CHAPTER 1: INTRODUCTION

1.1 ABSTRACT

The YouTube Video Comment Analyzer is an intelligent web-based tool designed to assist content creators in gaining deeper insights into their audience's feedback. Built using advanced natural language processing techniques, this tool performs Sentiment Analysis and Toxicity Detection on video comments to identify positive, negative, and potentially harmful feedback. Beyond just classification, it extracts valuable suggestions, highlights public demands, and provides a concise summary of the overall comment section. These features enable YouTubers to understand viewer sentiment, improve content quality, and make data-driven decisions for future videos. With its intuitive interface and real-time analysis, the tool bridges the gap between creators and their audience, transforming raw comments into actionable insights.

1.2 INTRODUCTION

In the age of digital content creation, understanding audience feedback is crucial for the growth and engagement of any YouTube channel. To address this need, I developed a YouTube Video Comment Analyzer Tool—a powerful web-based application designed to provide insightful analysis of comments posted on YouTube videos.

This tool leverages Natural Language Processing (NLP) and Machine Learning techniques to help content creators better understand their audience by analyzing the sentiment and toxicity levels of comments. It goes beyond simple analysis by offering actionable Suggestions, identifying common Public Demands, and generating a Concise Summary of viewer feedback. These features allow YouTubers to adapt and improve their content strategy based on real audience insights.

Key Features:

1. Sentiment Analysis: Classifies comments as Positive, Negative, or Neutral to gauge viewer emotions.
2. Toxicity Detection: Flags toxic, hateful, or abusive comments using deep learning models.
3. Viewer Suggestions Extraction: Extracts meaningful suggestions provided by the audience.
4. Public Demand Recognition: Highlights frequently requested topics, features, or improvements.
5. Automated Comment Summary: Generates a brief, informative summary of overall viewer feedback for quick understanding.

This tool serves as an AI-powered assistant for YouTube creators, enabling data-driven decisions to enhance content quality, viewer satisfaction, and community management.

CHAPTER 2 : LITERATURE SURVEY

Recent studies in NLP have explored various techniques for social media text analysis. Sentiment analysis tools like VADER[1] and TextBlob[2] demonstrate strong performance on short, informal text but struggle with contextual nuances. Hybrid approaches combining lexicon-based and machine learning methods show improved accuracy [3], motivating our multi-model system.

For toxicity detection, BERT-based models[4] outperform traditional classifiers by capturing semantic context. Similar systems like Google's Perspective API[5] validate the practical need for automated moderation. Our implementation adapts these findings while optimizing for YouTube's unique comment patterns.

In text summarization, LLM-based abstractive methods[6] have surpassed extractive techniques (e.g., TextRank [7]) in coherence. However, computational costs remain a challenge [8], leading us to use API-based solutions for scalability. Existing YouTube analytics tools (e.g., YouTube Studio) lack advanced NLP integration [9], creating a gap our tool addresses.

Topic modeling research [10] shows TF-IDF + clustering effectively identifies emerging themes in user-generated content. We apply this approach for demand analysis, balancing accuracy and speed—a key requirement for creators [11].

Key research gaps our work addresses:

Limited integrated solutions combining sentiment, toxicity, and summarization [12]
Actionable insights for content optimization [13]
Real-time processing challenges in comment analysis [14]

