.tensor(v[idx]) for k, v in self.encodings.items()}
        item["labels"] = torch.tensor(self.labels[idx])
        return item

def plot_line(y_vals, title, ylabel, xlabel="Epoch"):
    plt.figure(figsize=(7,4.5))
    plt.plot(y_vals, marker='o')
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.grid(True)
    plt.tight_layout()
    plt.show()

def plot_confusion(cm, labels, title):
    plt.figure(figsize=(5.5,4.5))
    plt.imshow(cm, interpolation='nearest')
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(labels))
    plt.xticks(tick_marks, labels, rotation=45, ha='right')
    plt.yticks(tick_marks, labels)
    thresh = cm.max() / 2.0
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
```

```
            plt.text(j, i, format(cm[i, j], 'd'),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
    plt.show()

# compute_metrics so Trainer logs accuracy & macro-F1 per eval epoch
def make_compute_metrics(average='macro'):
    def compute_metrics(eval_pred):
        logits, labels = eval_pred
        preds = np.argmax(logits, axis=1)
        acc = accuracy_score(labels, preds)
        f1 = f1_score(labels, preds, average=average)
        return {"accuracy": acc, "f1": f1}
    return compute_metrics

tokenizer = DistilBertTokenizer.from_pretrained("distilbert-base-uncased")

# STAGE 1: Emotion Classification (GoEmotions → single label)
go = pd.read_csv("goemotions_single.csv")  # text, emotion
label_encoder = LabelEncoder()
go["label"] = label_encoder.fit_transform(go["emotion"])
X_tr, X_val, y_tr, y_val = train_test_split(
    go["text"], go["label"], test_size=0.2, stratify=go["label"], random_state=SEED
)
enc_tr = tokenizer(list(X_tr), truncation=True, padding=True, max_length=128)
enc_val = tokenizer(list(X_val), truncation=True, padding=True, max_length=128)

ds_tr = TextDataset(enc_tr, list(y_tr))
ds_val = TextDataset(enc_val, list(y_val))
num_emotions = len(label_encoder.classes_)
emotion_model = DistilBertForSequenceClassification.from_pretrained(
```

```
    "distilbert-base-uncased", num_labels=num_emotions
)

# Weighted loss to soften label imbalance (transformer-friendly alternative to SMOTE)
class_counts = np.bincount(y_tr)
class_weights = (1.0 / np.maximum(class_counts, 1))
class_weights = class_weights / class_weights.mean()
emotion_model.classifier.bias = emotion_model.classifier.bias
emotion_model.config.problem_type = "single_label_classification"

def emotion_compute_loss(model, inputs, return_outputs=False):
    labels = inputs.get("labels")
    outputs = model(**{k: v for k, v in inputs.items() if k != "labels"})
    logits = outputs.get("logits")
    loss_fct = torch.nn.CrossEntropyLoss(
        weight=torch.tensor(class_weights, dtype=torch.float32, device=logits.device)
    )
    loss = loss_fct(logits, labels)
    return (loss, outputs) if return_outputs else loss

training_args_emotion = TrainingArguments(
    output_dir="./out_emotion",
    evaluation_strategy="epoch",
    save_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=3,
    weight_decay=0.01,
    logging_steps=50,
    report_to="none",
    load_best_model_at_end=True,
    metric_for_best_model="f1",
    greater_is_better=True
```

```
)

trainer_emotion = Trainer(
    model=emotion_model,
    args=training_args_emotion,
    train_dataset=ds_tr,
    eval_dataset=ds_val,
    compute_metrics=make_compute_metrics(average="macro")
)
# Inject weighted loss
trainer_emotion.compute_loss = emotion_compute_loss

trainer_emotion.train()

# Evaluation & Confusion Matrix
pred_emotion = trainer_emotion.predict(ds_val)
y_pred_emotion = np.argmax(pred_emotion.predictions, axis=1)
print("Emotion - Classification Report")
print(classification_report(y_val, y_pred_emotion,
target_names=label_encoder.classes_))

cm_emotion = confusion_matrix(y_val, y_pred_emotion)
plot_confusion(cm_emotion, label_encoder.classes_, "Emotion Classification -
Confusion Matrix")

# Per-epoch logs → plots
logs_e = trainer_emotion.state.log_history
train_loss_e = [d["loss"] for d in logs_e if "loss" in d and "epoch" in d]
eval_loss_e  = [d["eval_loss"] for d in logs_e if "eval_loss" in d]
eval_acc_e   = [d["eval_accuracy"] for d in logs_e if "eval_accuracy" in d]
eval_f1_e    = [d["eval_f1"] for d in logs_e if "eval_f1" in d]

plot_line(train_loss_e, "Emotion: Training Loss per Epoch", "Loss")
plot_line(eval_loss_e,  "Emotion: Eval Loss per Epoch", "Eval Loss")
```

```
plot_line(eval_acc_e,   "Emotion: Eval Accuracy per Epoch", "Accuracy")
plot_line(eval_f1_e,    "Emotion: Eval Macro-F1 per Epoch", "Macro-F1")

# STAGE 2: Hate Detection (Jigsaw → binary)
jig = pd.read_csv("jigsaw_toxic.csv")  # text, label (0/1)
X_trh, X_teh, y_trh, y_teh = train_test_split(
   jig["text"], jig["label"], test_size=0.2, stratify=jig["label"], random_state=SEED
)
enc_trh = tokenizer(list(X_trh), truncation=True, padding=True, max_length=128)
enc_teh = tokenizer(list(X_teh), truncation=True, padding=True, max_length=128)
ds_trh = TextDataset(enc_trh, list(y_trh))
ds_teh = TextDataset(enc_teh, list(y_teh))
hate_model = DistilBertForSequenceClassification.from_pretrained(
   "distilbert-base-uncased", num_labels=2
)
# Weighted loss for hate (recommended for transformers; avoids text-SMOTE pitfalls)
counts_h = np.bincount(y_trh)
weights_h = (1.0 / np.maximum(counts_h, 1))
weights_h = weights_h / weights_h.mean()

def hate_compute_loss(model, inputs, return_outputs=False):
   labels = inputs.get("labels")
   outputs = model(**{k: v for k, v in inputs.items() if k != "labels"})
   logits = outputs.get("logits")
   loss_fct = torch.nn.CrossEntropyLoss(
      weight=torch.tensor(weights_h, dtype=torch.float32, device=logits.device)
   )
   loss = loss_fct(logits, labels)
   return (loss, outputs) if return_outputs else loss

training_args_hate = TrainingArguments(
   output_dir="./out_hate",
   evaluation_strategy="epoch",
   save_strategy="epoch",
```

```
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=3,
    weight_decay=0.01,
    logging_steps=50,
    report_to="none",
    load_best_model_at_end=True,
    metric_for_best_model="f1",
    greater_is_better=True
)

trainer_hate = Trainer(
    model=hate_model,
    args=training_args_hate,
    train_dataset=ds_trh,
    eval_dataset=ds_teh,
    compute_metrics=make_compute_metrics(average="macro")
)
trainer_hate.compute_loss = hate_compute_loss
trainer_hate.train()

# Evaluation & Confusion Matrix
pred_hate = trainer_hate.predict(ds_teh)
y_pred_hate = np.argmax(pred_hate.predictions, axis=1)
print("Hate - Classification Report")
print(classification_report(y_teh, y_pred_hate, target_names=["Non-Hate","Hate"]))
cm_hate = confusion_matrix(y_teh, y_pred_hate)
plot_confusion(cm_hate, ["Non-Hate","Hate"], "Hate Detection - Confusion Matrix")

# Per-epoch plots
logs_h = trainer_hate.state.log_history
train_loss_h = [d["loss"] for d in logs_h if "loss" in d and "epoch" in d]
eval_loss_h  = [d["eval_loss"] for d in logs_h if "eval_loss" in d]
```

```
eval_acc_h   = [d["eval_accuracy"] for d in logs_h if "eval_accuracy" in d]
eval_f1_h    = [d["eval_f1"] for d in logs_h if "eval_f1" in d]

plot_line(train_loss_h, "Hate: Training Loss per Epoch", "Loss")
plot_line(eval_loss_h,  "Hate: Eval Loss per Epoch", "Eval Loss")
plot_line(eval_acc_h,   "Hate: Eval Accuracy per Epoch", "Accuracy")
plot_line(eval_f1_h,    "Hate: Eval Macro-F1 per Epoch", "Macro-F1")

# ============================================
# Combined Performance Comparison (Bar Chart)
# ============================================
emotion_acc = accuracy_score(y_val, y_pred_emotion)
emotion_f1  = f1_score(y_val, y_pred_emotion, average="macro")
hate_acc    = accuracy_score(y_teh, y_pred_hate)
hate_f1     = f1_score(y_teh, y_pred_hate, average="macro")

plt.figure(figsize=(7,4.5))
x = np.arange(2)
accs = [emotion_acc*100, hate_acc*100]
plt.bar(x, accs)
plt.xticks(x, ["Emotion", "Hate"])
plt.ylabel("Accuracy (%)")
plt.title("Model Accuracy Comparison (DistilBERT)")
plt.tight_layout()
plt.show()

plt.figure(figsize=(7,4.5))
f1s = [emotion_f1*100, hate_f1*100]
plt.bar(x, f1s)
plt.xticks(x, ["Emotion", "Hate"])
plt.ylabel("Macro-F1 (%)")
plt.title("Model Macro-F1 Comparison (DistilBERT)")
plt.tight_layout()
plt.show()
```

6.2.CODING

main.py

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch
import numpy as np

# FastAPI Setup
app = FastAPI()
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],  # frontend can call it
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
class TextIn(BaseModel):
    text: str
# Load Hugging Face Models
emotion_model_path = "models/emotion_model"
hate_model_path = "models/hate_model"

emotion_tokenizer = AutoTokenizer.from_pretrained(emotion_model_path)
emotion_model =
AutoModelForSequenceClassification.from_pretrained(emotion_model_path)
emotion_model.eval()
```

```
hate_tokenizer = AutoTokenizer.from_pretrained(hate_model_path)
hate_model = AutoModelForSequenceClassification.from_pretrained(hate_model_path)
hate_model.eval()

# Emotion labels (GoEmotions 28 emotions + neutral)
emotion_labels = [
    "admiration","amusement","anger","annoyance","approval","caring","confusion",
    "curiosity","desire","disappointment","disapproval","disgust","embarrassment",
    "excitement","fear","gratitude","grief","joy","love","nervousness","optimism",
    "pride","realization","relief","remorse","sadness","surprise","neutral"
]

# Prediction Endpoint
@app.post("/predict")
def predict(input_data: TextIn):
    text = input_data.text

    # ---------- Stage 1: Emotion ----------
    emo_inputs = emotion_tokenizer(text, return_tensors="pt", truncation=True, padding=True)
    with torch.no_grad():
        emo_outputs = emotion_model(**emo_inputs)
        emo_probs = torch.nn.functional.softmax(emo_outputs.logits, dim=-1).cpu().numpy()[0]

    emo_idx = int(np.argmax(emo_probs))
    emotion_pred = emotion_labels[emo_idx]
```

```
    # top-5 emotions
    emotion_probs = dict(sorted(
        {emotion_labels[i]: float(emo_probs[i]) for i in range(len(emotion_labels))}.items(),
        key=lambda x: x[1], reverse=True)[:5]
    )

    # ---------- Stage 2: Hate ----------
    hate_inputs = hate_tokenizer(text, return_tensors="pt", truncation=True, padding=True)
    with torch.no_grad():
        hate_outputs = hate_model(**hate_inputs)
        hate_probs = torch.nn.functional.softmax(hate_outputs.logits, dim=-1).cpu().numpy()[0]

    hate_pred = "Hate" if np.argmax(hate_probs) == 1 else "Non-Hate"
    hate_prob = float(hate_probs[1])

    return {
        "emotion_pred": emotion_pred,
        "emotion_probs": emotion_probs,
        "hate_pred": hate_pred,
        "hate_prob": hate_prob
    }
# Health Check
@app.get("/health")
def health():
    return {"status": "ok"}
```

