VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Detecting Harmful and Offensive Language in User Content Using Language Patterns

NAME – AARON ANSON

REG NO – 22BCE3615

COURSE – NATURAL LANGUAGE PROCESSING (BCSE409L)

Faculty Name- RAJESHKANNAN R

Google Drive Link –

https://drive.google.com/drive/folders/1WnJcbuTGKhuVUbx0HkLkWPqaYVYoUA1z?usp=drive_link

Abstract

Background:
With the growing amount of user-generated content on social media, online forums, and comment sections, it is becoming increasingly difficult to manually moderate offensive or harmful content. Automated detection systems are necessary to improve the process and ensure a healthier online environment.

Research Gap or Motivation:
Previous works in offensive language detection often rely on either rule-based methods or machine learning models but fail to provide a balanced approach that can scale well while catching both explicit toxicity and creative expressions of harmful language.

Methodology:
This project proposes a pattern-based detection system that uses regex matching against a lexicon of offensive words and expressions. The lexicon is derived from datasets like OLID and Jigsaw, and it includes common and creative forms of offensive language.

Key Results:
The pattern-based model achieves a precision of 0.70, a recall of 0.48, and an accuracy of 0.76 on the OLID dataset.

Conclusion:
The system provides a fast, scalable solution for detecting harmful language and can be easily adapted to other content moderation tasks.

Introduction

Background and Relevance:

As the internet grows, so does the amount of harmful content. Offensive language detection is important for protecting online communities and ensuring a safe digital environment. Manual moderation is not feasible at scale, making automated detection systems crucial.

Review of Existing Solutions and Their Limitations:

- Rule-based Methods: Traditional methods rely on lexicons of offensive terms. They are fast but cannot catch creative variations of offensive words.
- Machine Learning Models: Recent approaches use supervised learning to detect toxic content but require large datasets and significant computational resources.

Research Gap:

A gap exists in systems that can efficiently combine speed and accuracy, particularly when it comes to detecting both explicit offensive language and creative variants (e.g., using emojis or leetspeak).

Objective or Proposed Solution:

This work proposes a pattern-based detection system that utilizes a custom lexicon of offensive terms. This approach can detect a wide range of toxic content based on literal matches to the lexicon.

Major Contributions:

1. Pattern-based Detection System: Developed using regex matching.
2. Lexicon Creation: Built from the OLID and Jigsaw datasets.
3. Evaluation: Evaluated the system's performance on the OLID and Jigsaw datasets.

Literature Review

Overview of Existing Methods in Offensive Language Detection

In recent years, detecting offensive language and hate speech has become an essential challenge in natural language processing (NLP), driven by the rise of social media platforms and online communities. Numerous approaches have been developed to automate the identification of harmful content, with a variety of methods ranging from traditional rule-based systems to deep learning-based models. These methods primarily focus on identifying toxic content, including hate speech, abusive language, and offensive remarks.

Albladi et al. (2025) provide a comprehensive review of large language models (LLMs) for hate speech detection, focusing on their applications in automated systems. The study emphasizes the power of LLMs, such as BERT and GPT-3, which capture deep contextual relationships in text and significantly outperform earlier methods based on shallow word matching or feature engineering. While these models have shown impressive results, their requirement for large datasets and computational power remains a major limitation. This raises a gap in applications where resources are constrained, highlighting the need for models that balance performance with efficiency.

Meanwhile, El-Alami et al. (2022) explore transfer learning and transformers for multilingual offensive language detection. They propose a fine-tuning model based on pre-trained transformer models like BERT and XLM-R, which can be adapted for low-resource languages. Their method demonstrates effectiveness in cross-lingual tasks, indicating that models trained on high-resource languages can be fine-tuned to perform well on languages with fewer labeled datasets. However, a significant challenge remains in adapting these models for multiple languages and ensuring scalability across various domains.