CHAPTER 3: SCOPE OF THE PROJECT AND METHODOLOGY

3.1 SYSTEM ARCHITECTURE

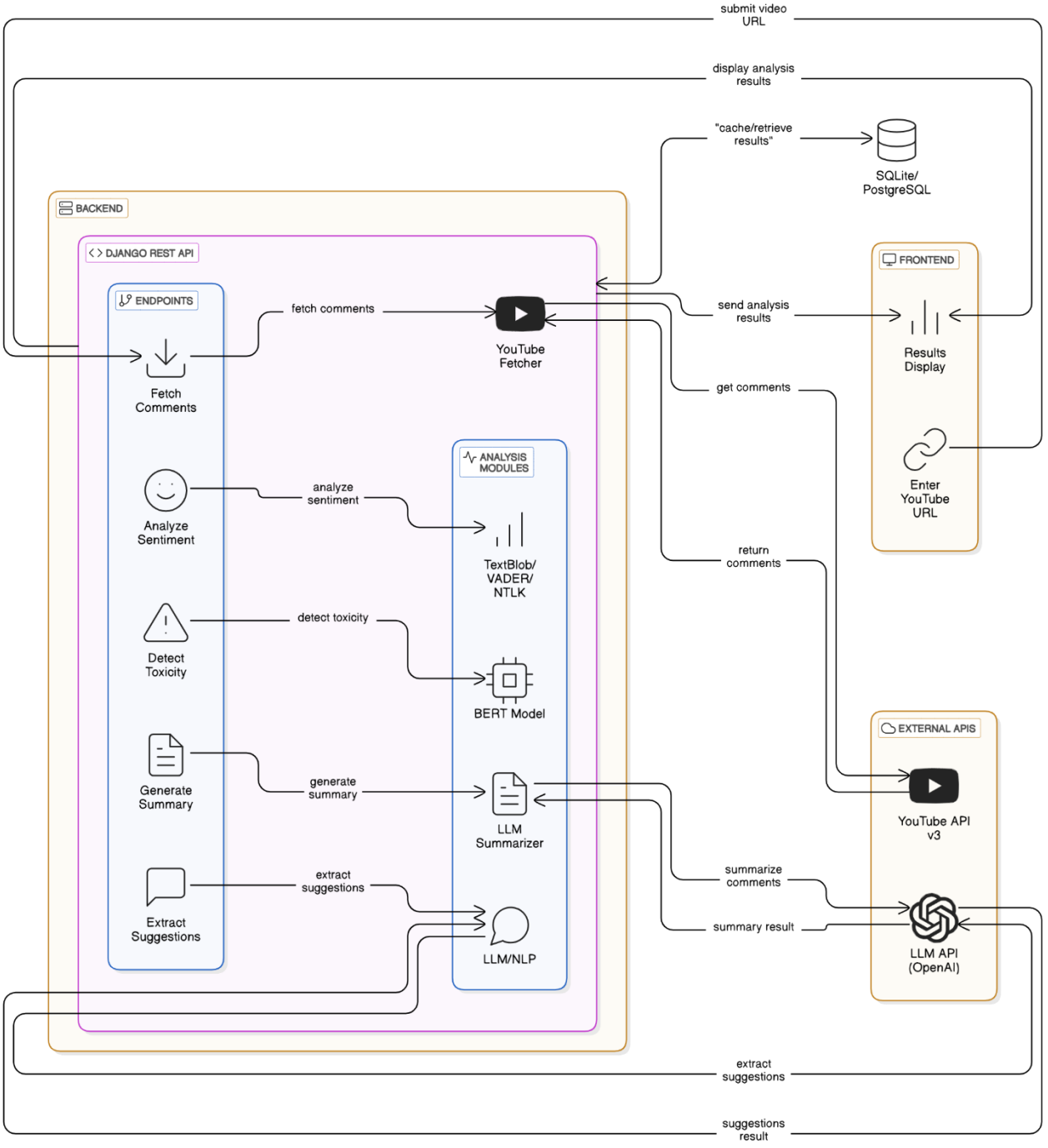

Fig 3.1 System Architecture

3.2 METHODOLOGY

The YouTube Comment Analyzer follows a structured pipeline to collect, process, and analyze YouTube comments for sentiment, toxicity, summaries, and viewer suggestions.

3.1 Comment Collection

Users input a YouTube video URL. The system extracts the video ID and uses the YouTube Data API to fetch comments in JSON format.

3.2 Preprocessing

Comments are cleaned by removing emojis, special characters, links, and stopwords. Tokenization and normalization are applied to prepare data for analysis.

3.3 Sentiment Analysis

Using TextBlob, VADER, and NLTK, each comment is analyzed to determine whether it is positive, negative, or neutral.

3.4 Toxicity Detection

A pre-trained BERT model classifies comments as toxic or non-toxic, identifying hate speech or offensive language.

3.5 Summarization

An LLM API generates a concise summary of the overall comment section, capturing the main points and tone.

3.6 Feedback Extraction

The system uses LLM-based methods to extract suggestions and public demand from the comment text, helping creators understand viewer needs

3.7 Visualization

Results are displayed on a user-friendly dashboard with sentiment charts, toxic comment highlights, summaries, and suggestions.