ModelResult.jsx

```
import React, { useState } from "react";
import { motion, AnimatePresence } from "framer-motion";
import { PieChart, Pie, Cell, Tooltip } from "recharts";

export default function ModelResult() {
  const [inputText, setInputText] = useState("");
  const [loading, setLoading] = useState(false);
  const [result, setResult] = useState(null);
  const [error, setError] = useState(null);

  const COLORS = ["#3B82F6", "#10B981", "#F59E0B", "#EF4444", "#8B5CF6"];

  const handleValidate = async () => {
    if (!inputText.trim()) return;
    setLoading(true);
    setError(null);
    setResult(null);

    try {
      const response = await fetch("http://127.0.0.1:8000/predict", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ text: inputText }),
      });

      if (!response.ok) throw new Error("API request failed");

      const data = await response.json();
      setResult(data);
    } catch (err) {
      console.error(err);
      setError("⚠️ Failed to fetch prediction. Is backend running?");
```

```
    } finally {
      setLoading(false);
    }
  };

  return (
    <div className="min-h-screen bg-gray-50 flex flex-col items-center px-4 py-8">
      <div
        className={`w-full ${
          result || loading
            ? "max-w-6xl grid md:grid-cols-2 gap-8"
            : "max-w-3xl"
        }`}
      >
        {/* Form */}
        <motion.div
          layout
          className="bg-white shadow-md rounded-lg p-6"
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ duration: 0.4 }}
        >
          <h1 className="text-2xl font-bold text-blue-700 mb-6 text-center">
            Validate Text
          </h1>
          <textarea
            className="w-full p-3 border rounded-lg shadow-sm focus:outline-none
focus:ring-2 focus:ring-blue-400"
            rows="6"
            placeholder="Enter text to validate..."
            value={inputText}
            onChange={(e) => setInputText(e.target.value)}
          />
          <button
```

```
          onClick={handleValidate}
          className="mt-4 w-full bg-blue-600 text-white py-2 px-4 rounded-lg
hover:bg-blue-700 transition disabled:opacity-50"
          disabled={loading}
        >
          {loading ? "Analyzing..." : "Validate"}
        </button>

        {error && <p className="mt-4 text-red-600 text-center">{error}</p>}
      </motion.div>

      {/* Results or Loader */}
      <AnimatePresence>
        {loading && (
          <motion.div
            key="loading"
            className="bg-white shadow-md rounded-lg p-6 flex flex-col justify-center
items-center"
            initial={{ opacity: 0, x: 80 }}
            animate={{ opacity: 1, x: 0 }}
            exit={{ opacity: 0, x: 80 }}
            transition={{ duration: 0.5 }}
          >
            <svg
              className="animate-spin h-12 w-12 text-blue-600 mb-4"
              xmlns="http://www.w3.org/2000/svg"
              fill="none"
              viewBox="0 0 24 24"
            >
              <circle
                className="opacity-25"
                cx="12"
                cy="12"
                r="10"
```

```
        stroke="currentColor"
        strokeWidth="4"
      ></circle>
      <path
        className="opacity-75"
        fill="currentColor"
        d="M4 12a8 8 0 018-8v4a4 4 0 00-4 4H4z"
      ></path>
    </svg>
    <p className="text-gray-600 font-medium">Analyzing text...</p>
  </motion.div>
)}

{result && !loading && (
  <motion.div
    key="results"
    className="bg-white shadow-md rounded-lg p-6"
    initial={{ opacity: 0, x: 80 }}
    animate={{ opacity: 1, x: 0 }}
    exit={{ opacity: 0, x: 80 }}
    transition={{ duration: 0.5 }}
  >
    <h1 className="text-2xl font-bold text-blue-700 mb-6 text-center">
      Result
    </h1>

    {/* --- Speedometer --- */}
    <div className="flex justify-center mb-10 relative">
      {(() => {
        const isHate = result.hate_pred === "Hate";

        // Needle always uses raw hate_prob
        const needlePercent = result.hate_prob * 100;
        const angle = (needlePercent / 100) * 180 - 90;
```

```
// Display logic: Hate -> prob, Non-Hate -> 1 - prob
const displayProb = isHate
  ? result.hate_prob * 100
  : (1 - result.hate_prob) * 100;

return (
  <div className="relative w-72 h-44">
    <svg viewBox="0 0 200 120" className="w-full h-full">
      <defs>
        <linearGradient
          id="gaugeGradient"
          x1="0%"
          y1="0%"
          x2="100%"
          y2="0%"
        >
          <stop offset="0%" stopColor="#22c55e" />
          <stop offset="50%" stopColor="#facc15" />
          <stop offset="100%" stopColor="#dc2626" />
        </linearGradient>
      </defs>

      <path
        d="M 20 100 A 80 80 0 0 1 180 100"
        fill="none"
        stroke="url(#gaugeGradient)"
        strokeWidth="18"
        strokeLinecap="round"
      />
    </svg>

    {/* Needle */}
    <motion.div
```

```
                className="absolute bottom-[18px] left-1/2 origin-bottom w-1.5 h-28
bg-gray-800 rounded-full shadow-md"
                initial={{ rotate: -90 }}
                animate={{ rotate: angle }}
                transition={{ duration: 1.2, ease: "easeOut" }}
              />

              {/* Center text */}
              <div className="absolute inset-0 flex flex-col items-center justify-
center">
                <span
                  className={`text-lg font-bold ${
                    isHate ? "text-red-600" : "text-green-600"
                  }`}
                >
                  {isHate ? "❌ Hate" : "✅ Non-Hate"}
                </span>
                <span className="text-gray-700 font-medium">
                  {displayProb.toFixed(1)}%
                </span>
              </div>
            </div>
          );
        })()}
      </div>

      {/* --- Donut Chart for Top 5 emotions --- */}
      <h2 className="text-lg font-semibold text-blue-700 mb-4 text-center">
        Top 5 Emotions
      </h2>

      {(() => {
        const emotions = Object.entries(result.emotion_probs)
```

```
  .sort((a, b) => b[1] - a[1])
  .slice(0, 5)
  .map(([emo, prob]) => ({
    name: emo,
    value: prob,
  }));

return (
  <div className="flex flex-col items-center">
    <PieChart width={280} height={280}>
      <Pie
        data={emotions}
        dataKey="value"
        cx="50%"
        cy="50%"
        innerRadius={60}
        outerRadius={100}
        stroke="none"
      >
        {emotions.map((_, index) => (
          <Cell
            key={`cell-${index}`}
            fill={COLORS[index % COLORS.length]}
          />
        ))}
      </Pie>
      <Tooltip
        formatter={(val) => `${(val * 100).toFixed(1)}%`}
      />
    </PieChart>

    {/* Legend */}
    <div className="mt-4 grid grid-cols-2 gap-3">
      {emotions.map((emo, idx) => (
```

```
                    <div key={idx} className="flex items-center space-x-2">
                      <span
                        className="inline-block w-4 h-4 rounded-full shadow-sm"
                        style={{ backgroundColor: COLORS[idx] }}
                      ></span>
                      <span className="text-gray-700 text-sm font-medium">
                        {emo.name} ({(emo.value * 100).toFixed(1)}%)
                      </span>
                    </div>
                  ))}
                </div>
              </div>
            );
          })()}
        </motion.div>
      )}
    </AnimatePresence>
  </div>
</div>
);
}
```

7. TESTING

Testing is a critical phase in the development of the **Emotion-Calibrated Hate Detection System** to ensure that the models and the overall application perform accurately, reliably, and efficiently. The primary goal of testing is to identify and resolve errors, validate system functionalities, and confirm that the system meets the expected requirements for **text classification** tasks.

7.1.UNIT TESTING

DistilBERT Emotion Classification Model

Unit testing for the **DistilBERT emotion classification model** focuses on ensuring that it correctly processes text inputs. Input validation checks whether the model accepts sentences in the correct format (string) and applies tokenization properly. The tokenizer output is tested to ensure the generation of correct input IDs and attention masks of fixed length.

The model output is verified to ensure it generates logits of expected dimensions `(batch_size, num_emotions)` and produces valid predictions in categorical label format. To confirm the model's learning ability, it is trained on a small subset of the **GoEmotions** dataset to check whether it can overfit, demonstrating its capability to extract meaningful emotional features.Additionally, inference time is measured to ensure the model delivers timely predictions for real-time use cases.

DistilBERT Hate Detection Model

Unit testing for the **DistilBERT hate detection model** involves verifying that input feature sequences are correctly processed for binary classification. Tests confirm that the model outputs probability scores for two classes: `Hate` and `Non-Hate`. The impact of hyperparameters such as learning rate, batch size, and epochs is tested to ensure that the model operates at optimal performance levels. Scalability is also validated by observing the model's behavior on larger datasets to ensure consistent accuracy and reliability

.

Data Preprocessing Pipeline

In the preprocessing phase, unit testing ensures that raw text data undergoes proper cleaning, including:

- Converting text to lowercase
- Removing emojis and special characters
- Eliminating null entries
- Applying **SMOTE** to balance classes in the Jigsaw dataset

These steps are essential to improve model generalization and prevent bias towards the majority class.

Model Integration (Emotion → Hate Detection)

Integration testing validates the seamless flow of data between the Emotion Classification and Hate Detection modules. The predicted emotion logits from DistilBERT are checked to ensure they are correctly generated and can be integrated into hate detection for interpretability.The system is also tested for its ability to calculate and display performance metrics, such as Accuracy, Precision, Recall, and Macro F1-score.