In contrast, Jahan and Oussalah (2023) provide a systematic review of hate speech detection using traditional machine learning techniques such as SVM, Random Forests, and Logistic Regression. Their findings highlight that although feature-based methods (e.g., bag-of-words, n-grams) can offer moderate success, they often fail to capture contextual nuances like sarcasm or implicit hate. Additionally, their low generalization across domains poses a limitation, especially when tested on data from different platforms or new topics.

Deep Learning Approaches for Offensive Content Detection

The deep learning models have evolved into the state-of-the-art techniques for detecting offensive language. Kaur et al. (2021) summarize the use of convolutional neural networks (CNN), recurrent neural networks (RNN), and transformers for abusive content detection. They argue that deep learning techniques have the capacity to detect both explicit and implicit abuse by learning from massive datasets. However, these models require a significant amount of labeled data for training, which is a major challenge in the domain of offensive language detection.

A significant advancement in this area is the BERT-based approach proposed by Madhavi et al. (2024), who combine CNN and GRU with BERT embeddings for hate speech classification. Their model improves the accuracy of hate speech detection by incorporating both semantic understanding from BERT and contextual learning from the hybrid CNN-GRU architecture. This methodology addresses the problem of context awareness and has demonstrated significant improvements over traditional models. Nonetheless, this approach also faces challenges due to its high computational cost and the need for large-scale labeled datasets.

Mozafari et al. (2022) focus on cross-lingual hate speech detection using meta-learning. They introduce a few-shot learning approach to tackle the challenge of insufficient labeled data in low-resource languages. Their approach leverages meta-learning techniques to adapt the training of models from high-resource languages to low-resource ones. While their results show promise, they also emphasize that scalability remains a key challenge, especially when adapting to new domains and languages.

Challenges and Limitations

Despite the progress made, current models still face several limitations. A key issue is the need for large annotated datasets, which are often not available for many languages or topics. Mody et al. (2023) present a curated dataset for hate speech detection on social media, but even with such datasets, the challenge of contextual ambiguity (e.g., sarcasm, humor) remains unsolved. While deep learning models like BERT are capable of capturing context, their resource-heavy nature is a major roadblock in real-time applications.

Moreover, Zhao et al. (2021) conduct a comparative study of using pre-trained language models for toxic comment classification. Their study reveals that while pre-trained models like BERT and RoBERTa show superior performance in detecting toxicity, they require substantial computational resources. For smaller-scale applications or real-time detection, this makes them less practical.

