Toxic Content Filtering in Real-Time Speech-to-Text Streams

Prajwal Hosahalli Dayananda
hosap02@pfw.edu
Purdue University Fort Wayne

Abstract

The increasing use of voice-based applications in digital communication has introduced challenges in content moderation, particularly in the detection and filtering of toxic, offensive, and harmful speech in real-time. This paper presents a system that converts live audio into text and employs natural language processing (NLP) to instantly identify and mitigate toxic content. The study explores different NLP models for real-time toxicity detection, discusses dataset selection, and evaluates the effectiveness of the proposed approach based on standard evaluation metrics. The code and models for this research project are available at https://github.com/hdprajwal/Guardscribe

1 Introduction

The rise of live streaming platforms, voice-based social networks, and virtual assistant applications has created a need for real-time moderation of spoken content. Harmful speech, including hate speech, abusive language, and offensive remarks, poses a significant challenge to ensuring safe online interactions. Unlike traditional text-based moderation, speech-to-text processing introduces additional complexities, such as transcription errors and variations in spoken language. The objective of this research project is to develop a real-time system capable of detecting and filtering toxic speech efficiently using NLP techniques.

2 Motivation

Toxic content in live speech streams can have severe consequences, including cyberbullying, misinformation, and online harassment. Social media platforms, gaming communities, and live discussions often struggle to enforce moderation on a scale. Existing solutions rely heavily on manual moderation, which is slow and inconsistent. A robust, real-time, automated system can significantly improve content safety, ensuring a healthier digital environment.

3 Related Work

End-to-end sequence-to-sequence models first demonstrated ASR viability [3, 8], but recent self-supervised approaches such as wav2vec 2.0 [2] and OpenAI's Whisper [9] have substantially improved transcription accuracy, although with differing latency profiles. In parallel, toxicity classification evolved from feature-based and recurrent neural network methods [4, 1] to Transformer-based fine-tuned models like BERT and DistilBERT, which achieve F_1 scores above 0.80 on the Jigsaw dataset [10]. Toxic span detection, popularized by SemEval-2021 Task 5, has progressed from BiLSTM-CRF architectures to Transformer taggers achieving up to 0.65 span F_1 [7]. While real-time moderation pipelines have been explored in conversational agents and live-stream filtering , they typically address text-only inputs or offline processing; this research benchmarks contemporary ASR back-ends and toxicity detectors in an integrated, low-latency speech moderation pipeline.

4 Methodology

4.1 System Architecture

The system implements real-time toxic content filtering through three core components: audio capture, speech-to-text transcription, and toxicity classification. To minimize latency, a multi-threaded design enables concurrent processing: one thread continuously captures microphone input while another handles transcription and toxicity detection, ensuring responsiveness for real-time applications without blocking operations.

4.2 Speech-to-Text Pipeline

I implemented a flexible speech-to-text pipeline that supports multiple ASR engines as mentioned

in Section 4.1 to accommodate various deployment scenarios. As shown in Algorithm 1, the pipeline employs WebRTC's Voice Activity Detection (VAD) with configurable aggressiveness levels to reduce unnecessary processing of non-speech segments, decreasing computational load while maintaining high transcription quality.

Algorithm 1: AudioCapture - microphone ingest & VAD gating

Input: Microphone stream (16 kHz, mono)
Output: Speech frames → `audioFrames` queue

```
open_mic( rate= 16 kHz, block= HOP_MS)
while system_running do
    frame ← mic.read()
    if VAD.detectSpeech(frame, VAD_MODE) then    // WebRTC VAD
        audioFrames.enqueue(frame)
    sleep(10 ms)
```

Algorithm 2: SpeechToText - buffered ASR with latency recording

Input: `audioFrames` queue
Output: (text, asrLatency) → `transcripts` queue

```
buffer ← [ ]
silence ← 0
while system_running do
    frame ← audioFrames.dequeue(block=true)
    buffer.append(frame)
    silence  isSilence(frame)
    if silence > 10 or duration(buffer)≥BUFFER_MS then
        audio ← concat(buffer)
        start ← now()
        txt ← ASR.transcribe(audio, ASR_BACKEND, LANGUAGE_CODE)
        asrLat ← elapsed_ms(start)
        transcripts.enqueue((txt, asrLat))
        buffer.clear(), silence ← 0
```

4.3 Transcription Process

Algorithm 2 shows that a buffer-based approach is used to accumulate audio frames until a speech segment is completed (identified by a configurable silence threshold). With Whisper, a 2-second buffer with 500 ms step size provides optimal accuracy-responsiveness balance, processing overlapping audio windows to maintain contextual continuity.