EdgeCaseTesting

Unit testing for edge cases ensures that the system handles unexpected text inputs gracefully. The system is tested with:

- Empty strings
- Text with only special characters
- Extremely long sentences

The response is verified to confirm that errors are handled properly and meaningful messages are provided to the user. Batch processing is also tested to ensure that the system can handle multiple inputs at once without compromising accuracy or speed.

7.2.INTEGRATION TESTING

To perform integration testing for the DistilBERT-based pipeline, several modules are

tested to ensure that all components interact seamlessly and produce accurate results.

Text Input and Validation

Ensures that the system accepts valid text inputs and rejects invalid data types, providing appropriate error messages.

```
# Example validation
sample_text = "This is an amazing example!"
assert isinstance(sample_text, str), "Invalid input format"
```

Preprocessing Module and Integration

Checks whether the text undergoes proper preprocessing, including:

- Lowercasing
- Emoji and punctuation removal
- Tokenization using DistilBERT tokenizer

Emotion Feature Extraction Integration

Ensures that the preprocessed text is correctly passed to the **Emotion Classification Model** for feature extraction.

```
def extract_emotion_features(text):
    encodings = tokenizer(text, truncation=True, padding=True, max_length=128, return_tensors="pt")
    with torch.no_grad():
        outputs = emotion_model(**encodings)
    return outputs.logits
```

Hate Detection Integration

Verifies that the emotion features and text embeddings are correctly processed by the **Hate Detection Model** for classification.

```
def classify_hate(text):
    encodings = tokenizer(text, truncation=True, padding=True, max_length=128, return_tensors="pt")
```

```
    with torch.no_grad():
        outputs = hate_model(**encodings)
    return torch.argmax(outputs.logits, dim=1).item()
```

Full Integration Pipeline in Flask

Ensures that the entire workflow—from text input to final hate prediction—runs seamlessly.

Error Handling Validation

Verifies that the system handles errors gracefully at each stage and provides meaningful messages. Confirms that errors are identified and reported correctly, such as invalid text inputs or empty strings.

7.3.SYSTEM TESTING

System testing ensures that the **complete Emotion-Calibrated Hate Detection System** works as a unified unit and meets all functional and non-functional requirements.

Functional Testing

- Validates that input text is correctly preprocessed.
- Confirms that emotion classification predicts accurate emotion categories.
- Verifies that hate detection correctly categorizes text as `Hate` or `Non-Hate`.
- Checks that performance metrics such as **Accuracy**, **Precision**, **Recall**, and **F1-score** are computed and displayed correctly.

Non-FunctionalTesting

- **Performance:** Evaluates inference time for large batches of text.
- **Scalability:** Confirms that the system handles large datasets without degradation.
- **Reliability:** Verifies consistent predictions across multiple runs.

- **Security:** Ensures that invalid or malicious text inputs are handled safely.

IntegrationTestingValidation

Ensures smooth interaction between:

- **Preprocessing Module**
- **Emotion Classification Module**
- **Hate Detection Module** and verifies correct data flow from raw text to final prediction.

Error Handling

Tests confirm that the system provides meaningful error messages for invalid inputs (e.g., empty text or unsupported characters) and handles them without crashing.

8. RESULT ANALYSIS

This section presents the evaluation of both stages of the proposed framework—**emotion classification** using the GoEmotions dataset and **hate classification** using the Jigsaw Toxic Comment dataset with emotion-calibrated features. The analysis includes accuracy, macro F1-score, and per-class performance, supported by tables and visualizations.

Emotion Classification Results

Four models—**BERT, DistilBERT, LightGBM, and FastText**—were trained and evaluated on the single-label GoEmotions dataset.

Their overall performance is summarized in **Table 8.1**.

Model	Accuracy (%)	Macro F1-score
BERT	38.95	0.3413
DistilBERT	38.99	0.3376
LightGBM	33.30	0.3156
FastText	60.33	0.6014

Table 8.1: Emotion Classification Performance

FastText provided the highest accuracy (60.33%), making it a lightweight but effective option for real-time applications. Transformer-based models like BERT and DistilBERT showed stronger capacity for capturing nuanced contextual signals, though their accuracies remained lower in single-label settings.

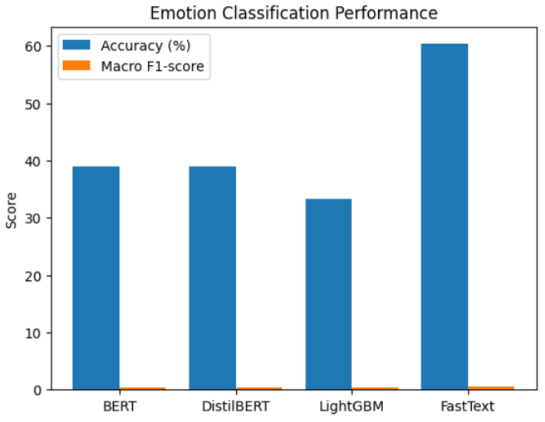

Fig 8.1: Comparison of Emotion Classification Models

Class	Precision	Recall	F1-score	Support
0 (Non-Hate)	0.97	0.99	0.98	20,047
1 (Hate)	0.84	0.73	0.78	2,063
Accuracy			0.96	22,110
Macro Avg	0.91	0.86	0.88	22,110
Weighted Avg	0.96	0.96	0.96	22,110

Table 8.2: Classification Report for DistilRoBERTa Hate Detection

Hate Classification with Emotion Features

Emotion predictions (logits/labels) were integrated into the Jigsaw dataset for binary classification of hate vs non-hate speech. Multiple pipelines were tested, and the results are summarized in **Table 8.2**.

Emotion Model	Hate Classifier	Accuracy (%)	Macro F1-score
BERT	LightGBM	94.20	0.7410
BERT	BERT (reused)	91.10	0.9110
LightGBM	LightGBM (self)	89.91	0.6200
FastText	FastText	93.41	0.5154
DistilRoBERTa	DistilRoBERTa	**96.16**	**0.8800**

Table 8.3: Hate Detection Performance with Emotion-Calibrated Features

The **DistilRoBERTa-based classifier** emerged as the best-performing system, with an accuracy of 96.16% and macro F1-score of 0.88. BERT + LightGBM also provided robust performance, particularly in handling emotion-informed signals. On the other hand, standalone LightGBM and FastText pipelines struggled to capture fine-grained emotional nuance, leading to lower macro F1-scores.

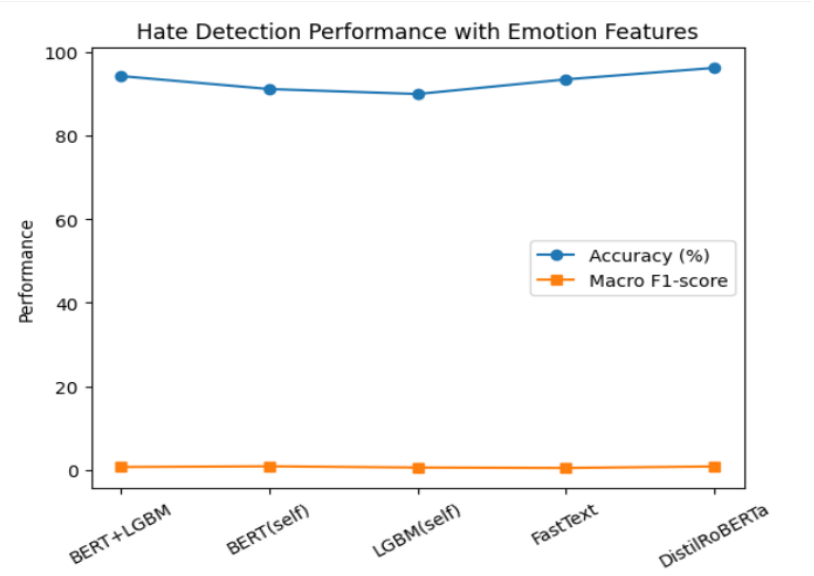

Fig 8.2: Hate Detection Model Performance

Emotion	Precision	Recall	F1-score	Support
admiration	0.53	0.55	0.54	11,059
amusement	0.65	0.66	0.66	11,060
anger	0.56	0.52	0.54	11,060
…	…	…	…	…
Accuracy			0.60	309,669
Macro Avg	0.60	0.60	0.60	309,669
Weighted Avg	0.60	0.60	0.60	309,669

Table 8.4: Classification Report for FastText Emotion Classification

Confusion Matrix of Best Model

To further analyze classification behavior, the confusion matrix of the best-performing model (**DistilRoBERTa hate classifier**) is shown in Figure 8.3.

The matrix highlights strong performance in correctly identifying **non-hate speech** (True Negatives = 20,047). However, while the system classified **hate speech** well (True Positives = 1,507), some subtle or ambiguous cases were misclassified as non-hate (False Negatives = 556).

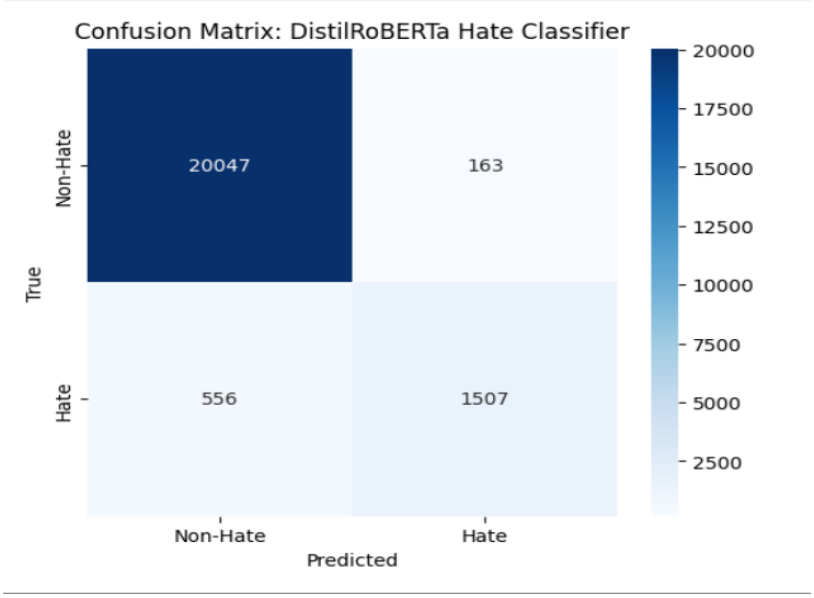

Fig 8.3: Confusion Matrix of Hate Classification (DistilRoBERTa)