CHAPTER 4 : PROPOSED METHOD

4.1 DFD LEVEL0 DIAGRAM

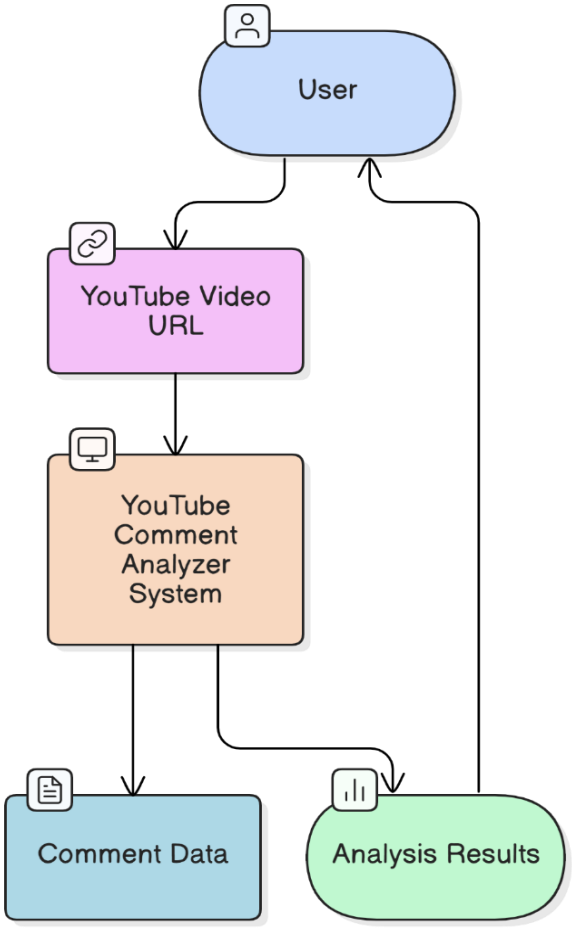

Fig. 4.1.1 DFD Level Diagram

4.2 DFD LEVEL 1 DIAGRAM

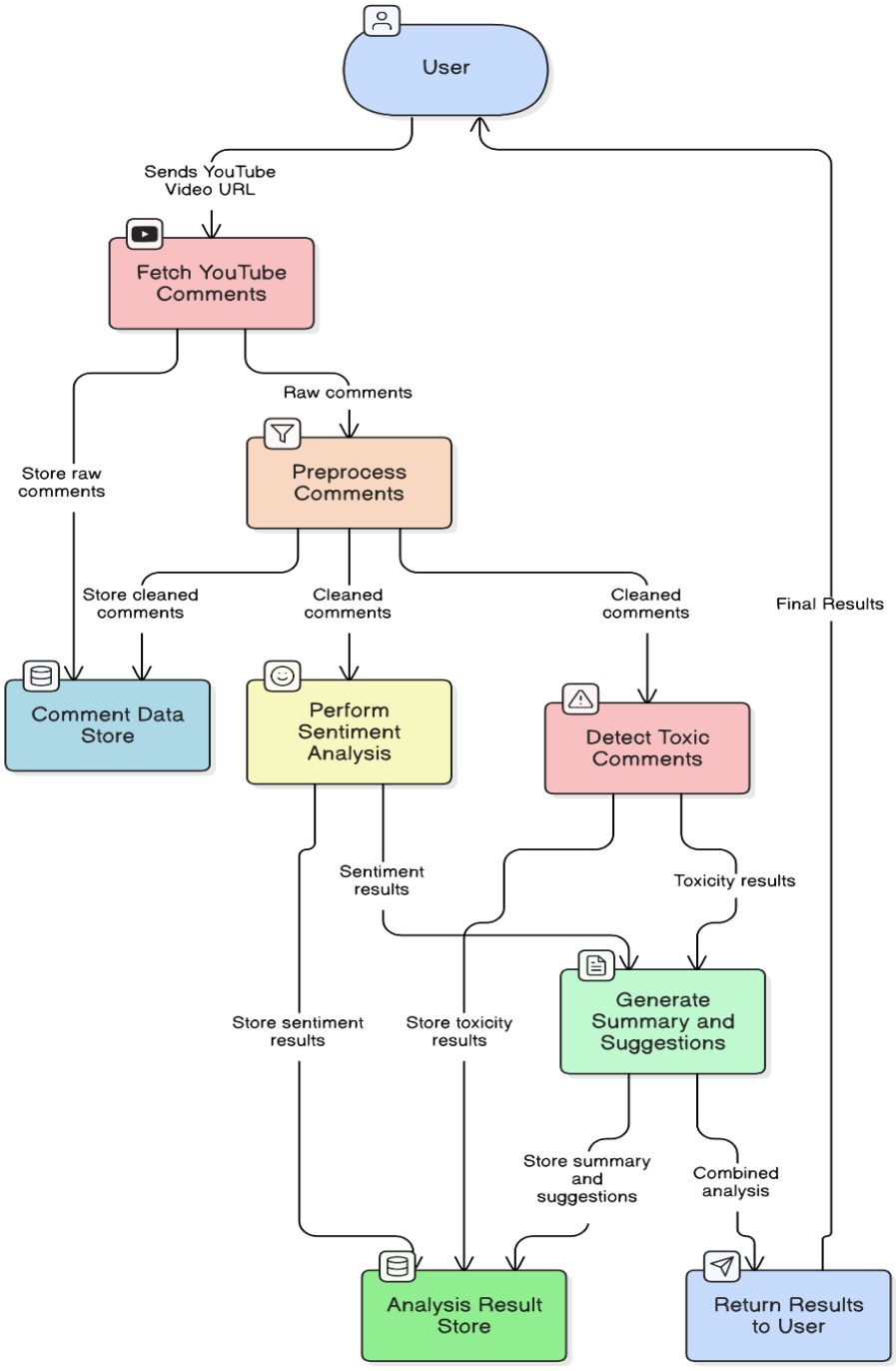

Fig 4.2: Data Flow Diagram Level 1

4.3 Flow Chart

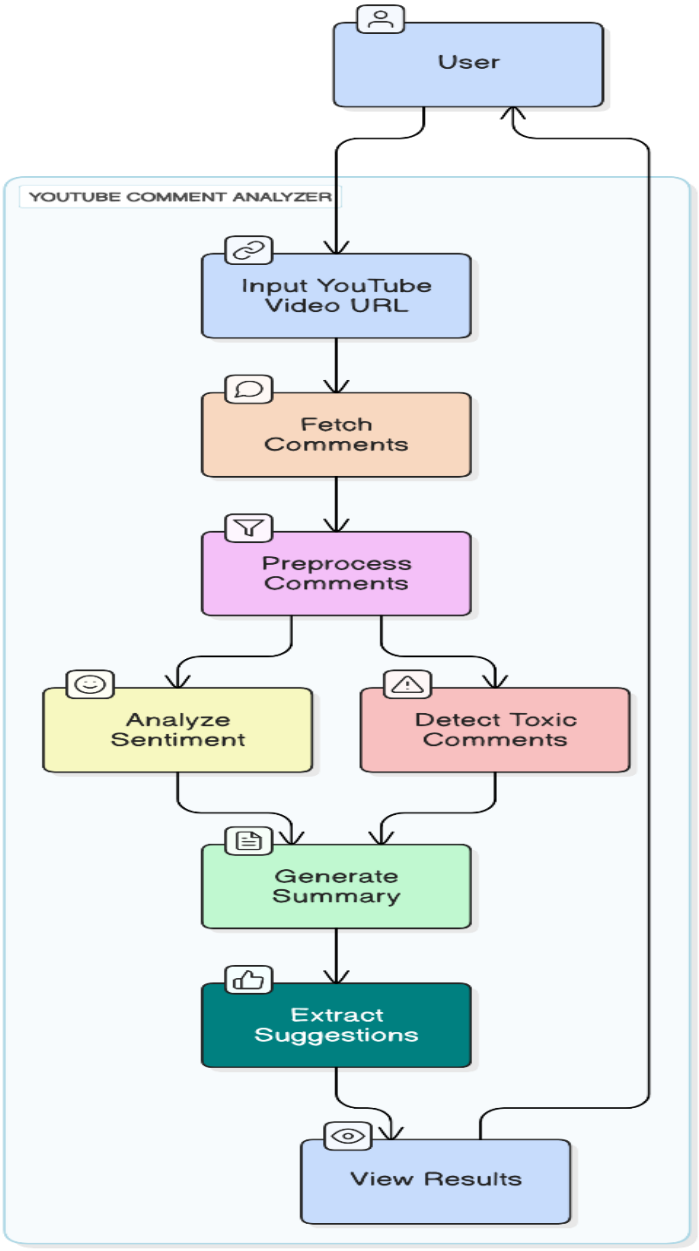

Fig 4.3: Flow Chart

CHAPTER 5 : SOFTWARE REQUIREMENT SPECIFICATION

SOFTWARE REQUIREMENT SPECIFICATION (SRS):

The Software Requirements Specification (SRS) document lays the foundation for all software engineering activities. It is constructed after a thorough elicitation and analysis of the project's requirements. This formal document serves as a blueprint for the software, enabling stakeholders to review whether the specified functionalities align with their expectations and needs.

The SRS includes both user-level requirements and detailed system specifications. It acts as a comprehensive guide that defines the scope, functionality, and constraints of the system. Depending on its origin, the SRS can serve different purposes. If written by the client, it reflects the user's needs and expectations. If authored by the developer, it outlines technical specifications and serves as a contractual agreement between the client and the development team. For the YouTube Video Comment Analyzer Tool, the SRS includes specifications for sentiment analysis, toxicity detection, suggestion extraction, public demand analysis, and automated comment summarization. The system is designed to operate within a web environment and leverages modern web technologies and machine learning models to deliver accurate and actionable insights to YouTube content creators.

