Cyberbullying Detection

Using RNN and hybrid LSTM

Introduction

Cyberbullying is a serious issue that can have devastating effects on victims. Traditional methods of detection often fall short, especially when dealing with subtle forms of cyberbullying. Deep learning techniques, such as Recurrent Neural Networks (RNNs) and hybrid models combining RNNs and Convolutional Neural Networks (CNNs), offer a promising solution. These models can analyze text data, identify patterns, and accurately classify content as cyberbullying or non-cyberbullying. By leveraging the power of deep learning, we can create safer online environments and protect individuals from the harmful effects of cyberbullying.

Software Requirements

1. **Programming Languages and Frameworks**: Python (for data processing, model development, and training).

2. **Development Tools**:
   - Google COLAB or an integrated development environment (IDE) like VS Code for coding and testing.
   - APIs for web scraping or social media API integration for data collection.

3. **Libraries for Data Processing and Analysis**:
   - Pandas and NumPy (for data manipulation and numerical operations).
   - NLTK or SpaCy (for text preprocessing, tokenization, and normalization).

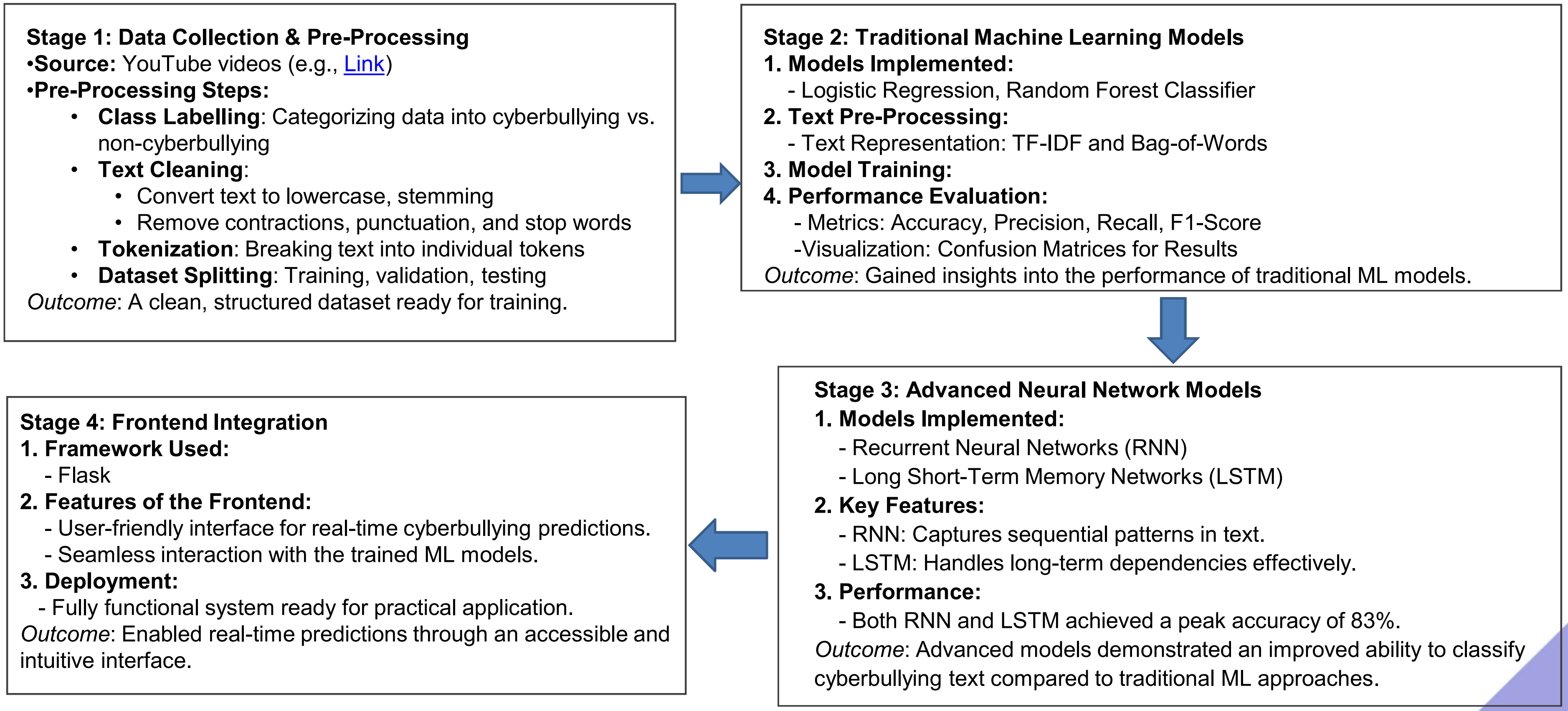
4. **Deep Learning Libraries**: TensorFlow & Libraries for hybrid architecture integration, such as Keras while using TensorFlow.