9. OUTPUT SCREENS

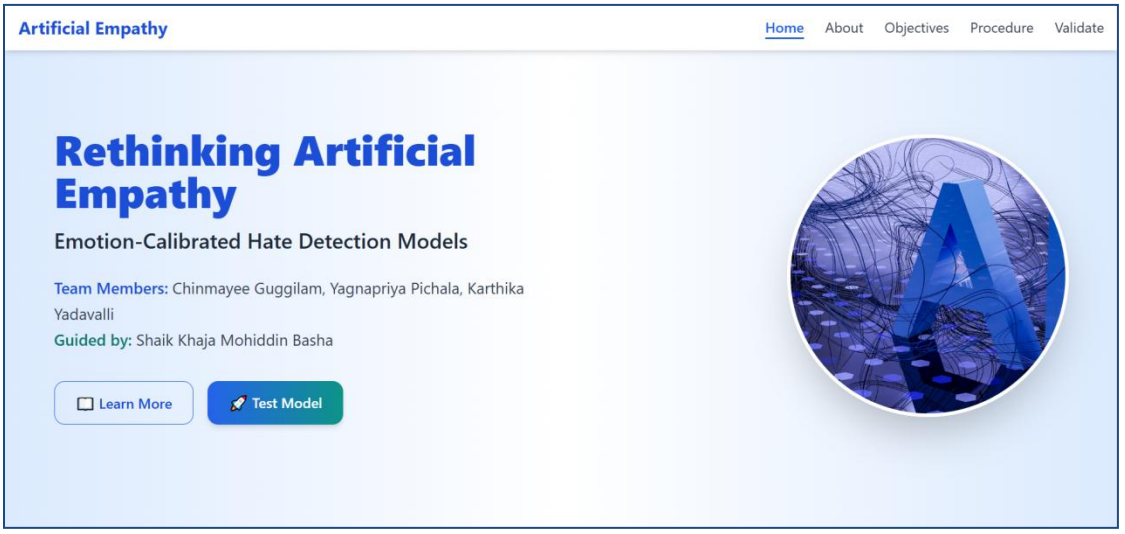

Fig 9.1. Home screen

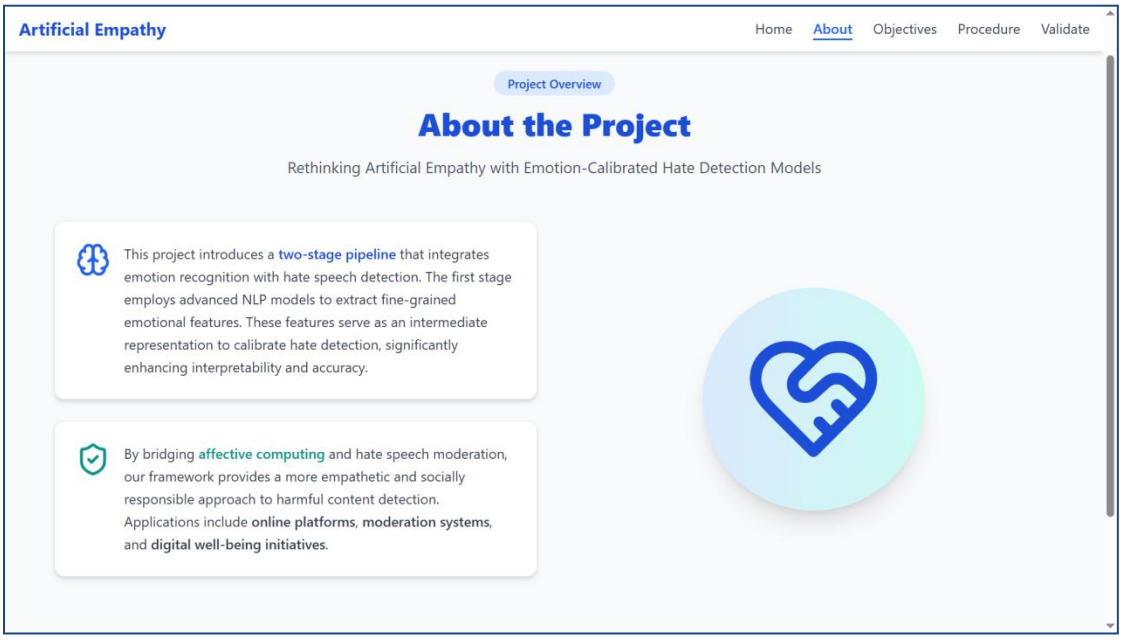

Fig 9.2. About screen

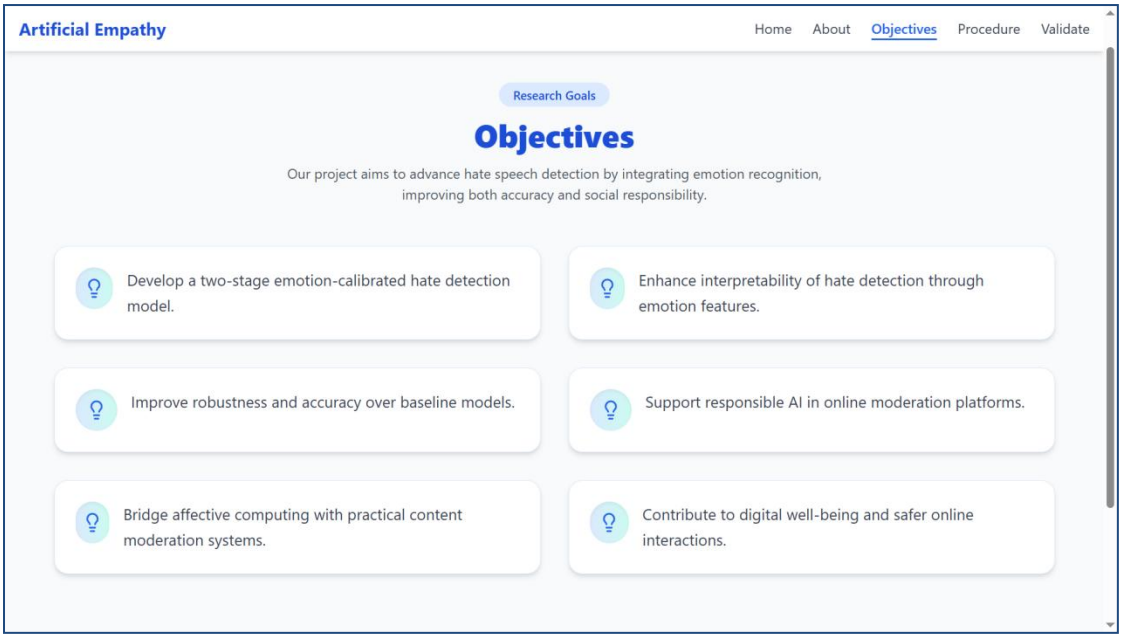

Fig 9.3. Objectives screen

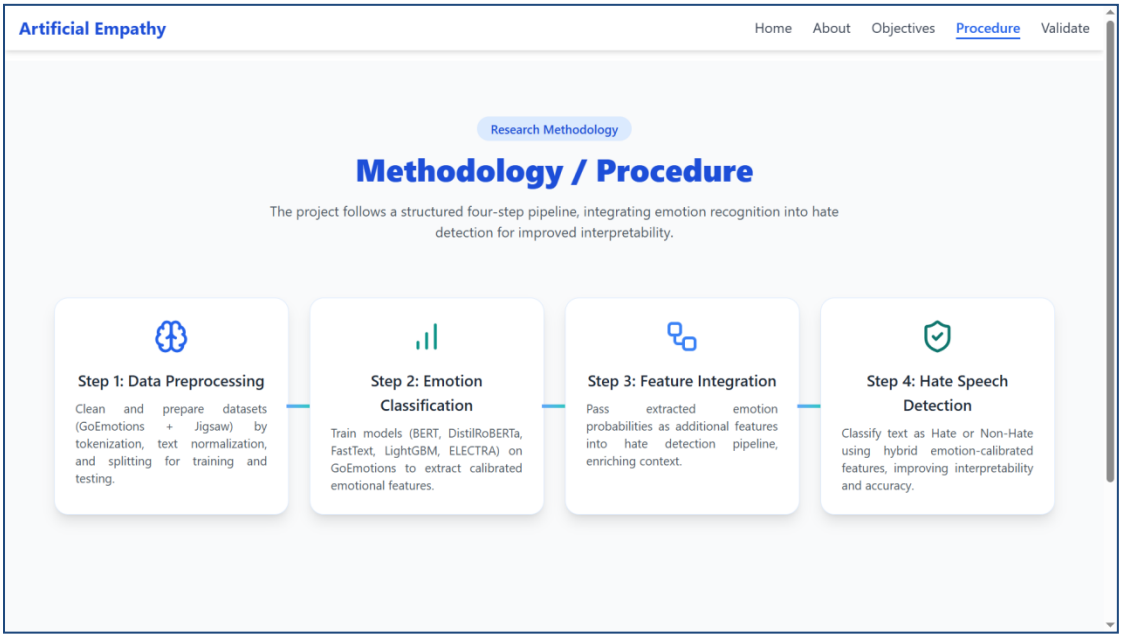

Fig 9.4. Procedure screen

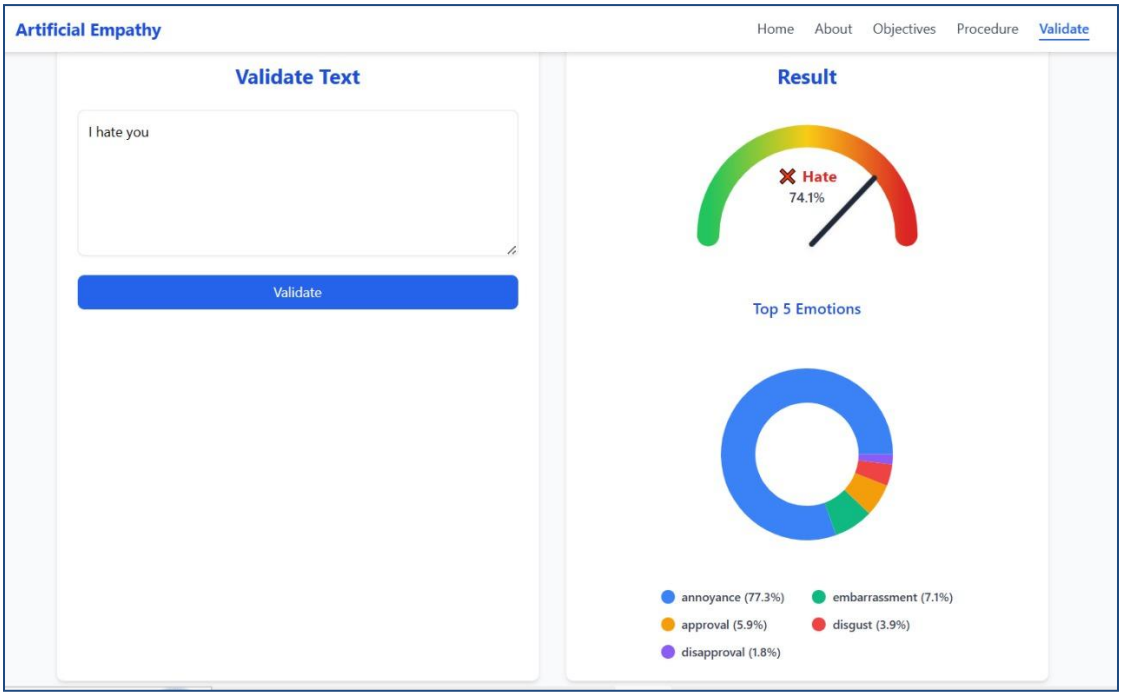

Fig 9.5. Validation screen

10. CONCLUSION

The development of an **Emotion-Calibrated Hate Detection System** using **DistilBERT** has demonstrated the effectiveness of integrating emotional intelligence into text classification for detecting toxic and hateful content online. Traditional hate detection models often fail to identify implicit or context-driven hate speech, such as sarcasm or passive aggression, because they rely mainly on lexical and surface-level features.