Algorithm 3: ModerateText - sentence classification and optional span masking

Input: `transcripts` queue
Output: (cleanText, totalLatency) → `moderatedText` queue

```
while system_running do
    (txt, asrLat) ← transcripts.dequeue(block=true)
    start ← now()
    (prob, lbl) ← SentenceClf.predict(txt)
    sentLat ← elapsed_ms(start)
    if lbl = toxic and prob ≥ TOXICITY_THRESHOLD then
        if ENABLE_TOKEN_MASK then
            spans ← TokenClf.predict_spans(txt)
            clean ← Mask(txt, spans, SPAN_MASKING_MODE)
        else
            clean ← MaskAll(txt, SPAN_MASKING_MODE)
    else
        clean ← txt
    totalLat ← asrLat + sentLat + extraSpanLat
    moderatedText.enqueue((clean, totalLat))
```

4.4 Content Moderation

Algorithm 3 shows the moderation pipeline that processes classifier scores through a multi step decision process:

1. If sentence level probability exceeds the threshold (default 0.70), the utterance is flagged as potentially toxic.
2. Token classifier returns BIO tags localizing abusive spans.
3. Context evaluator examines surrounding dialogue to reduce false positives.
4. System applies appropriate masking strategy while preserving nontoxic words.

4.5 Performance Considerations

In real-time filtering, measured average latencies were: Whisper (tiny): 450-650ms, Whisper (base): >650ms, Google Speech-to-Text: 700ms (network-dependent), Wav2Vec: < 75ms. Testing conducted

on quad-core CPU with 32GB RAM with a GTX 1050 GPU.

5 Experiments

5.1 Datasets

To train and evaluate the system, I rely on two publicly available toxic language corpora:

- **Jigsaw Toxic Comment Dataset** (Kaggle)[5] - a large-scale collection of Wikipedia discussion fragments annotated for multiple categories of toxicity (toxic, severe toxic, obscene, threat, insult, identity hate).
- **Toxic Spans Detection Dataset** (GitHub)[11] - sentences drawn from social media posts with character-level span annotations that mark the exact location of toxic language.
- **The LJ Speech Dataset** (Kaggle)[6] - a collection of 13,100 short audio clips of a single speaker reading passages from nonfiction books, providing clean speech samples to evaluate the accuracy of the ASR components in transcribing content.

These resources provide supervision at both the sentence and the span levels, eliminating the need for additional external datasets.

5.2 Model Configurations

- ASR back-ends - *wav2vec 2.0 (base)* fine-tuned on LibriSpeech, and *Whisper (base)*.
- Sentence-level detector - DistilBERT initialised from `distilbert-base-uncased` and fine-tuned for 3 epochs on Jigsaw toxic comments.
- Token-level detector - DistilBERT with a 3-way BIO head (`B-toxic`, `I-toxic`, `O`); initialized from `distilbert-base-uncased` and fine-tuned for 3 epochs on Toxic-Spans.

All models were fine-tuned on a single T4 GPU.

5.3 Evaluation

- **ASR.** Each clip was transcribed once. The metrics reported are word-error-rate(WER), character-error-rate(CER), end-to-end latency (ms), and real-time-factor(RTF).
- **Token detector.** Predicted BIO tags were converted to character spans; precision, recall, and f_1 scores were analyzed.

6 Results

This section presents the experimental results of research, covering both the automatic speech-recognition (ASR) front-end and the toxic–content detection pipeline.

6.1 Speech-to-Text Performance

I evaluated two ASR back-ends for the final result (Whisper-base and wav2vec 2.0 base) on 100 randomly selected LJSpeech utterances.

wav2vec offers real-time performance (RTF 0.02) at the cost of a 3% higher WER.

6.2 Toxicity Detection

Sentence classifcation and token level span detection were evaluated on the toxic span test dataset (n=2000).

Threshold Selection. At threshold 0.7 (chosen via grid search), I found that it balances false positives and negatives (see Table 5).

6.3 End-to-End Latency and Limitations

Combined ASR + classification + span detection incurs 1.1–1.8 s per utterance for Whisper asr model while resulting in < 400ms when using Wav2Vec. The main constraints remain accurate ASR to achieve reliable real time usability.

7 Analysis

7.1 ASR Error Characteristics

Whisper-base exhibits a heavily skewed WER distribution: over 60% of utterances are transcribed perfectly (WER = 0), while the worst 7% exceed WER 0.20, typically on rapid speech or rare words. In contrast, wav2vec 2.0 shows a flatter error profile with only $\approx 58\%$ perfect transcriptions but fewer extreme errors. This suggests wav2vec may be more robust to phonetic variability at the expense of peak accuracy.

