

Reddit Hate Speech Detection System - Architecture

Overview

A microservices-based system for automated detection, analysis, and monitoring of hate speech content on Reddit. The system collects posts, enriches user data, scores content for risk, and provides a REST API for data access and alerting.

Project Structure

```
active_fence_assignment/
├── services/
│   ├── api/                                # FastAPI REST service
│   │   ├── src/
│   │   │   ├── api.py                    # FastAPI application (30+ endpoints)
│   │   │   ├── config.py                 # API configuration
│   │   │   └── database/                 # Database layer
│   │   │       ├── models.py             # SQLAlchemy ORM models
│   │   │       ├── schemas.py            # Pydantic schemas
│   │   │       ├── crud.py               # Database operations
│   │   │       └── database.py            # Connection management
│   │   ├── tests/                        # Unit tests (pytest)
│   │   └── Dockerfile
│   └── scraper/                          # Data collection & processing
│       ├── src/
│       │   ├── main.py                   # Entry point
│       │   ├── pipeline.py               # Complete ETL pipeline
│       │   ├── config.py                 # Scraper configuration
│       │   ├── api_client.py             # API communication
│       │   ├── monitoring.py              # Daily monitoring system
│       │   ├── collectors/               # Data collection
│       │   │   └── reddit_scraper.py
│       │   ├── enrichers/               # Data enrichment
│       │   │   └── user_enricher.py
│       │   └── scorers/                  # Risk assessment
│       │       └── hate_speech_scorer.py
│       ├── tests/                        # Comprehensive tests
│       └── Dockerfile
├── shared/                               # Shared models
│   ├── models.py                         # Pydantic models (ScoredPost, ScoredUser)
│   └── pyproject.toml
├── data/
│   ├── hurtlex_processed.json             # Hate speech lexicon (114+ keywords)
│   ├── reddit_detection.db               # SQLite database
│   └── exports/                          # Output data (CSV/JSON)
```

— docker-compose.yml	# Multi-service orchestration
— README.md	# Project documentation

Services Architecture

Service 1: Scraper Service (services/scraper)

Purpose: Automated data collection, enrichment, and scoring

Components

Component	File	Description
RedditScraper	collectors/reddit_scraper.py	Uses Reddit's public JSON API. Collects posts from subreddits, searches for controversial content, fetches user history (60+ days lookback). Rate-limited (2s between requests).
UserEnricher	enrichers/user_enricher.py	Calculates activity metrics, tracks subreddit diversity, aggregates content from 2-month lookback period. Detects edge cases (new users, deleted accounts, private profiles).
HateSpeechScorer	scorers/hate_speech_scorer.py	Rule-based risk assessment using HurtLex lexicon (114+ categorized keywords). Context-aware scoring with hate, violence, and slur pattern detection.
DataPipeline	pipeline.py	Orchestrates full ETL: collect → score → enrich → submit → monitor
UserMonitor	monitoring.py	Daily monitoring of flagged users, generates alerts for high-risk activity
APIClient	api_client.py	HTTP client for API service communication

Pipeline Flow

1. Collect posts from subreddits + search terms
2. Score all posts for hate/violence content
3. Extract unique authors (filter bots/deleted)
4. Prioritize high-risk authors
5. Fetch user enrichment data
6. Score users based on content analysis
7. Send data to API service
8. Flag high-risk users for monitoring
9. Create alerts for critical content
10. Run daily monitoring of flagged users

Service 2: API Service (services/api)

Purpose: REST API for data storage, querying, and monitoring

Technology Stack: FastAPI, SQLAlchemy, SQLite, Pydantic

API Endpoints

Category	Endpoints
Health & Stats	GET /health, GET /statistics
Posts	POST /posts, GET /posts, GET /posts/high-risk, GET /posts/{id}, PATCH /posts/{id}, DELETE /posts/{id}
Users	POST /users, GET /users, GET /users/high-risk, GET /users/monitored, GET /users/{username}, PATCH /users/{username}, DELETE /users/{username}
Bulk Operations	POST /bulk/posts, POST /bulk/users
Alerts	POST /alerts, GET /alerts, GET /alerts/{id}, PATCH /alerts/{id}, DELETE /alerts/{id}
Monitoring Logs	Query by username, activity type, date range

Interactive Docs: <http://localhost:8000/docs> (Swagger UI)

Database Schema

Database: SQLite (`data/reddit_detection.db`)