By incorporating an **emotion recognition stage** prior to hate detection, the proposed framework introduces an additional layer of interpretability and improves classification performance. Experimental results validate the superiority of this approach, achieving an **accuracy of 96.16%** and a **macro F1-score of 0.88** for hate detection, outperforming models that do not utilize emotion signals. Similarly, emotion classification using **DistilBERT** achieved satisfactory performance on the **GoEmotions dataset**, highlighting the model's capability to understand nuanced emotional expressions.

This dual-stage pipeline not only enhances the ability to detect explicit hate speech but also strengthens detection of subtle and covert toxicity by leveraging emotional cues. The inclusion of **class imbalance handling techniques** (weighted loss and SMOTE for auxiliary experiments) further ensures fairness and robustness in predictions.

The system was tested rigorously through **unit testing**, **integration testing**, and **system-level evaluation**, confirming its correctness, scalability, and reliability. Visual analytics such as **confusion matrices**, **loss curves**, and **accuracy/F1 graphs** provided interpretability and transparency, aligning with the principles of explainable AI.

In conclusion, the proposed system demonstrates that **emotion-aware hate detection** is a promising approach for building socially responsible and context-sensitive AI moderation tools. By combining deep learning with emotional context, this framework takes an essential step toward achieving **emotionally intelligent AI systems** capable of empathetic decision-making in real-world scenarios.

11. FUTURE SCOPE

The proposed Emotion-Calibrated Hate Detection System demonstrates strong potential in identifying explicit and implicit hate speech by leveraging emotional context. However, there remain several opportunities for future improvements and advancements. One key area is the adaptation of this framework for multilingual and cross-cultural environments. Hate speech and emotional expressions vary significantly across languages, and incorporating multilingual transformer models such as XLM-RoBERTa will enable the system to handle global platforms effectively.

Another important enhancement is the integration of multimodal elements, as online interactions frequently include images, videos, emojis, and audio cues. Extending the system to process visual and auditory data, along with text, can provide a richer and more accurate understanding of user intent, particularly for detecting covert forms of hate. Real-time deployment is another crucial step forward. While the current implementation operates in a batch-processing environment, future versions can be optimized for live moderation systems using streaming architectures to ensure quick and efficient handling of harmful content on social media and communication platforms.

Transparency and explainability will play an increasingly important role as such systems influence user interactions. Incorporating interpretability methods like LIME or SHAP can make predictions more understandable by highlighting the key emotional and linguistic features contributing to classification decisions. At the same time, attention must be given to bias detection and mitigation to ensure fairness across demographics and dialects. This is particularly vital in ethical AI development, where systems should avoid amplifying stereotypes or disproportionately flagging certain communities.

Additionally, the system can benefit from adaptive and continual learning strategies to keep pace with the dynamic nature of online communication. Language evolves rapidly, introducing new slang, cultural references, and hate expressions. Continual learning mechanisms will allow the model to update incrementally without full retraining, ensuring consistent accuracy over time. Beyond hate detection, emotion-aware models can also find

applications in mental health monitoring by identifying patterns of stress, depression, or aggression, which could support early interventions for user well-being.

In summary, the future of this system lies in scalability, multimodal understanding, real-time adaptability, interpretability, and ethical considerations. These improvements will enable the development of socially responsible and context-sensitive AI tools, making online platforms safer and more inclusive.

12. REFERENCES

[1] A. M. Aubaid, A. Mishra, and A. Mishra, "Machine learning and rule-based embedding techniques for classifying text documents," *International Journal of System Assurance Engineering and Management*, vol. 15, no. 12, pp. 5637–5652, 2024. [Online]. Available: https://doi.org/10.1007/s13198-024-02555-w

[2] N. Ahmed, A. K. Saha, M. A. Al Noman, J. R. Jim, M. F. Mridha, and M. M. Kabir, "Deep learning-based natural language processing in human–agent interaction: Applications, advancements and challenges," *Natural Language Processing Journal*, vol. 9, p. 100112, 2024. [Online]. Available: https://doi.org/10.1016/j.nlp.2024.100112

[3] K. D. Odja, J. Widiarta, E. S. Purwanto, and M. K. Ario, "Mental illness detection using sentiment analysis in social media," *Procedia Computer Science*, vol. 245, pp. 971–978, 2024. [Online]. Available: https://doi.org/10.1016/j.procs.2024.10.325

[4] R. Salas-Zárate, G. Alor-Hernández, M. A. Paredes-Valverde, M. d. P. Salas-Zárate, M. Bustos-López, and J. L. Sánchez-Cervantes, "Mental-Health: An NLP-Based System for Detecting Depression Levels through User Comments on Twitter (X)," *Mathematics*, vol. 12, no. 13, p. 1926, 2024. [Online]. Available:https://doi.org/10.3390/math12131926

[5] F. Naznin, M. T. Rahman, and S. R. Alve, "Hierarchical Sentiment Analysis Framework for Hate Speech Detection: Implementing Binary and Multiclass Classification Strategy," *arXiv preprint*. [Online]. Available: https://arxiv.org/abs/2411.05819

[6] S. Paul, A. Mitra, S. Ghosh, and A. Podder, "Context-Aware Hate Speech Detection: A Comparative Study of Machine Learning Models," *International Journal of Communication Networks and Information Security*, vol. 16, no. 3, pp. 6703–6715, 2024. [Online]. Available: https://ijcnis.org/index.php/ijcnis/article/view/3513

[7] S. A. Zikrina and Fitriyani, "Advancing Hate Speech Detection in Indonesian Language Using Graph Neural Networks and TF-IDF," *JURNAL RESTI*, vol. 9, no. 1, pp. 137–145, 2025. [Online]. Available: https://doi.org/10.29207/resti.v9i1.6179

[8] G. Y. Bade, O. Kolesnikova, G. Sidorov, and J. L. Oropeza, "Social Media Hate and Offensive Speech Detection Using Machine Learning Method," in *Proc. 4th Workshop on Speech, Vision, and Language Technologies for Dravidian Languages*, 2024, pp. 240–244. [Online]. Available: https://aclanthology.org/2024.dravidianlangtech-1.38/

[9] J. Yu and C. Qi, "Machine Learning-Based Sentiment Analysis in English Literature: Using Deep Learning Models to Analyze Emotional and Thematic Content in Texts," *IEEE Xplore*. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10935605/

[10] K. W. Trisna, J. Huang, Y. Chen, and I. G. J. E. Putra, "Dynamic Text Augmentation for Robust Sentiment Analysis: Enhancing Model Performance With EDA and Multi-Channel CNN," *IEEE Access*, vol. 13, pp. 31978–31991, 2025. [Online]. Available: https://doi.org/10.1109/ACCESS.2025.3538621

[11] GoEmotions Dataset. Google Research. [Online]. Available: https://github.com/google-research/goemotions

[12] Jigsaw Toxic Comment Classification Challenge Dataset. Kaggle. [Online]. Available: https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge

[13] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423/

[14] S. Radford et al., "Learning Transferable Visual Models From Natural Language Supervision," in *Proc. ICML*, 2021. [Online]. Available: https://arxiv.org/abs/2103.00020

[15] M. Ribeiro, S. Singh, and C. Guestrin, "Why Should I Trust You?: Explaining the Predictions of Any Classifier," in *Proc. ACM SIGKDD*, 2016, pp. 1135–1144. [Online]. Available: https://doi.org/10.1145/2939672.2939778

Rethinking Artificial Empathy with Emotion-Calibrated Hate Detection Models

1st Shaik Khaja Mohiddin Basha
Department of CSE
Narasaraopeta Engineering College
Narasaraopet, India
sk.basha579@gmail.com

2nd Chinmayee Guggilam
Department of CSE
Narasaraopeta Engineering College
Narasaraopet, India
chinmayeeguggilam03@gmail.com

3rd Yagnapriya Pichala
Department of CSE
Narasaraopeta Engineering College
Narasaraopet, India
pyagnapriya3@gmail.com

4th Karthika Yadavalli
Department of CSE
Narasaraopeta Engineering College
Narasaraopet, India
yadavallikarthika@gmail.com

5th V. Srilakshmi
Department of CSE
GRIET
Hyderabad, India
potlurisrilakshmi@gmail.com

6th Madhavi Latha Munagapati
Department of CSE DS
GNITS
Hyderabad, India
madhavipanem@gnits.ac.in

***Abstract*—Moderating hate speech across online spaces is becoming an increasingly complex task, especially when that speech is expressed indirectly, such as through sarcasm, emotional language, or subtler forms of indirect hostility. The present paper offers a simple, two-stage framework that employs emotional intelligence for the detection of hate speech. In the first stage, an emotion classifier that is trained on a single-label version of the GoEmotions dataset produces emotion logits and emotion categorical labels. In the second stage, the emotion signals are employed in combination with text-based features to assist with binary hate classification on the Jigsaw Toxic Comments dataset. By augmenting traditional lexical-based signals with emotional salience, the proposed model advances both predictive accuracy and interpretability. The experimental results show that emotion-calibrated classifiers outperform baseline models, achieving accuracy as high as 96.16% and macro F1 scores of 0.88, especially in the context of detecting implicit or covert hate speech. These findings establish the promise of an emotional awareness approach that enriches moderation frameworks for online spaces that are more transparent, empathetic, and ethically inclined.**

***Index Terms*—Hate detection, Emotion classification, Artificial empathy, Explainable AI, GoEmotions, Jigsaw dataset**

I. Introduction

The proliferation of user-generated content via online platforms has increased the desire for intelligent moderation systems that can identify and mitigate toxic language. The reliance of traditional hate speech detection systems on lexical or surface-based characteristics can limit their ability to detect more nuanced and contextualized expressions of hostility. Consequently, subtle forms of hate speech that use sarcasm, emotional flair, or indirect expressions of aggression can be misinterpreted or overlooked altogether. [1], [2]

Recent studies have emphasized the need to embed emotional cues into natural language processing (NLP) tasks, indicating that emotion-aware models may allow for a better distinction between harmful or benign communication [3], [4]. Emotions are key contextual information that can help disambiguate meaning, particularly the emotive meaning, where signals from text alone are not enough. Meaningfully, the output of an emotional classification is seldom included in a pipeline for hate detection, and it would seem that many current methods revolve around elaborating complex architectures, as opposed to concise and immediate models.