5.1 SOFTWARE REQUIREMENTS

- Operating System: Windows 10/11 (64-bit) or any OS that supports Docker or Python environments (Linux/macOS)
- Editor/IDE: Visual Studio Code / PyCharm (Recommended)
- Languages & Frameworks:
- Frontend: HTML, CSS, JavaScript (React.js)
- Backend: Python (Django REST Framework)
- Machine Learning: scikit-learn, TensorFlow, or Hugging Face Transformers (for BERT-based models)
- Other Tools:
    - Git & GitHub for version control
    - Postman for API testing

5.2 HARDWARE REQUIREMENTS

- Processor: Intel i3 or higher (64-bit architecture)
- RAM: Minimum 4 GB (8 GB or more recommended for ML model inference)
- Storage: Minimum 2 GB of free disk space
- Mobile Support: Android (for testing if mobile responsiveness or PWA features are implemented)

CHAPTER 6 : IMPLEMENTATION

SOURCE CODE:

```
import asyncio
import time
import sys
import os
sys.path.append(os.path.dirname(os.path.dirname(os.path.dirname(os.path.abspath(__file__)))))

from yt_module.api_client import YouTubeClient, CommentFetcher
from yt_module.parser import YouTubeURLParser
from processing.cleaners import clean_text
from analysis.sentiment import SentimentAnalyzer
from analysis.emojis import EmojiAnalyzer
from analysis.ToxicityAnalyzer import BertToxicityAnalyzer
from analysis.geminiAnalyzer import YouTubeCommentAnalyzerWithAI as GeminiAnalyzer
import nltk

nltk.download('vader_lexicon')

class YouTubeCommentAnalyzer:
    def __init__(self):
        self.client = YouTubeClient()
        self.fetcher = CommentFetcher(self.client)
        self.sentiment = SentimentAnalyzer()
        self.emoji = EmojiAnalyzer()
        self.toxicity_bert = BertToxicityAnalyzer()

    async def analyze(self, video_url, comment_limit=100):
        video_id = YouTubeURLParser().extract_video_id(video_url)
        if not video_id:
            raise ValueError("Invalid YouTube URL")

        # Fetch comments
        start_time = time.time()
        raw_comments = self.fetcher.fetch_comments(video_id, comment_limit)
```

```
    fetch_time = time.time() - start_time

    start_time = time.time()
    cleaned_comments = [clean_text(c) for c in raw_comments]
    clean_time = time.time() - start_time

    # Perform sentiment analysis
    start_time = time.time()
    sentiment_vader = self.sentiment.analyze_vader(cleaned_comments)
    sentiment_vader_time = time.time() - start_time

    start_time = time.time()
    sentiment_textblob = self.sentiment.analyze_textblob(cleaned_comments)
    sentiment_textblob_time = time.time() - start_time

    # Perform toxicity analysis
    start_time = time.time()
    toxic_comments = self.toxicity_bert.analyze(cleaned_comments)  # Get toxic comments
    toxicity_bert_time = time.time() - start_time

    top_emojis = self.emoji.top_emojis(raw_comments)

    # Perform Gemini analysis
    start_time = time.time()
    gemini_analyzer = GeminiAnalyzer(raw_comments)
    gemini_summary = gemini_analyzer.get_summary()
    gemini_demands = gemini_analyzer.get_public_demand()
    gemini_suggestions = gemini_analyzer.get_suggestions()
    gemini_time = time.time() - start_time

    return {
        'sentiment': {
            'vader': sentiment_vader,
            'textblob': sentiment_textblob
        },
        'toxicity': {
            'bert': toxic_comments,
        },
        'emojis': top_emojis,
```

```
        'gemini': {
            'summary': gemini_summary,
            'demands': gemini_demands,
            'suggestions': gemini_suggestions
        },
        'total_comments': len(cleaned_comments),
        'time_complexity': {
            'fetch_comments': fetch_time,
            'clean_comments': clean_time,
            'sentiment_vader': sentiment_vader_time,
            'sentiment_textblob': sentiment_textblob_time,
            'toxicity_bert': toxicity_bert_time,
            'gemini_analysis': gemini_time
        }
    }

def print_results(results):
    print("\n📊 ---------- YouTube Comment Analysis ---------- 📊\n")

    # Display total comments analyzed
    print(f"📌 Total Comments Analyzed: {results['total_comments']}\n")

    # Display Sentiment Analysis
    print("🔹 Sentiment Analysis:")
    for method, data in results['sentiment'].items():
        print(f"  ✅ {method.title()} Sentiment:")
        print(f"  - Positive: {data['breakdown'].get('positive', 0)}%")
        print(f"  - Neutral: {data['breakdown'].get('neutral', 0)}%")
        print(f"  - Negative: {data['breakdown'].get('negative', 0)}%")
        print(f"  - Overall Score: {data['overall']:.2f}\n")

    # Display Toxic Comments
    print("⚠️ Toxic Comments Detected:")
    if results['toxicity']['bert']:
        for idx, (comment, score) in enumerate(results['toxicity']['bert'], start=1):
            print(f"  {idx}. [{score:.2f} Toxicity] {comment[:50]}...")
    else:
        print("  🎉 No significantly toxic comments detected!")
```

```
    print("\n")

    # Display Most Used Emojis
    print("😀 Most Used Emojis:")
    if results['emojis']:
        for emoji, count in results['emojis']:
            print(f"  {emoji} - {count} times")
    else:
        print("  😕 No emojis detected in comments.")
    print("\n")

    # Display Gemini Analysis
    print("□□ Gemini Analysis:")
    print(f"  Summary: {results['gemini']['summary']}")
    print(f"  Public Demands: {results['gemini']['demands']}")
    print(f"  Suggestions: {results['gemini']['suggestions']}")
    print("\n")

    print("⏳ Processing Time Analysis:")
    for feature, time_taken in results['time_complexity'].items():
        print(f"  🔹 {feature.replace('_', ' ').title()}: {time_taken:.4f} seconds")

    print("\n✅ Analysis Complete!")

async def main():
    analyzer = YouTubeCommentAnalyzer()
    video_url = input("🔗 Enter YouTube video URL: ")
    comment_limit = int(input("📝 Number of comments to analyze (max 5000): "))

    print("\n⏳ Analyzing comments... Please wait...\n")
    results = await analyzer.analyze(video_url, comment_limit)

    print_results(results)

if __name__ == "__main__":
    asyncio.run(main())
```