Table: Posts

Column	Type	Description
id	String (PK)	Reddit post ID
title	Text	Post title
selftext	Text	Post body content
author	String (FK)	Username of author
subreddit	String	Subreddit name
created_utc	DateTime	Post creation time
score	Integer	Reddit score (upvotes - downvotes)
upvote_ratio	Float	Upvote ratio
num_comments	Integer	Number of comments
is_self	Boolean	Is self post
over_18	Boolean	NSFW flag

risk_score	Integer	Calculated risk score (0-100)
risk_level	String	minimal/low/medium/high/critical
hate_score	Float	Hate content score
violence_score	Float	Violence content score
risk_explanation	Text	Human-readable explanation
risk_flags	JSON	Detected flags/keywords
scored_at	DateTime	When scoring occurred
collected_at	DateTime	When collected

Indexes: `idx_risk_level_score` , `idx_subreddit_date` , `idx_author_date`

Table: Users

Column	Type	Description
username	String (PK)	Reddit username
account_created_utc	DateTime	Account creation date
link_karma	Integer	Link karma
comment_karma	Integer	Comment karma
is_gold	Boolean	Reddit Gold status
is_mod	Boolean	Is moderator
risk_score	Integer	Calculated risk score (0-100)
risk_level	String	minimal/low/medium/high/critical
hate_score	Float	Aggregate hate score
violence_score	Float	Aggregate violence score
risk_explanation	Text	Human-readable explanation
risk_factors	JSON	Detailed risk factors
total_posts_analyzed	Integer	Posts analyzed count
flagged_posts_count	Integer	Flagged posts count
avg_post_risk_score	Float	Average risk across posts
max_risk_score_seen	Integer	Highest single risk score
is_monitored	Boolean	Monitoring flag
last_monitored_at	DateTime	Last monitoring scan

alert_count	Integer	Total alerts generated
-------------	---------	------------------------

Indexes: `idx_risk_level_score_user` , `idx_monitored_updated`

Relationships: One-to-many with Posts, One-to-many with Alerts

Table: Alerts

Column	Type	Description
id	Integer (PK)	Auto-increment ID
username	String (FK)	Associated user
post_id	String (FK)	Associated post (optional)
alert_type	String	high_risk_post, critical_risk_user
severity	String	critical, high, medium, low
risk_score	Integer	Score that triggered alert
description	Text	Alert description
details	JSON	Additional context
status	String	new, reviewed, dismissed, escalated
reviewed_at	DateTime	Review timestamp
reviewed_by	String	Reviewer username
resolution_notes	Text	Resolution notes
created_at	DateTime	Alert creation time

Indexes: `idx_severity_status` , `idx_created_severity`

Table: MonitoringLogs

Column	Type	Description
id	Integer (PK)	Auto-increment ID
username	String	Monitored user
activity_type	String	scan, alert, update
description	Text	Activity description
findings	JSON	Monitoring findings
created_at	DateTime	Log timestamp

Indexes: `idx_username_date`

Risk Scoring System

Scoring Mechanism

Category	Points per Keyword
Extreme hate	+30
High hate	+20
Medium hate	+10
Extreme violence	+30
High violence	+20
Medium violence	+10
Slur patterns (regex)	+40

Context Reduction Factors

Context	Score Reduction
Discussion context	-20%
Quotation detected	-20%
Negation present	-10%