5. **Visualization Tools**: Matplotlib, Seaborn, or Plotly (for visualizing model performance and comparative analysis).

6. **Testing Tools**: Scikit-learn (for performance metrics and model validation).

7. **Hardware Requirements**: High-performance computing resources with GPUs are used to efficiently train the deep learning model.

Project Stages



Stage 1: Data Collection & Pre-Processing

•**Source:** YouTube videos (e.g., Link)

•**Pre-Processing Steps:**

- **Class Labelling**: Categorizing data into cyberbullying vs. non-cyberbullying
- **Text Cleaning**:
  - Convert text to lowercase, stemming
  - Remove contractions, punctuation, and stop words
- **Tokenization**: Breaking text into individual tokens
- **Dataset Splitting**: Training, validation, testing

Outcome: A clean, structured dataset ready for training.

Stage 2: Traditional Machine Learning Models

1. Models Implemented:

- Logistic Regression, Random Forest Classifier

2. Text Pre-Processing:

- Text Representation: TF-IDF and Bag-of-Words

3. Model Training:

4. Performance Evaluation:

- Metrics: Accuracy, Precision, Recall, F1-Score

-Visualization: Confusion Matrices for Results

Outcome: Gained insights into the performance of traditional ML models.

Stage 3: Advanced Neural Network Models

1. Models Implemented:

- Recurrent Neural Networks (RNN)
- Long Short-Term Memory Networks (LSTM)

2. Key Features:

- RNN: Captures sequential patterns in text.
- LSTM: Handles long-term dependencies effectively.

3. Performance:

- Both RNN and LSTM achieved a peak accuracy of 83%.

Outcome: Advanced models demonstrated an improved ability to classify cyberbullying text compared to traditional ML approaches.

Stage 4: Frontend Integration

1. Framework Used:

- Flask

2. Features of the Frontend:

- User-friendly interface for real-time cyberbullying predictions.
- Seamless interaction with the trained ML models.

3. Deployment:

- Fully functional system ready for practical application.

Outcome: Enabled real-time predictions through an accessible and intuitive interface.

WEB SCRAPING

Web scraping is the process of using automated tools or scripts to extract specific data from websites. It involves sending requests to web servers, receiving HTML responses, parsing the content, and extracting information based on HTML tags or patterns. This data is stored in a structured format for easy analysis. Web scraping is widely used in fields like market research, price monitoring, news aggregation, and social media tracking.

- **Sending a Request:** The scraper first sends a request to the website's server for a specific webpage (usually an HTTP GET request).
- **Retrieving the HTML:** If the request is successful, the server responds with the HTML content of the webpage.
- **Parsing the HTML:** The scraper then processes the HTML to locate and extract specific information. It involves identifying particular HTML tags, classes, IDs, or other elements that contain the desired data.
- **Extracting and Storing Data**: Once the information is identified, it’s extracted, transformed as needed, and stored in a structured format like CSV, Excel, or a database for easy access and analysis.

Pre-processing:

Pre-processing cleans and structures raw data for analysis by removing irrelevant symbols, correcting errors, and handling missing values. It ensures consistency and reduces noise, which is essential for accurate results. Missing data can be imputed, flagged, or removed depending on its relevance. Pre-processing transforms messy data into a reliable, usable format, improving the quality and performance of machine learning models or analytical tools.