CHAPTER 7 : RESULT ANALYSIS

7.1 RESULT ANALYSIS

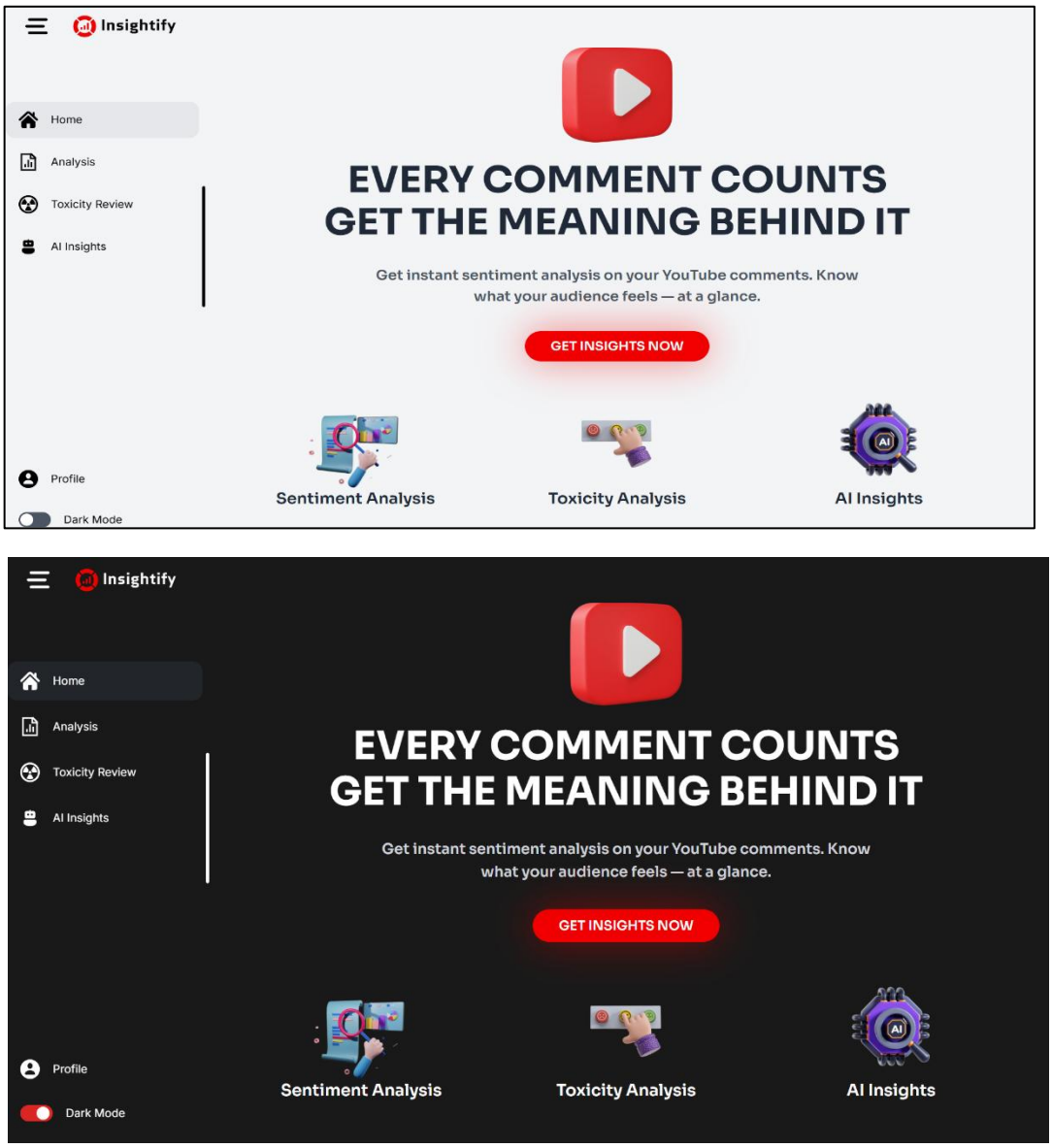

Fig 7.2 Home Page

The homepage of the **Insightify** platform is designed with a clean, modern, and user-friendly interface aimed at providing creators with instant, AI-powered insights on YouTube video comments. This page acts as the entry point to the tool's main functionalities, effectively communicating its purpose and offering intuitive navigation.