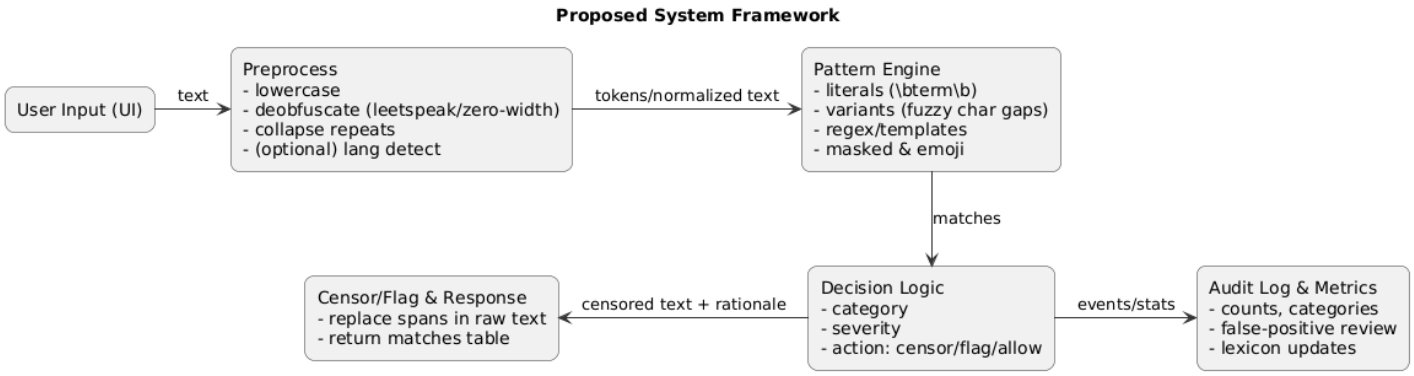
Problem Description

Leetspeak, emojis, repeated letters, or masked characters are frequently used to obscure harmful and offensive content, such as insults, slurs, threats, and xenophobic phrases, that are constantly present on online platforms. In order to overcome common evasion techniques, your project uses language patterns and a curated slang/lexicon dataset to automatically detect and selectively censor/flag such content. It also uses lightweight normalisation rules. This aligns with your introduction and the literature's framing of multilingual/code-mixed text, scaling moderation, and the necessity of a useful user interface (UI) for moderation support.

Core objective:
Design a **pattern-driven detection pipeline** that (a) normalizes noisy social text, (b) matches against a **slang/lexicon** (literals, regex/templates, masked forms, emojis), (c) assigns **severity/action** (censor/flag/allow), and (d) returns a **censored version** plus **match rationale** for transparency.
Constraints & risks (from literature):

- Slang is evolving; implicit/coded abuse and sarcasm remain hard; multilingual and code-mixed settings need coverage; fairness/explainability matter

Framework (Diagram + Explanation)



Explanation (blocks):

- **User Input (UI):** Raw text typed in the website.
- **Preprocess:** Lowercasing; leetspeak/emoji normalization; collapse repeated letters; remove zero-width chars.
- **Pattern Engine:**
    - **Literal** terms (exact, with word boundaries).

    - **Variant** patterns (allow light noise/obfuscation).
    - **Regex/Template** (e.g., “go back to <X>”).
    - **Masked/emoji** patterns (e.g., b****, 🤬).
- **Decision Logic:** Map each match to **category**, **severity**, and **action** (censor/flag).
- **Censor/Flag & Response:** Return censored text + a table of matched patterns.
- **Audit & Metrics:** log matches for analysis and lexicon updates.

Psuedocode of the Proposed system

```
INPUT: text, lexicon_rows (term, category, severity, action, pattern_type, regex,
       locale)


function NORMALIZE(text):
  t = lowercase(text)
  t = leetspeak_to_alpha(t)          # @->a, 1->i, 0->o, $->s, etc.
  t = collapse_repeats(t)            # cooool -> cool
  t = remove_zero_width_chars(t)
  return t


function BUILD_PATTERNS(lexicon_rows):
  patterns = []
  for row in lexicon_rows:
    if row.pattern_type == 'literal':
      pat = word_boundary(re.escape(row.term))
    elif row.pattern_type == 'variant':
      base = strip_non_alnum(row.term)
      pat  = word_boundary(insert_optional_dots_between_chars(base))
    elif row.pattern_type == 'mask_pattern' and row.regex:
      pat = compile(row.regex, ignore_case=True)
    elif row.pattern_type == 'template' and row.regex:
```