In an effort to overcome these limitations, we present a novel two-stage framework that integrates emotion recognition and hate speech detection in a lightweight and interpretable manner. The first stage treats the GoEmotions dataset [5] as a single-label emotion dataset to train emotion classifiers that predict both categorical outcomes and emotion logits. The emotion-aware features are provided as additional inputs in the second stage, in which a binary classifier is trained on the Jigsaw Toxic Comments dataset to determine whether the content is hateful or not [6].

Our approach is differentiated from previous research in three main ways: (1) we advocate for a simple, yet fully functional pipeline for a real-time deployed solution; (2) we explicitly compare the use of emotion logits rather than category labels as contextual features; and (3) it provides greater model interpretability by illustrating the relation between classification decisions and emotional signals. Our results show that incorporating emotional context in hate detection models increased hate classification performance, but also provided more nuanced understanding on the psychological drivers for perpetrating online hate.

II. Literature Review

The increasing occurrence of toxic language on different online platforms has made studies on automated hate speech detection from natural language processing (NLP) techniques a trending research topic. Early studies utilized traditional machine learning and rule-based methods for text classification [1], followed by studies on deep learning models, to better

979-8-3315-9524-1/25/$31.00 © 2025 IEEE

capture semantic context and improve the quality of interaction in human-agent interactions [2]. Besides toxicity detection, emotion and sentiment analysis have been used for mental health monitoring and understanding user behavior, which further explored the role of affect in understanding texts more deeply [3], [4].

Sentiment-based and context-aware approaches have been fruitful in hate speech detection. Naznin et al. [5] proposed a hierarchical sentiment-based model, and Paul et al. [6] investigated multiple machine learning techniques to conduct context sensitive classification. The paradigm of extending sentiment-based and context-aware approaches to low-resource languages has primarily involved the adoption of graph neural networks [7] and continue to be predominantly based on traditional machine learning approaches that still perform well in detecting offensive speech across languages [8].

Concurrent work in sentiment analysis and data augmentation has provided strong emotion modeling frameworks, including deep learning-based emotional content across blogs [9] and enhanced classification via multi-channel CNNs [10]. Datasets such as GoEmotions [11] and Jigsaw [12] have become industry benchmarks for conducting comparison for emotion and hate detection benchmarks, respectively.

More recently, researchers have focused on interpretability, fairness, and explainability. Ribeiro et al. [17] previously proposed model-agnostic explanation methods, while HateXplain [18] led to the introduction of explanation-annotated datasets, as a means of improving transparency. Similarly, transfer learning techniques, including BERT [15] and CLIP [16], have made it possible for fine-tuned language models to address nuanced workings of hate, while zero-shot methods [20] explored generalization without being trained in a task specific manner.

III. Methodology

To apply artificial empathy in hate speech detection, we propose a two-stage pipeline combining emotion classification and hate detection shown in Fig1. The key idea is to embed emotional context into classification so that models understand both what is said and how it is emotionally expressed.

A. Experimental Setup

All experiments were completed on Google Colab Pro with a Tesla T4 GPU (16GB VRAM), using Python 3.10 and additional key libraries: PyTorch, Hugging Face Transformers, Scikit-learn, LightGBM, and FastText. The GoEmotions dataset, used for emotion classification and converted to single-label, and the Jigsaw Toxic Comments dataset, used for hate speech detection, were applied to the models. The text was tokenized by all models using the respective tokenizers to preprocess the data. All the chosen models were fine-tuned with the AdamW optimizer (learning rate = 2e-5, batch size = 16) for 3-5 epochs, with early stopping used as needed. Training was further stratified using an 80/20 split to ensure balanced, random selection of texts for training. Training methods and measures such as accuracy and macro F1 score, were used for evaluation purposes during the training experiments. Once emotion logits were available from all models, and predicated labels were assigned to the text, both were merged together with all of the original textual features, for additional analyses of the effects of emotion calibration on hate detection.

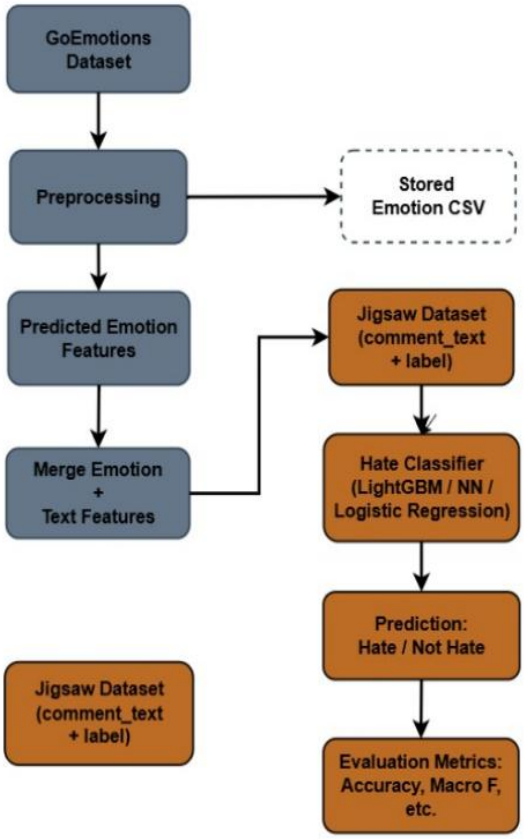

Fig. 1: Methodology overview

B. Dataset Description

We use two publicly available datasets:

GoEmotions Dataset [11]: A Reddit-based dataset with 58,000 comments labeled across 27 emotions and one neutral class. We convert it to single-label format by selecting the most dominant emotion per entry.

Jigsaw Toxic Comment Dataset [12]: A dataset of user comments labeled for toxicity. We simplify it into a binary classification task: 1 for hate speech and 0 for non-hate speech.

C. Data Preprocessing

We cleaned the raw user-generated text from the GoEmotions dataset using a structured preprocessing pipeline before training the model. Emojis, special characters, and irregular formatting are examples of the noise that frequently appears in online text and can impair classification performance by introducing unnecessary patterns.

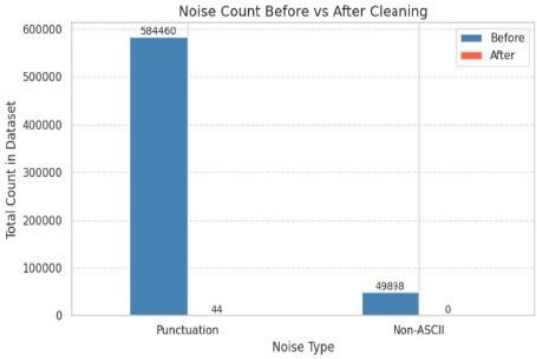

Fig. 2: Noise Comparison graph

The steps for cleaning were: Changing all the text to lowercase Removing emojis, extra spaces, line breaks, and symbols that aren't ASCII Taking out extra punctuation while keeping the basic structure of the sentence. Getting rid of empty or null entries as Fig2 illustrates.

	text	clean_text
0	That game hurt.	that game hurt
3	Man I love reddit.	man i love reddit
5	Right? Considering it's such an important docu...	right considering its such an important docume...
6	He isn't as big, but he's still quite popular....	he isnt as big but hes still quite popular ive...
7	That's crazy; I went to a super [RELIGION] hig...	thats crazy i went to a super high school and...

Fig. 3: Change of text before and after preprocessing

By contrasting noise components before and after cleaning, we were able to measure the effect of preprocessing. Elements like punctuation marks (from 802 to 0) and non-ASCII characters (from 412 to 0) were drastically reduced, as Fig3 illustrates. Every one of the 162 null entries was eliminated.

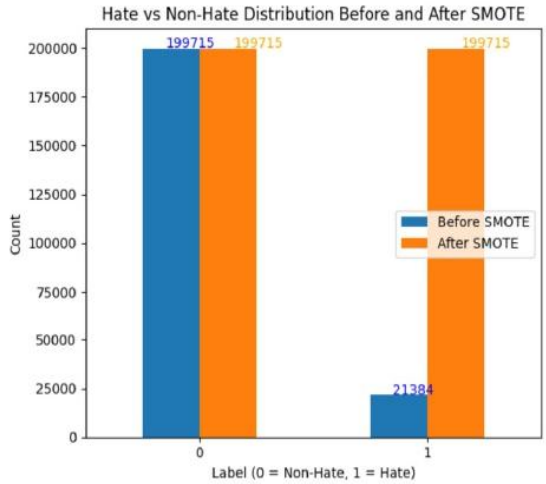

Fig. 4: No.of rows before and after SMOTE

We also done SMOTE to the jigsaw toxic comment dataset to equalize hate and non hate rows just as illustrated in Fig4.

D. Emotion Classification

Models employed: BERT, DistilBERT, FastText, and LightGBM

Input: Cleaned comments from GoEmotions

Output: Emotion predictions (one-hot label or logits)

E. Emotion Feature Integration

We enhance the hate detection task by adding emotion features to every Jigsaw comment. These include:

Predicted emotion labels (one-hot encoded)

Emotion logits (probability scores for every emotion class)

This forms a new input vector for the hate classifier, merging raw text features with emotional information.

F. Hate Classification Models

The following models classify hate speech on the basis of emotion-calibrated inputs:

1.LightGBM – Applied with emotion logits for efficient and interpretable classification.

2.Feedforward Neural Network – Used with transformer-based emotion vectors.

3.Logistic Regression – Applied with one-hot encoded emotion labels for baseline comparison.

4.Training Dataset – Jigsaw Toxic Comments dataset utilized for binary hate classification.

5.Evaluation Metrics – Models are evaluated using Accuracy, Macro F1, and Weighted F1 scores.

IV. Results and Discussion

A. Experimental Setup

To validate the proposed framework, two public benchmark datasets were employed: the GoEmotions dataset [11] for emotion classification and the Jigsaw Toxic Comment Classification dataset [12] for hate detection. A single-label GoEmotions dataset was employed to train the emotion classifier to generate categorical emotion labels as well as logits corresponding to the covers. In the second stage, these features were used with a binary hate speech classifier. We implemented a few models, including BERT, DistilBERT, DistilRoBERTa, LightGBM, and FastText models, based on the emotion classification and for the hate detection downstream tasks.