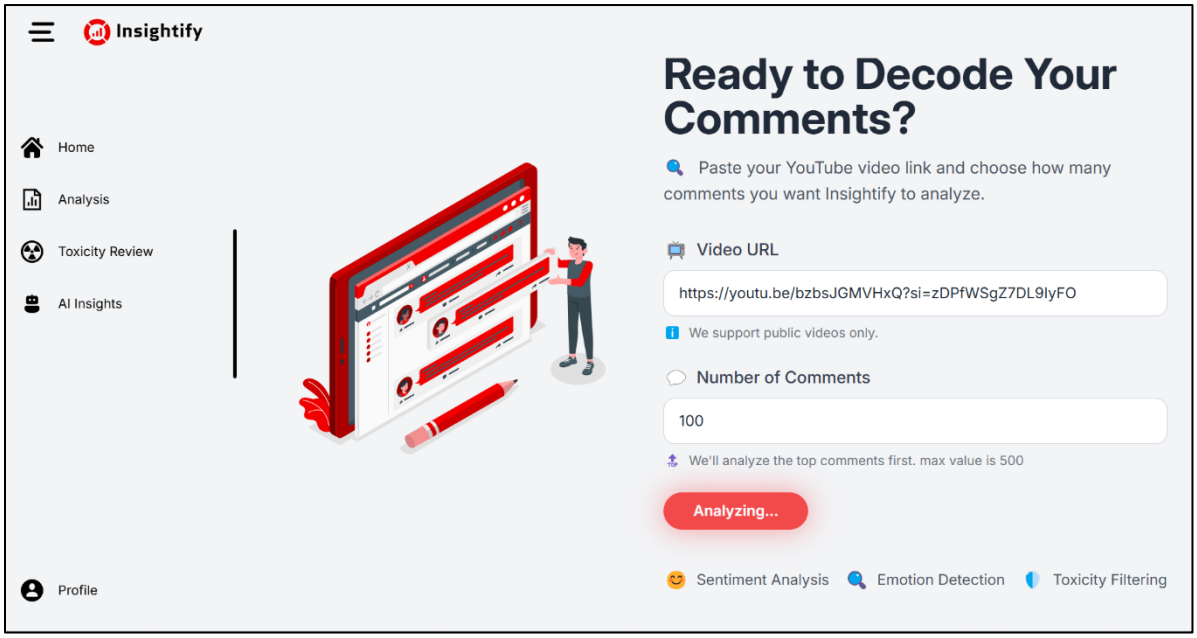

Fig 7.2 Input Page

The Analyze Input Page of Insightify allows users to paste a YouTube video link and specify the number of comments (up to 500) for analysis. Once submitted, the tool begins analyzing the top comments using Sentiment Analysis, Emotion Detection, and Toxicity Filtering.

The page features a clean UI with clear instructions, a responsive form, and a red “Analyzing…” button that indicates real-time processing.

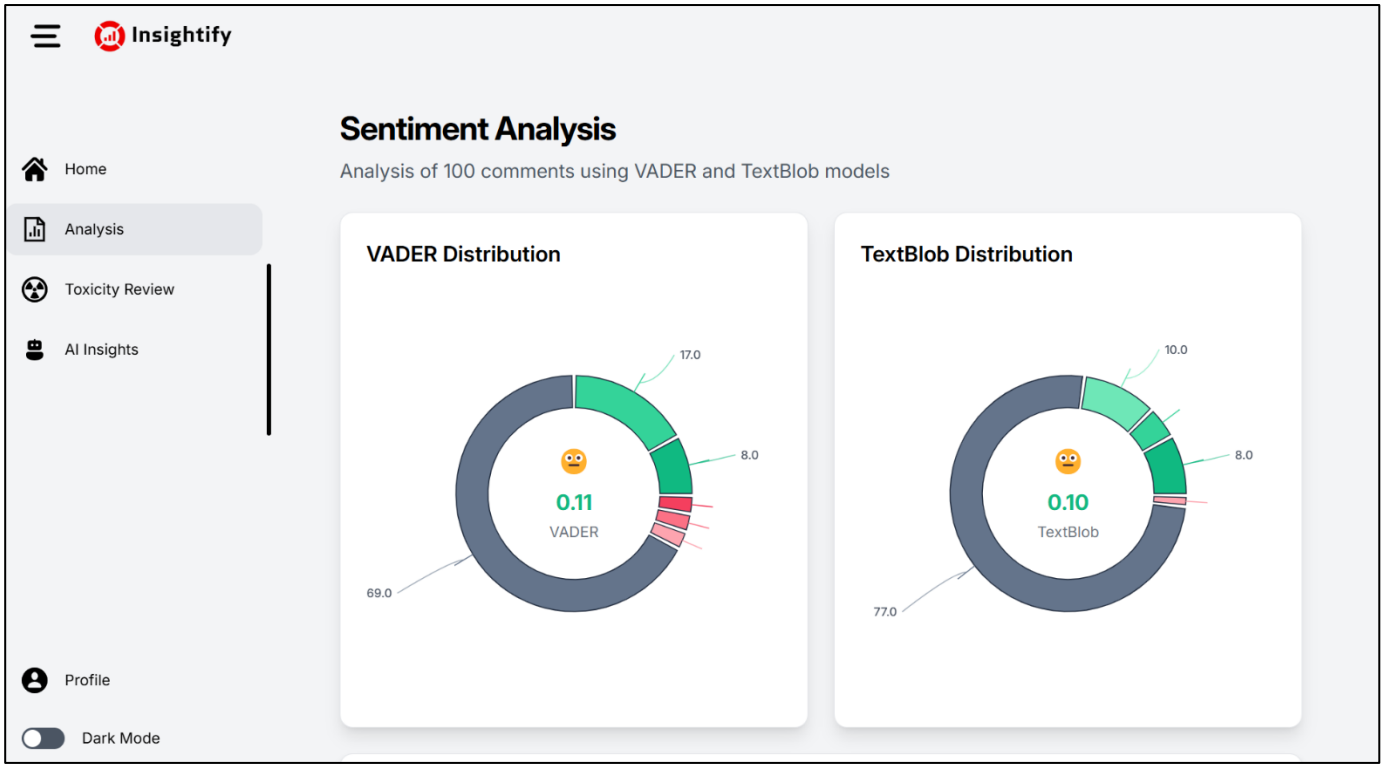

Fig 7.2 Sentiment Analysis

This page is from a sentiment analysis dashboard of a platform called Insightify. The purpose of this page is to show the results of sentiment analysis performed on 100 comments using two different models: VADER and TextBlob.