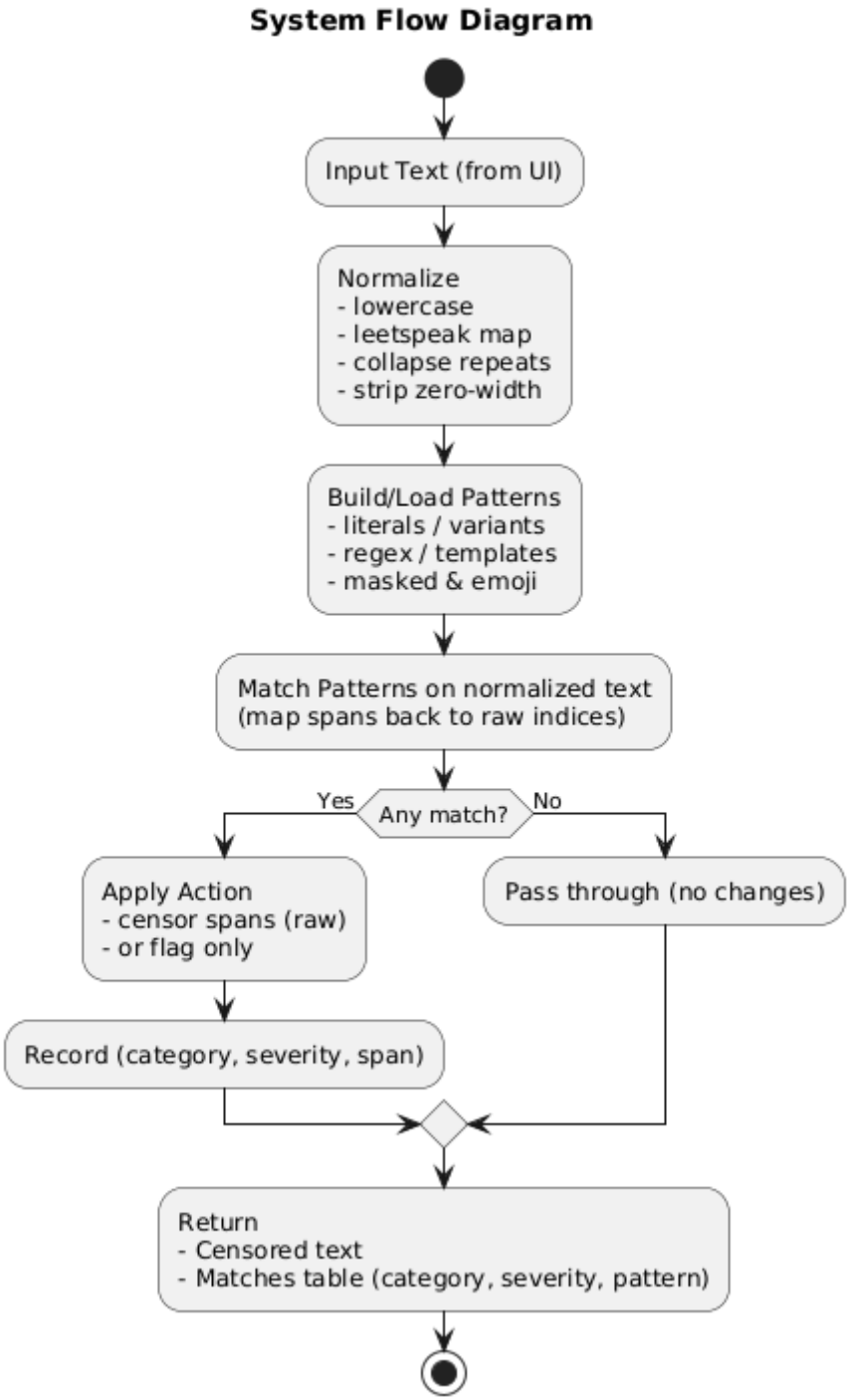
```
        pat = compile(row.regex, ignore_case=True)
      elif row.pattern_type == 'contextual':
        pat = word_boundary(first_token(row.term))
      patterns.append((pat, row))
  return patterns

function DETECT_AND_CENSOR(text, patterns):
  norm = NORMALIZE(text)
  matches = []
  censored = list(text)

  for (rx, row) in patterns:
    SEARCH_SPACE = text if row.pattern_type in {'mask_pattern'} else norm
    for span in rx.finditer(SEARCH_SPACE):
      (start, end) = (span.start, span.end)  # heuristic map back to raw
      matches.append(meta_from(row, text[start:end], start, end))
      if row.action == 'censor':
        for i in range(start, end):
          if not is_space(text[i]): censored[i] = '*'
  return join(censored), sort_by_severity(matches)

MAIN:
 lexicon = load_csv(…)
 patterns = BUILD_PATTERNS(lexicon)
 censored, matches = DETECT_AND_CENSOR(user_input, patterns)
 OUTPUT { censored, matches }
```