7.2 Latency–Accuracy Trade-off

wav2vec's 139 ms latency (RTF 0.02) enables true real-time operation but incurs a 3.3 % increase in WER relative to Whisper. Whisper's 1.24 s delay (RTF 0.31) yields highest fidelity. A hybrid deployment using wav2vec for short utterances and Whisper for longer or low confidence segments could attain sub 0.15 RTF with near-Whisper WER.

Component	Quality Metric(s)	Latency	Notes
wav2vec 2.0	WER $= 9.97\%$, CER $= 4.80\%$	139 ms / RTF 0.021	Fastest ASR back-end
Whisper-base	WER $= 6.69\%$, CER $= 1.86\%$	1235 ms / RTF 0.309	Highest ASR accuracy
Token detector	Span-$F_1 = 0.623$	4.9 ms	Sub-frame latency

Table 1: Aggregate performance and latency (mean values).

Table 2: ASR accuracy metrics on LJSpeech (100 utterances).

Metric	Whisper-base	wav2vec 2.0 base
Mean WER	0.067	0.100
Median WER	0.000	0.005
Mean CER	0.019	0.048
Median CER	0.000	0.007

Table 3: ASR latency (CPU; ms) and RTF

Model	Mean latency	RTF
Whisper-base	1235	0.31
wav2vec 2.0	139	0.02

7.3 Toxicity Detection Errors

Sentence-level classification prioritizes precision (0.835) over recall (0.781), resulting in conservative flagging of toxicity. Manual inspection reveals most false positives arise from benign idioms ("break a leg") or quoted slurs, while false negatives often involve implied or multi-sentence abuse. Token-level span detection (F_1 0.623) correctly identifies explicit terms but misses multi-token expressions and contextual insults. Adjusting the decision threshold shifts this balance: lower thresholds improve recall at minor precision cost.

7.4 End-to-End Pipeline Implications

Total pipeline latency is 1.1-1.8 s with Whisper or < 0.4 s with wav2vec dictates use cases. For buffered live moderation (e.g., chat delay), Whisper's delay is acceptable; for interactive voice assistants, wav2vec is preferable. Pipeline parallelism and chunked streaming can reduce perceived delay to $\approx 1s$ for Whisper, narrowing the gap.

8 Conclusion

8.1 Conclusion and Future Work

This work demonstrates that modern ASR models coupled with lightweight toxicity detectors can support near real-time speech moderation. Wav2Vec offers sub-400 ms latency with acceptable accuracy, while Whisper delivers higher fidelity at $\approx 1.2s$ delay. DistilBERT-based classifiers achieve f_1 scores of 0.807 (sentence) and 0.623 (token) with end-to-end latency under 1.8 s.

Table 4: Toxicity detection performance on test sets.

Task	Metric	Value
Sentence Classification	Accuracy	0.701
	F_1 (toxic)	0.807
	Precision	0.835
	Recall	0.781
	PR-AUC	0.845
Token-Level Span Detection	Span F_1	0.623
	Precision	0.621
	Recall	0.675

Table 5: Sentence-level trade-off at select thresholds

Thresh.	Acc.	F_1	P / R
0.5	0.713	0.818	0.832 / 0.804
0.7	0.701	0.807	0.835 / 0.781

Future work will explore adaptive back-end selection, richer contextual models for nuanced abuse detection, and on-device quantization to meet sub-500 ms real-time requirements.

References

[1] Pinkesh Badjatiya, Shashwat Gupta, Manish Gupta, and Vasudev Varma. 2017. Deep learning for hate speech detection in tweets. In *Companion Proceedings of the The Web Conference 2017*, pages 759–760.

[2] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460.

[3] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. 2016. Listen, attend and spell. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964.

[4] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In

Proceedings of the 11th International AAAI Conference on Web and Social Media (ICWSM), pages 512–515.

[5] Jigsaw. 2018. Jigsaw toxic comment classification challenge. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge. Accessed: 18 Apr 2025.

[6] LJSpeech. 2021. The lj speech dataset. https://www.kaggle.com/datasets/mathurinache/the-lj-speech-dataset?select=LJSpeech-1.1. Accessed: 02 May 2025.

[7] John D. Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval)*, pages 1060–1066.

[8] Vineel Pratap, Qiantong Xu, Jacob Kahn, Gilad Avidov, Tatiana Likhomanenko, Awni Hannun, Vitaliy Liptchinsky, Gabriel Synnaeve, and Ronan Collobert. 2020. Scaling up online speech recognition using convnets. In *Proceedings of Interspeech*, pages 371–375.

[9] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. arXiv preprint arXiv:2212.04356.

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

[11] ToxicSpans. 2020. Toxic spans detection dataset. https://github.com/ipavlopoulos/toxic_spans. Accessed: 18 Apr 2025.