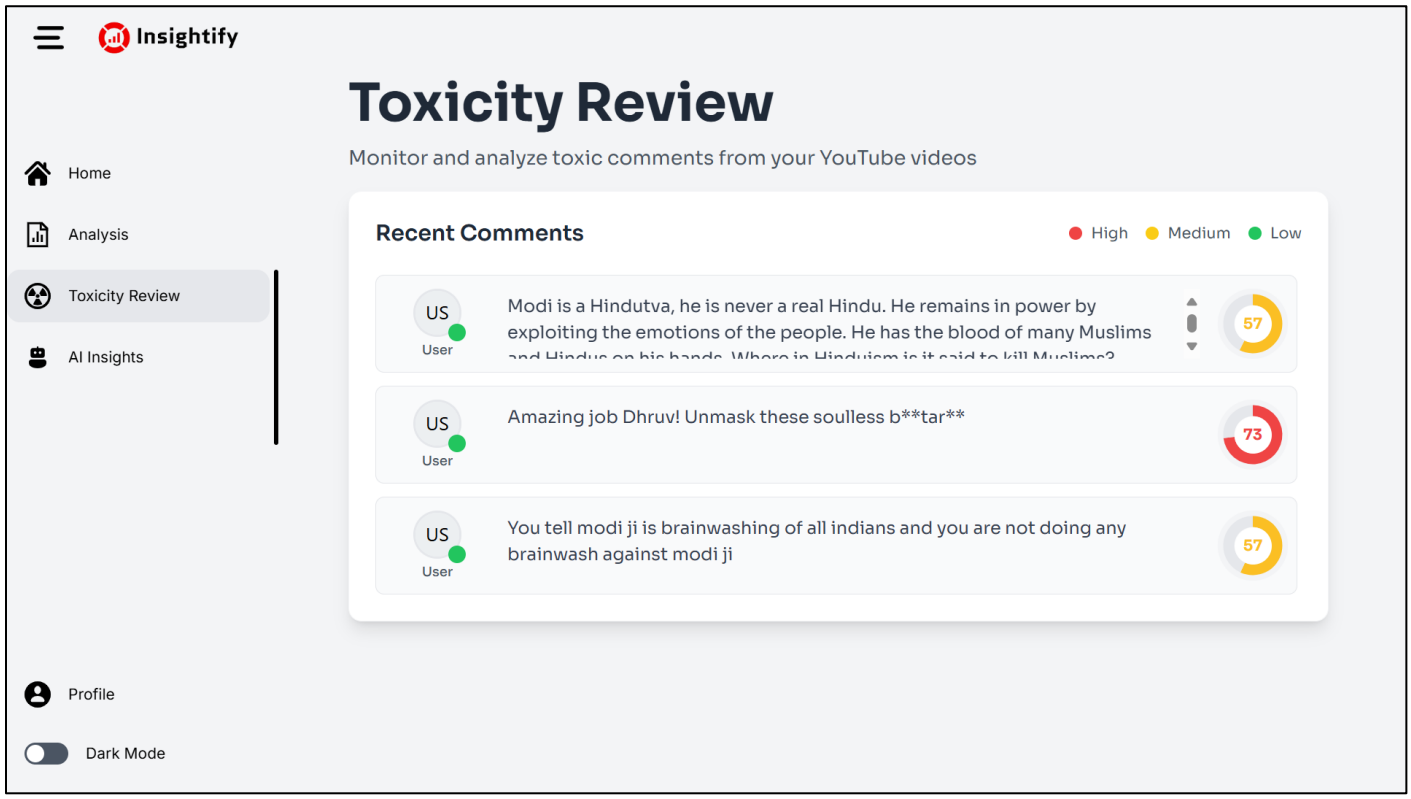

Fig 7.3 Toxicity review

Displays a list of recent user comments with toxicity scores and severity levels.

Each comment entry includes:

- **Comment text**
- **Toxicity Score Badge**:
  - Numeric score (e.g., **57**, **73**) inside a circular progress graphic
  - Color-coded severity indicator:
    - **Red** = High toxicity
    - **Yellow** = Medium toxicity
    - **Green** = Low toxicity (not shown in the current view)

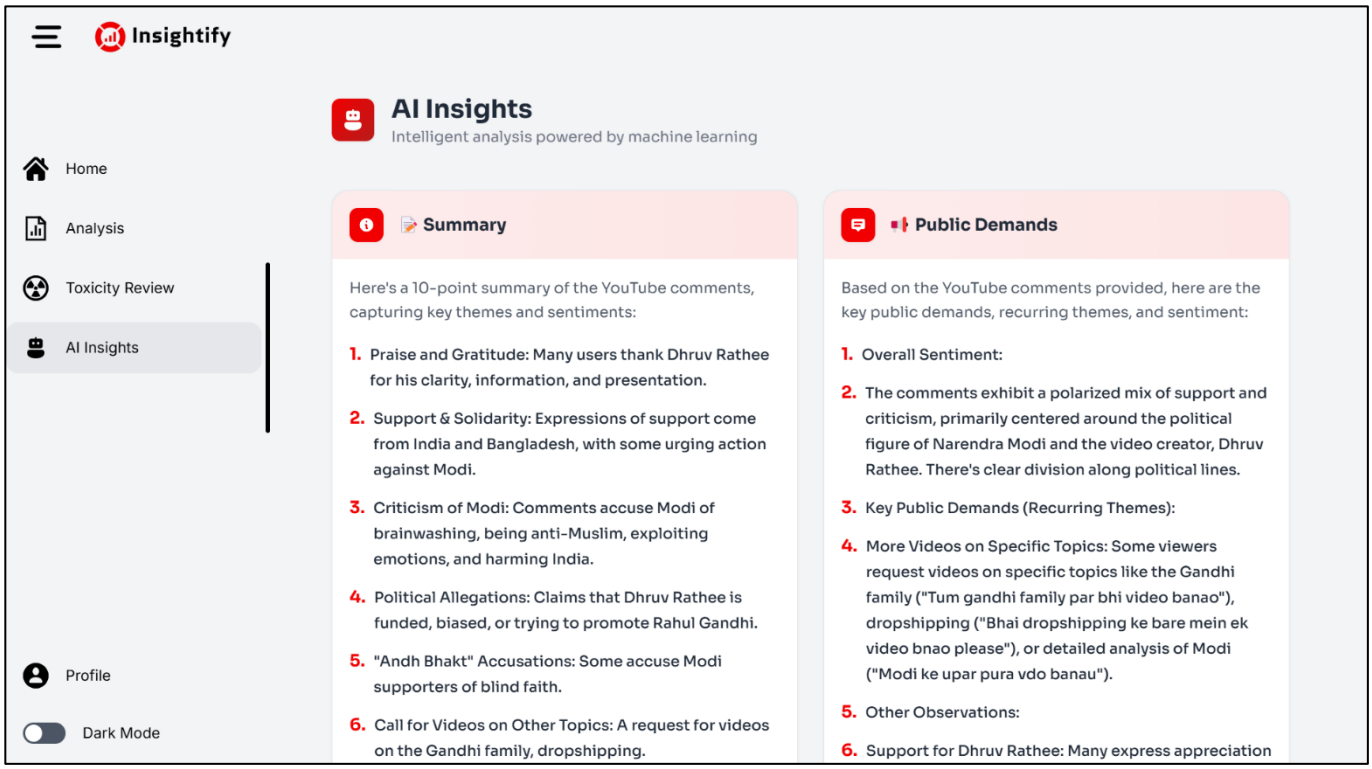

Fig 7.4. AI Insights Summary and Public Demands

This section distills vast amounts of user-generated content into concise, actionable insights. It helps creators or analysts understand public sentiment, political alignment, and viewer expectations from their audience