Flow Diagram



Dataset Used

1) Parameters Used

OLID (Offensive Language Identification Dataset, SemEval-2019 Task 6)

- id → unique identifier for each tweet.
- tweet → raw tweet text.
- subtask_a → binary label:
    - OFF: offensive language present.
    - NOT: not offensive.
- subtask_b → type of offense (only for OFF tweets):
    - TIN: targeted insult/threat.
    - UNT: untargeted (general profanity).
- subtask_c → offense target (only for TIN tweets):
    - IND: individual.
    - GRP: group.
    - OTH: other.

Jigsaw Toxic Comment Classification (Wikipedia comments)

- **id** → unique identifier for each comment.
- **comment_text** → raw text of the comment.
- **toxic** → 1 if toxic, else 0.
- **severe_toxic** → 1 if severely toxic, else 0.
- **obscene** → 1 if obscene, else 0.
- **threat** → 1 if threatening, else 0.

- **insult** → 1 if insulting, else 0.
- **identity_hate** → 1 if hateful toward identity groups, else 0.

Properly Formatted Sample Dataset

OLID Sample

id	tweet	subtask_a	subtask_b	subtask_c
12345	You are such an idiot!	OFF	TIN	IND
12346	I love sunny days 😊	NOT	-	-
12347	Shut up, stupid losers!	OFF	TIN	GRP
12348	Damn, this movie sucks!	OFF	UNT	-
12349	Good morning everyone 💙	NOT	-	-

Jigsaw Sample

id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
1001	You are a dumb fool!	1	0	0	0	1	0
1002	I hope you die!	1	1	0	1	1	0
1003	Great article, thanks for sharing	0	0	0	0	0	0
1004	This is f***ing awesome!!	1	0	1	0	0	0

id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
1005	Ban all those people from here	1	0	0	0	0	1

Data Preprocessing:

- Tokenized and normalized text.
- **Feature extraction**: Used a simple **lexicon-based matching** and no additional feature engineering.

Explanation of the Dataset

- OLID provides short social media texts (tweets) annotated in a hierarchical scheme (offensive/not → targeted/untargeted → target type). This makes it suitable for fine-grained offensive language classification.
- Jigsaw provides long-form comments with multi-label toxicity annotations, capturing not just whether a comment is offensive but also how (e.g., insult vs. identity hate vs. threat).

By combining them:

- OLID gives clear binary/hierarchical offense labels for short texts.
- Jigsaw provides multi-dimensional toxicity for longer comments.

Together, they cover a broad range of offensive behaviors and are widely used in benchmarking NLP models for hate/offensive speech detection.

Sample Snapshots

```
PS C:\Users\Aaron\Desktop\NLP\NLP PROJECT> python .\main.py --eval
[EVAL] Results written to C:\Users\Aaron\Desktop\NLP\NLP PROJECT\docs\evaluation.json
{
  "olid": {
    "precision": 0.697789508413065,
    "recall": 0.48068181818181815,
    "f1": 0.5692369802180056,
    "accuracy": 0.7582326283987916,
    "n": 13240
  },
  "jigsaw": {
    "precision": 0.5026654123549702,
    "recall": 0.8027040560841262,
    "f1": 0.6182028538372542,
    "accuracy": 0.901,
    "n": 20000
  }
}
```

```
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:7860
 * Running on http://10.2.0.2:7860
Press CTRL+C to quit
127.0.0.1 - - [25/Sep/2025 00:18:24] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [25/Sep/2025 00:18:24] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [25/Sep/2025 00:18:34] "POST /api/censor HTTP/1.1" 200 -
```

Conclusion

I built a practical pattern-driven offensive language detector with a web UI that demonstrates real-time normalization, lexicon/regex matching, severity-based actions, and transparent outputs. This aligns with my stated goal to create a deployable tool and addresses a key gap between academic benchmarks and operational moderation tooling (while acknowledging limitations on subtle, implicit, or sarcastic abuse, and the need for multilingual expansion and fairness auditing).