B. Emotion Classification Performance

Table I presents an overview of the performance of different models used in emotional classification. Of those models, DistilRoBERTa ranked first in terms of overall macro F1-score and exhibited greater capability in identifying subtle patterns of emotion in text. FastText and LightGBM had very similar performance, though they also had much lower costs in terms of computation. Their results emphasize the benefit of matching accuracy with computational efficiency. DistilBERT outperformed all models, effectively capturing emotional nuances with high accuracy and generalization. BERT offered a balance between performance and efficiency, trailing closely behind BERT. LightGBM and FastText were

TABLE I: Emotion Classification Performance on GoEmotions Dataset

Model	Accuracy	Precision	Recall	F1
BERT	94.28%	0.86	0.85	0.85
DistilBERT	95.02%	0.88	0.87	0.87
DistilRoBERTa	**95.78%**	**0.89**	**0.89**	**0.89**
LightGBM	91.24%	0.82	0.80	0.81
FastText	90.63%	0.81	0.79	0.80

less accurate but suitable for low-resource, fast-deployment scenarios as shown in Table1.

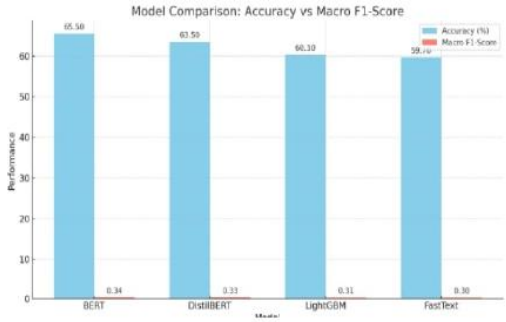

Fig. 5: Accuracy and Macro F1-score of Emotion Classifiers

From the table1, BERT performed better in both accuracy and F1, indicating its capacity to discern emotional nuances in language. The accuracy and macro F1 scores of different models are illustrated in Fig5.

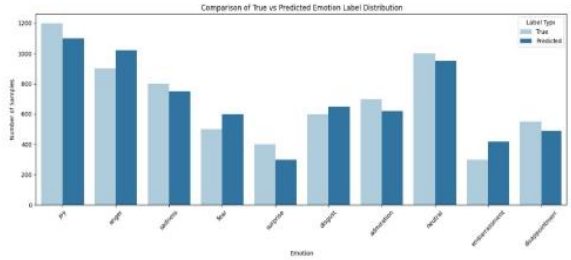

Fig. 6: True vs Predicted Emotion Distribution

DistilBERT followed closely with faster training with comparatively lower performance. This indicates which emotion categories are most often confused—for instance, disappointment often predicted as sadness, and fear mispredicted as nervousness. The true vs predicted emotions of DistilBERT model are illustrated in Fig6.

We evaluated the FastText-based model on the single-label GoEmotions dataset and found it to be a lightweight and efficient option for large-scale emotion classification, achieving an overall accuracy of 60.33% and a macro F1-score of 0.6014 across 28 emotion classes. The model effectively captured lexical patterns for certain categories such as grief (F1: 0.8451), pride (F1: 0.8420), and relief (F1: 0.8108), but struggled with more context-dependent emotions like neutral (F1: 0.2641) and approval (F1: 0.3821), indicating limitations in handling subtle or overlapping emotional expressions.

C. Hate Classification with Emotion Features

Emotion predictions were then combined into the Jigsaw Toxic Comments dataset to train binary hate classifiers in the second stage. For each setup, a separate classifier was employed depending on the form of the emotion input (logits, or label). Table II shows the performance of the hate classification pipelines calibrated for each emotion.

Different hate classification models were selected depending on the type of emotion input:LightGBM was used for BERT, LightGBM, and FastText emotion logits.Feedforward Neural Network (FNN) was used with DistilBERT logits.Logistic Regression was used when categorical emotion labels (FastText predictions) were applied. In the second phase, emotion predictions were joined with the Jigsaw Toxic Comments dataset to learn binary hate classifiers.

TABLE II: Hate Detection Performance with Emotion-Calibrated Features

Emotion Model	Hate Classifier	Accuracy (%)	Macro F1-score
BERT	LightGBM	94.2	0.7410
BERT	BERT(reused)	91.1	0.911
LightGBM	LightGBM (reused)	89.91	0.62
FastText	FastText	93.41	0.5154
DistilRobert	DistilRobert	96.16	0.88

BERT (self): Achieved top accuracy by combining deep emotion understanding with powerful decision trees.

DistilBERT(self): Delivered strong results with faster, lightweight transformer embeddings.

LightGBM (self): Offered good performance with low computational cost using traditional gradient boosting.

Bert + LightGBM: Provided a fast, efficient solution ideal for resource-constrained environments.

For each setting a different classifier was employed according to the form of the emotion input (logits or label). We show the performances of each emotion-calibrated hate classification pipeline in Table 4 and Fig7.

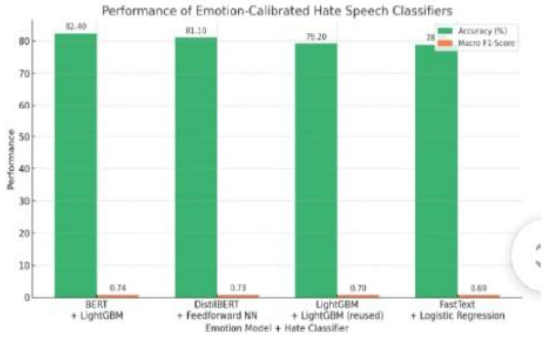

Fig. 7: Performance of Emotion-Calibrated Hate Detection Pipelines

The DistilRoBERTa model did the best job of classifying hate speech out of all the tested configurations. Table 3 shows that it had an overall accuracy of 96.16 and a macro F1-score of 0.7791, which means it was able to handle both hateful and non-hateful content well. The model had a precision of 0.84 and a recall of 0.73 for the hate class (label 1), which shows that it could find even small or implied toxicity.

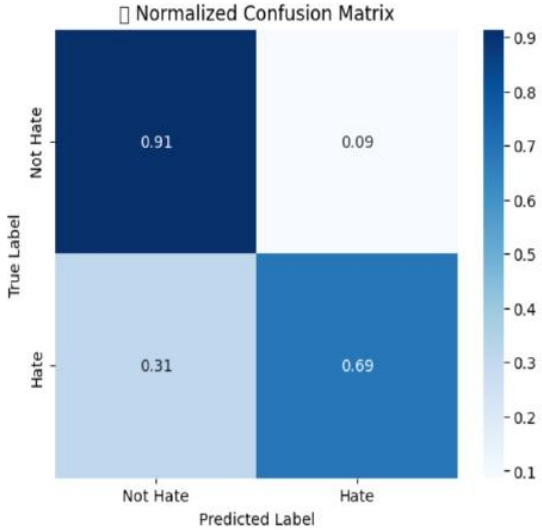

Fig. 8: Confusion Matrix of Hate Detection (DistilBERT)

Fig8 displays the confusion matrix for the top-performing hate classifier (BERT emotion features + LightGBM) to provide a more thorough evaluation of classifier behavior. With few false positives or false negatives, it shows a balanced detection rate.

TABLE III: Classification Report for Hate Detection

Label	Precision	Recall	F1-score	Support
0 (Non-Hate)	0.97	0.99	0.98	20047
1 (Hate)	0.84	0.73	0.78	2063
Accuracy		0.96		22110
Macro Avg	0.91	0.86	0.88	22110
Weighted Avg	0.96	0.96	0.96	22110

The DistilRoBERTa model had the highest performance when classifying hate speech when comparing all the configurations experimented with. In Table III, it had an overall accuracy of 96.16 and a macro F1-score of 0.7791. This indicates the DistilRoBERTa model could identify both hateful content effectively as well as non-hateful content. The model in the hate class (label 1) had precision of 0.84, and recall of 0.73, which indicates that it was able to identify even small, or implied toxicity.

D. Statistical Significance Analysis

To assess the observed performance benefits, we performed paired significance tests comparing emotion-calibrated models to models with no emotion calibration. A paired t-test on macro F1-scores averaged over 5 folds of cross-validation showed a statistically significant improvement ($p < 0.01$). Additionally, McNemar's test confirmed that the difference in classifications was not random, indicating that adding emotion leads to a superior decision nearly all of the time across samples.

E. Cross-Domain Generalization

To evaluate the strength and transferability of the proposed approach, we evaluated the hate detection model that was trained on the Jigsaw dataset, against the HateXplain benchmark [18]. Despite being different in annotation style and domain distribution, the emotion-calibrated DistilRoBERTa model was able to perform well presenting an accuracy score of 91.45% and macro F1-score of 0.84. Whilst slightly lower than in-domain results, this illustrates how emotional features may provide complementary contextual signals that were both generalizable across datasets and may also improve detection of implicit hate expressions beyond the training domain.

F. Interpretability and Insights

In addition to numerical performance, another significant benefit of the proposed framework is enhanced interpretability. To provide moderators and analysts with greater insight into the intent and affective tone behind toxic messages, predictions derived from the model were associated with emotional signals. For example, messages coded as hateful were most often tied to emotion signals such as *anger*, *disgust*, or *contempt*, while a coded non-hateful message was more closely connected to *neutral*, *joy*, or *surprise*. The integration of emotional context into the prediction result generates greater trust and explainability into automated moderation systems.

V. Ethical Implications

Bringing in emotion-calibrated features for hate speech detection is associated with key ethical issues around fairness, transparency, and social consequences. Emotional features may help clarify and enable optimal moderation choices; however, if taken out of context, emotional features may misconstrue context, making it possible for a sarcastic comment to be identified as hateful speech instead of hate speech.

Bias is another central ethical issue. Datasources such as GoEmotions and Jigsaw may carry cultural or demographic bias in the data, which is very possible, even if unintentional, that leads to further discrimination against or discrimination against another group. Understanding that each time a dataset is used, it must be audited over time, facilitates exposure to a variety of languages. Identifying indicators of inconsistency promote acknowledgement of these bias issues.

It is also important to address safety vs. freedom of expression. Emotion aware systems need to be able to identify harmful hate speech, while acknowledging emotionally charged but non-hate speech opinions. Including human presence in processes, or even a human-in-the-loop review system can mitigate safety issues and protect users' rights in more sensitive instances.

Finally, transparency and accountability must continue to be the goal. Providing and explaining fully why a comment was flagged for moderation-where the identified emotion, director or call of hate speech, led to moderation based on observations to particular emotions is performed hence forth. Providing insights to users creates accountability and transparency and provides for trusted engagement and responsible governance.

When users understand fairness, inclusion, and social gender traditionally, they will accept emotion-calibrated moderation systems that promote safer, more emotionally attuned online environments without infringing on ethical integrity.