📄 Summary (Left Panel - Light Red Header)
A 10-point summary of analyzed YouTube comments, highlighting key

📣 Public Demands (Right Panel - Bright Red Header)
Extracts key public demands and recurring patterns from the comments:

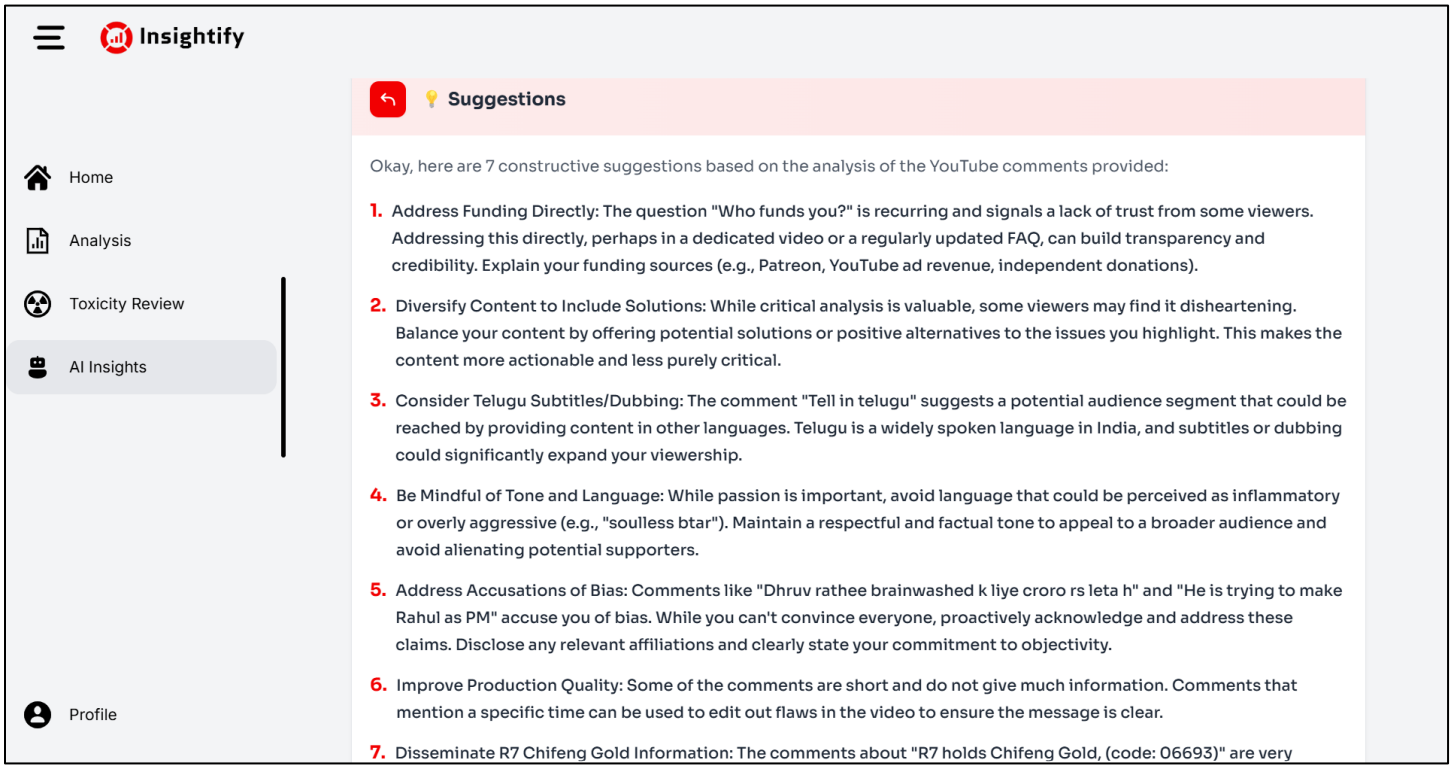

Fig 7.5 AI Insights Suggestion For YouTuber

This "Suggestions" page from Insightify's AI Insights section provides 7 constructive recommendations based on the analysis of YouTube comments

Help the content creator improve credibility, reach, tone, and impact based on public feedback.

CHAPTER 8 : ADVANTAGES AND DISADVANTAGE

8.1 ADVANTAGES

1. Quick sentiment insights for YouTubers
2. Auto-detects toxicity in comments
3. Summarizes viewer suggestions and demands
4. Saves manual analysis time
5. Clean and user-friendly interface
6. Works in real-time
7. Improves content planning with data-driven insights

8.2 DISADVANTAGES

1. Slow on large datasets (high time complexity)
2. Works only for public videos
3. Depends on third-party APIs
4. Limited support for non-English languages

CHAPTER 9: CONCLUSION AND FUTURE SCOPE:

9.1 CONCLUSION:

The YouTube Comment Analyzer – Insightify successfully provides creators with valuable insights by analyzing viewer comments through sentiment analysis, toxicity detection, and AI-powered summaries. This tool helps YouTubers better understand their audience's opinions, emotions, and suggestions, ultimately aiding in content improvement and audience engagement. The user-friendly interface and real-time processing make it accessible and efficient for everyday use.

9.2 FUTURE SCOPE:

1. Multilingual Support – Extend analysis to regional and non-English languages for broader usability.
2. Advanced Emotion Detection – Include deeper emotional layers like sarcasm, humor, and excitement.
3. Customizable Filters – Allow creators to define custom keywords, emotions, or themes to focus on.
4. Mobile App Integration – Launch mobile version for creators to access insights on the go.

CHAPTER 10 : REFERENCES

[1] Hutto & Gilbert, 2014

[2] Loria, 2018

3] Medhat et al., 2014

[4] Devlin et al., 2018

[5] Google Jigsaw, 2017

[6] Brown et al., 2020

[7] Mihalcea & Tarau, 2004

[8] Liu & Lapata, 2019