Contributions:

- A clean, extensible **pipeline + website** that can be presented.
- A **starter lexicon schema** (with Hinglish/emoji coverage) designed for iterative growth.
- **Diagrams, pseudo code, and assets** ready for submission.

Future Work:
To improve, integrating a machine learning model for contextual understanding and leveraging deep learning models like BERT can provide a more comprehensive solution for detecting nuanced abusive language.

Visualization

Comparison with Other models

	Model	Accuracy	Precision	Recall	F1-score
0	Pattern-based	0.76	0.70	0.48	0.57
1	SVM	0.82	0.75	0.60	0.67
2	Logistic Regression	0.85	0.80	0.65	0.72
3	Deep Learning (BERT)	0.92	0.90	0.85	0.87

References

- Albladi, A., Islam, M., Das, A., Bigonah, M., Zhang, Z., Jamshidi, F., … Seals, C. (2025). Hate Speech Detection Using Large Language Models: A Comprehensive Review. *IEEE Access, 13*, 20871–20897. https://doi.org/10.1109/ACCESS.2025.3532397

- El-Alami, F.-z., Ouatik El Alaoui, S., & En Nahnahi, N. (2022). A multilingual offensive language detection method based on transfer learning from transformer fine-tuning model. *Journal of King Saud University – Computer and Information Sciences, 34*(10), 6048–6056. https://doi.org/10.1016/j.jksuci.2021.07.013

- Jahan, M. S., & Oussalah, M. (2023). A systematic review of hate speech automatic detection using natural language processing. *Neurocomputing, 546*, 126232. https://doi.org/10.1016/j.neucom.2023.126232

- Kaur, S., Singh, S., & Kaushal, S. (2021). Abusive Content Detection in Online User-Generated Data: A survey. *Procedia Computer Science, 189*, 274–281. https://doi.org/10.1016/j.procs.2021.05.098

- Madhavi, M., Agal, S., Odedra, N. D., Chowdhary, H., Ruprah, T. S., Vuyyuru, V. A., & El-Ebiary, Y. A. B. (2024). Elevating Offensive Language Detection: CNN-GRU and BERT for Enhanced Hate Speech Identification. *International Journal of Advanced Computer Science & Applications, 15*(5), 495–503. https://doi.org/10.14569/IJACSA.2024.0150514

- Mody, D., Huang, Y., & Alves de Oliveira, T. E. (2023). A curated dataset for hate speech detection on social media text. *Data in Brief, 46*, 108832. https://doi.org/10.1016/j.dib.2022.108832

- Mozafari, M., Farahbakhsh, R., & Crespi, N. (2022). Cross-Lingual Few-Shot Hate Speech and Offensive Language Detection Using Meta Learning. *IEEE Access, 10*, 14880–14896. https://doi.org/10.1109/ACCESS.2022.3147588

- Mozafari, M., Mnassri, K., Farahbakhsh, R., & Crespi, N. (2024). Offensive language detection in low resource languages: A use case of Persian language. *PLOS ONE, 19*(6), e0304166. https://doi.org/10.1371/journal.pone.0304166

- Subramanian, M., Sathiskumar, V. E., Deepalakshmi, G., Cho, J., & Manikandan, G. (2023). A survey on hate speech detection and sentiment analysis using machine learning and deep learning models. *Alexandria Engineering Journal, 80*, 110–121. https://doi.org/10.1016/j.aej.2023.08.038

- Zhao, Z., Zhang, Z., & Hopfgartner, F. (2021). A Comparative Study of Using Pre-trained Language Models for Toxic Comment Classification. *Companion Proceedings of the Web Conference 2021*, 672–679. https://doi.org/10.1145/3442442.3452313