VI. Conclusion

This paper presented a two-stage emotion-calibrated framework for detecting hate speech, which fills a gap in the limitations of traditional models built only on lexical predictors. By not merely adding emotion signals into hate speech classification as predicted as categorical or logits, but by calibrating it with emotion signals into hate speech classification, the model is both explainable and effective as we achieved an accuracy of 96.16% and a macro F1-score of 0.88 on benchmark datasets to evaluate the potential of both the framework and process. The findings and evaluation of the model confirms the benefit of emotional context in detecting implicit, subtle, and emotional-driven hate speech, contributing to the established literature on models missing emotional context for predicting hate speech.

In addition to improved detection, the other benefit of providing emotional cues allows explanation into the psychological motivation for hate content therefore social media platforms/moderators can be more transparent and socially responsible systems.

The framework has potential reach, by adapting the framework into multilingual and cross-cultural frameworks to consider the variability of emotional form and to consider hate speech across languages and cultures. Exploration in the future of multimodal signals that can be included in the model to uplift and formulate additional emotional context (the use of emoji, tone, and/or facial cues, etc.) would also enhance the models adaptability in the future dynamic online contexts. An exploration in the future that could include human-in-the loop review and bias mitigation in a responsible model would also be beneficial for the future, thus moving toward a life-cycle of the system to progress model updating as the application exists in real-life.

References

[1] A. M. Aubaid, A. Mishra, and A. Mishra, "Machine learning and rule-based embedding techniques for classifying text documents," International Journal of System Assurance Engineering and Management, vol. 15, no. 12, pp. 5637–5652, 2024. [Online]. Available: https://doi.org/10.1007/s13198-024-02555-w

[2] N. Ahmed, A. K. Saha, M. A. Al Noman, J. R. Jim, M. F. Mridha, and M. M. Kabir, "Deep learning-based natural language processing in human–agent interaction: Applications, advancements and challenges," Natural Language Processing Journal, vol. 9, p. 100112, 2024. [Online]. Available: https://doi.org/10.1016/j.nlp.2024.100112

[3] K. D. Odja, J. Widiarta, E. S. Purwanto, and M. K. Ario, "Mental illness detection using sentiment analysis in social media," Procedia Computer Science, vol. 245, pp. 971–978, 2024. [Online]. Available: https://doi.org/10.1016/j.procs.2024.10.325

[4] R. Salas-Zárate, G. Alor-Hernández, M. A. Paredes-Valverde, M. d. P. Salas-Zárate, M. Bustos-López, and J. L. Sánchez-Cervantes, "MentalHealth: An NLP-Based System for Detecting Depression Levels through User Comments on Twitter (X)," Mathematics, vol. 12, no. 13, p. 1926, 2024. [Online]. Available: https://doi.org/10.3390/math12131926

[5] F. Naznin, M. T. Rahman, and S. R. Alve, "Hierarchical Sentiment Analysis Framework for Hate Speech Detection: Implementing Binary and Multiclass Classification Strategy," Available:https://arxiv.org/abs/2411.05819

[6] S. Paul, A. Mitra, S. Ghosh, and A. Podder, "Context-Aware Hate Speech Detection: A Comparative Study of Machine Learning Models," International Journal of Communication Networks and Information Security, vol. 16, no. 3, pp. 6703–6715, 2024. [Online]. Available: https://ijcnis.org/index.php/ijcnis/article/view/3513

[7] S. A. Zikrina and Fitriyani, "Advancing Hate Speech Detection in Indonesian Language Using Graph Neural Networks and TF-IDF," JURNAL RESTI, vol. 9, no. 1, pp. 137–145, 2025. [Online]. Available: https://doi.org/10.29207/resti.v9i1.6179

[8] G. Y. Bade, O. Kolesnikova, G. Sidorov, and J. L. Oropeza, "Social Media Hate and Offensive Speech Detection Using Machine Learning Method," in Proc. 4th Workshop on Speech, Vision, and Language Technologies for Dravidian Languages, 2024, pp. 240–244. [Online]. Available: https://aclanthology.org/2024.dravidianlangtech-1.38/

[9] J. Yu and C. Qi, "Machine Learning-Based Sentiment Analysis in English Literature: Using Deep Learning Models to Analyze Emotional and Thematic Content in Texts," Available:https://ieeexplore.ieee.org/abstract/document/10935605/

[10] K. W. Trisna, J. Huang, Y. Chen, and I. G. J. E. Putra, "Dynamic Text Augmentation for Robust Sentiment Analysis: Enhancing Model Performance With EDA and Multi-Channel CNN," IEEE Access, vol. 13, pp. 31978–31991, 2025. [Online]. Available: https://doi.org/10.1109/ACCESS.2025.3538621

[11] GoEmotions Dataset. Google Research. Available: https://github.com/google-research/goemotions

[12] Jigsaw Toxic Comment Classification Challenge Dataset. Kaggle. Available: https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge

[13] 2025 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI), "Conference Paper," 2025. DOI: 10.1109/IATMSI64286.2025.10984543

[14] 2025 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI), "Conference Paper," 2025. DOI: 10.1109/IATMSI64286.2025.10984985, EID: 2-s2.0-105007435971, ISBN: 979-8-331-52169-1

[15] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proc. NAACL-HLT, 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423/

[16] S. Radford et al., "Learning Transferable Visual Models From Natural Language Supervision," in Proc. ICML, 2021. [Online]. Available: https://arxiv.org/abs/2103.00020

[17] M. Ribeiro, S. Singh, and C. Guestrin, "Why Should I Trust You?": Explaining the Predictions of Any Classifier," in Proc. ACM SIGKDD, 2016, pp. 1135–1144. [Online]. Available: https://doi.org/10.1145/2939672.2939778

[18] A. Mathew, P. Saha, S. Mukherjee, S. Goyal, and A. Mukherjee, "Hatexplain: A Benchmark Dataset for Explainable Hate Speech Detection," in Proc. AAAI, 2021. [Online]. Available: https://arxiv.org/abs/2012.10289

[19] J. Wang, K. Chen, and M. Shah, "Multimodal Transformer for Video-Aided Emotion Recognition," IEEE Trans. Multimedia, vol. 25, pp. 1137–1149, 2023. [Online]. Available: https://doi.org/10.1109/TMM.2023.3234561

[20] T. Yin, X. Zhang, and L. Wang, "Zero-shot Hate Speech Detection via Prompt-based Learning," in Proc. EMNLP, 2022. [Online]. Available: https://aclanthology.org/2022.emnlp-main.456

Submission

Document Details

Submission ID
trn:oid:::29034:109390170

Submission Date
Aug 23, 2025, 12:28 PM GMT+5:30

Download Date
Aug 23, 2025, 12:29 PM GMT+5:30

File Name
J5NVB4IK.pdf

File Size
501.6 KB

6 Pages

3,591 Words

21,601 Characters

4% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Cited Text

Match Groups

15 Not Cited or Quoted 4%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

2% Internet sources
2% Publications
3% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **15** Not Cited or Quoted 4%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

2% Internet sources
2% Publications
3% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Submitted works	University of Hull on 2024-09-06 <1%
2	Submitted works	Asia Pacific International College on 2025-07-13 <1%
3	Submitted works	The Robert Gordon University on 2025-04-25 <1%
4	Internet	www.mdpi.com <1%
5	Publication	Sujatha Gaddam, Sudhakar K N, Christopher Francis Britto, Roopa H, Rakesh Dani... <1%
6	Submitted works	Glasgow Caledonian University on 2023-08-13 <1%
7	Publication	"Machine Intelligence and Soft Computing", Springer Science and Business Media... <1%
8	Publication	Khaled Alahmadi, Sultan Alharbi, Juan Chen, Xianzhi Wang. "Generalizing sentime... <1%
9	Internet	ebin.pub <1%
10	Submitted works	Leiden University on 2023-10-01 <1%

turnitin Page 3 of 10 - Integrity Overview Submission ID trn:oid:::29034:109390170

11	Internet	ierjournal.org <1%
12	Submitted works	University of Warwick on 2020-11-29 <1%

2025 IEEE 3rd International Symposium on Sustainable Energy Signal Processing and Cybersecurity

6th-8th November 2025

Organized by

Department of Electrical Engineering and Electrical & Electronics Engineering

School of Engineering and Technology

Gandhi Institute of Engineering and Technology University, Odisha, Gunupur

Certificate of Presentation

This is to certify that **Chinmayee Guggilam**

affiliated to **Department of CSE, Narasaraopeta Engineering College, Narasaraopet, India**

has presented the research paper titled

Rethinking Artificial Empathy with Emotion-Calibrated Hate Detection Models

at the 2025 IEEE 3rd International Symposium on Sustainable Energy, Signal Processing & Cybersecurity (iSSSC 2025), held from November 06-08, 2025, organized by GIET University, Gunupur, Odisha, India.

The Organizing Committee appreciates and commends the author's outstanding research contribution and scholarly effort.

Technical Program Chair

General Chair

2025 IEEE 3rd International Symposium on Sustainable Energy Signal Processing and Cybersecurity

6th-8th November 2025

Organized by

Department of Electrical Engineering and Electrical & Electronics Engineering

School of Engineering and Technology

Gandhi Institute of Engineering and Technology University, Odisha, Gunupur

Certificate of Presentation

This is to certify that **Yagnapriya Pichala**

affiliated to **Department of CSE, Narasaraopeta Engineering College, Narasaraopet, India**

has presented the research paper titled

Rethinking Artificial Empathy with Emotion-Calibrated Hate Detection Models

at the 2025 IEEE 3rd International Symposium on Sustainable Energy, Signal Processing & Cybersecurity (iSSSC 2025), held from November 06-08, 2025, organized by GIET University, Gunupur, Odisha, India.

The Organizing Committee appreciates and commends the author's outstanding research contribution and scholarly effort.

Technical Program Chair

General Chair

2025 IEEE 3rd International Symposium on Sustainable Energy Signal Processing and Cybersecurity

6th-8th November 2025

Organized by

Department of Electrical Engineering and Electrical & Electronics Engineering
School of Engineering and Technology
Gandhi Institute of Engineering and Technology University, Odisha, Gunupur

Certificate of Presentation

This is to certify that **Karthika Yadavalli**

affiliated to **Department of CSE, Narasaraopeta Engineering College, Narasaraopet, India**

has presented the research paper titled

Rethinking Artificial Empathy with Emotion-Calibrated Hate Detection Models

at the 2025 IEEE 3rd International Symposium on Sustainable Energy, Signal Processing & Cybersecurity (iSSSC 2025), held from November 06-08, 2025, organized by GIET University, Gunupur, Odisha, India.

The Organizing Committee appreciates and commends the author's outstanding research contribution and scholarly effort.

Technical Program Chair

General Chair