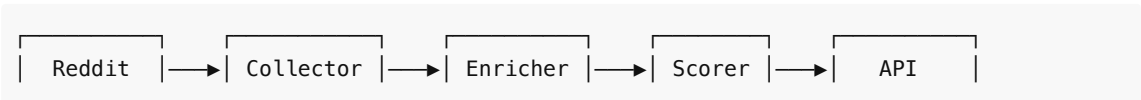
Risk Levels

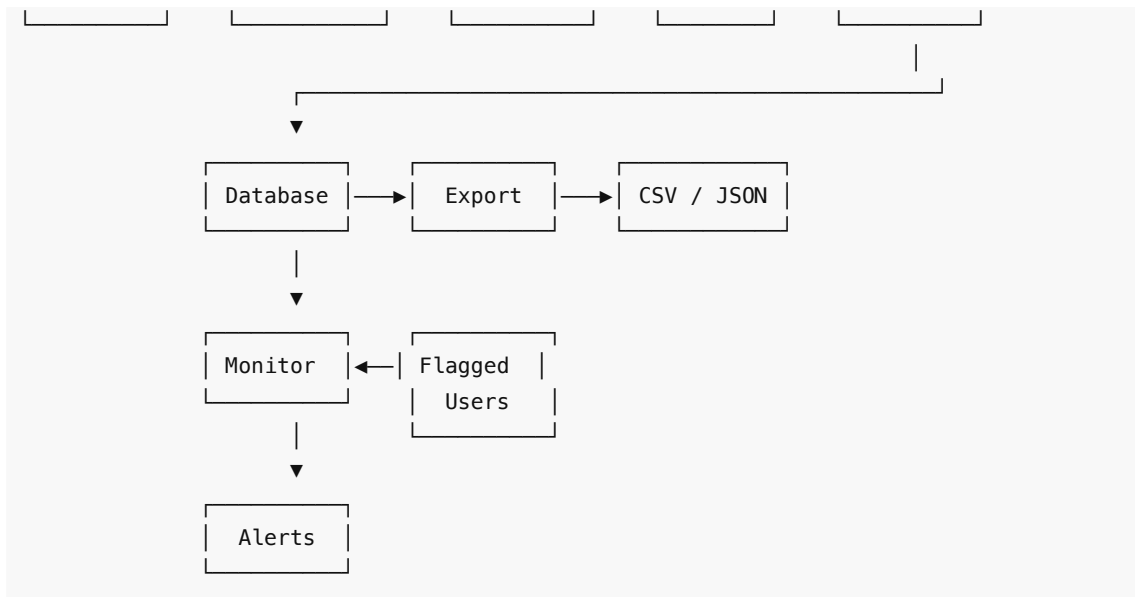
Level	Score Range
Minimal	0-9
Low	10-29
Medium	30-49
High	50-69
Critical	70-100

HurtLex Lexicon

- **Source:** GitHub [valeribasile/hurtlex v1.2](#)
- **License:** CC-BY-SA 4.0
- **Contents:** 68 hate keywords, 46 violence keywords, 12 regex patterns

Data Flow





Configuration

Scraper Configuration (`.env`)

```
LOG_LEVEL=INFO
RATE_LIMIT_DELAY=2.0
TARGET_SUBREDDITS=news,politics,unpopularopinion,...
POSTS_PER_SUBREDDIT=10
MAX_USERS_TO_ENRICH=20
SEARCH_TERMS=hate,violence,threat,kill,...
USER_HISTORY_DAYS=60
CRITICAL_RISK_THRESHOLD=70
HIGH_RISK_THRESHOLD=50
API_URL=http://api:8000
API_TIMEOUT=30
```

API Configuration (`.env`)

```
LOG_LEVEL=INFO
DATABASE_URL=sqlite:///app/data/reddit_detection.db
HOST=0.0.0.0
PORT=8000
```

Deployment

Docker Compose

```
services:
  api:
```

```
build: ./services/api
ports:
  - "8000:8000"
volumes:
  - ./data:/app/data
  - ./logs:/app/logs
networks:
  - reddit-network

scraper:
build: ./services/scraper
volumes:
  - ./data:/app/data
  - ./logs:/app/logs
depends_on:
  - api
networks:
  - reddit-network

networks:
  reddit-network:
    driver: bridge
```

Running the System

```
# Start all services
docker-compose up -d

# View logs
docker-compose logs -f

# Stop services
docker-compose down
```

Technology Stack

Category	Technologies
Backend	Python 3.11, FastAPI, SQLAlchemy, Pydantic
Database	SQLite
HTTP Client	Requests, BeautifulSoup4
Testing	pytest, pytest-cov
Linting	Black, Ruff, MyPy
Deployment	Docker, Docker Compose, UV package manager

Edge Case Handling

Case	Handling
New users	Detected via <code>total_posts = 0</code>
Deleted accounts	Skipped if no content returned
Private profiles	Handled with <code>no_content</code> status
Bot filtering	Filters AutoModerator, [deleted], [removed]
Deduplication	Uses post IDs to prevent duplicates
Rate limiting	2-second delays between requests

Exports

Output files are saved to `data/exports/` :

- `posts.json` / `posts.csv` - All processed posts
- `users.json` / `users.csv` - All processed users
- `alerts.json` / `alerts.csv` - Generated alerts
- `monitoring_logs.json` / `monitoring_logs.csv` - Monitoring history