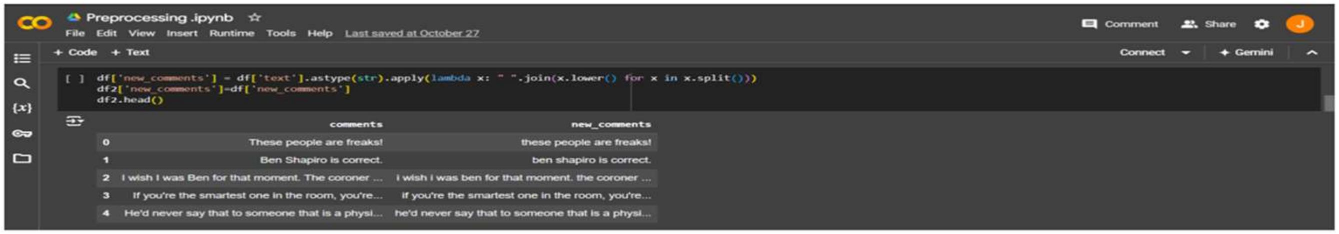
Text Normalization:

Text normalization standardizes text data for analysis in natural language processing. It includes converting text to lowercase, removing stop words (like "the" or "is") that add little value, stripping punctuation, and applying lemmatization to reduce words to their root forms. This process ensures consistency, reduces redundancy, and focuses on the most meaningful parts of the text, enabling more accurate analysis and machine learning results.

Tokenization and Text Conversion

- Tokenization converts text into smaller units, typically words or sub-words, to help the model understand the input text. This is necessary for processing text data in machine learning models.

- **Why Tokenization?** Cyberbullying detection requires analyzing the semantics and context of sentences. Breaking down text into tokens allows the model to handle each word as a meaningful unit.


Preprocessing .ipynb ☆
File Edit View Insert Runtime Tools Help Last saved at October 27
Comment
Share
J
+ Code + Text
Connect
+ Gemini

```
df['new_comments'] = df['text'].astype(str).apply(lambda x: " ".join(x.lower() for x in x.split()))
df2['new_comments']=df['new_comments']
df2.head()
```

	comments	new_comments
0	These people are freaks!	these people are freaks!
1	Ben Shapiro is correct.	ben shapiro is correct.
2	I wish I was Ben for that moment. The coroner ...	i wish i was ben for that moment. the coroner ...
3	If you're the smartest one in the room, you're...	if you're the smartest one in the room, you're...
4	He'd never say that to someone that is a physi...	he'd never say that to someone that is a physi...

1. Converts text to lowercase, making it easier to work with consistently.
2. Tokenization happens here indirectly: the text is split by whitespace (via split()) and joined back together, giving us lowercase tokens without changing word order.

ACCURACY RATE (Traditional Models)

	0.2	0.25	0.3	0.33
LOGISTIC REGRESSION	70.70 %	70.45 %	69.36 %	68.50 %
RANDOM FOREST REGRESSION	68.68 %	68.01 %	70.03 %	71.25 %
NAÏVE BAYERS CLASSIFIER	62.94 %	63.82 %	64.74 %	65.74 %
DECISION TREE CLASSIFIER	61.92 %	60.56 %	61.35 %	62.96 %
SUPPORT VECTOR MACHINE	61.92 %	60.56 %	61.35 %	62.96 %
K-NEAREST NEIGHBOUR(KNN):	65.98 %	67.07 %	67.11 %	69.44 %

Recurrent Neural Network

A **Recurrent Neural Network (RNN)** is a type of artificial neural network designed for processing sequential data. Unlike traditional feedforward neural networks, RNNs have connections that form cycles, enabling them to maintain a "memory" of previous inputs and capture temporal dependencies. This makes them well-suited for tasks where the order or context of data matters.

Advantages Of RNN

1. Sequence Handling:

RNNs are explicitly designed to handle sequential data, such as time-series data, speech, and text, making them well-suited for applications like language modeling and speech recognition.

2. Contextual Memory:

RNNs maintain a hidden state that carries information from previous steps in the sequence, enabling them to remember and process contextual information over time.

Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a specialized Recurrent Neural Network (RNN) designed to handle the vanishing gradient problem, allowing it to learn and retain long-term dependencies in sequential data. It achieves this through a more sophisticated structure that includes memory cells and gates, which control the flow of information.

Advantages Of LSTM

1. Effective Long-Term Dependency Learning
LSTMs can capture long-term dependencies in sequential data due to their gated structure, which helps retain important information over extended sequences while forgetting irrelevant data.

2. Prevention of Vanishing/Exploding Gradients
The gating mechanisms in LSTMs mitigate the problem of vanishing or exploding gradients during backpropagation, enabling the network to learn from longer sequences effectively.

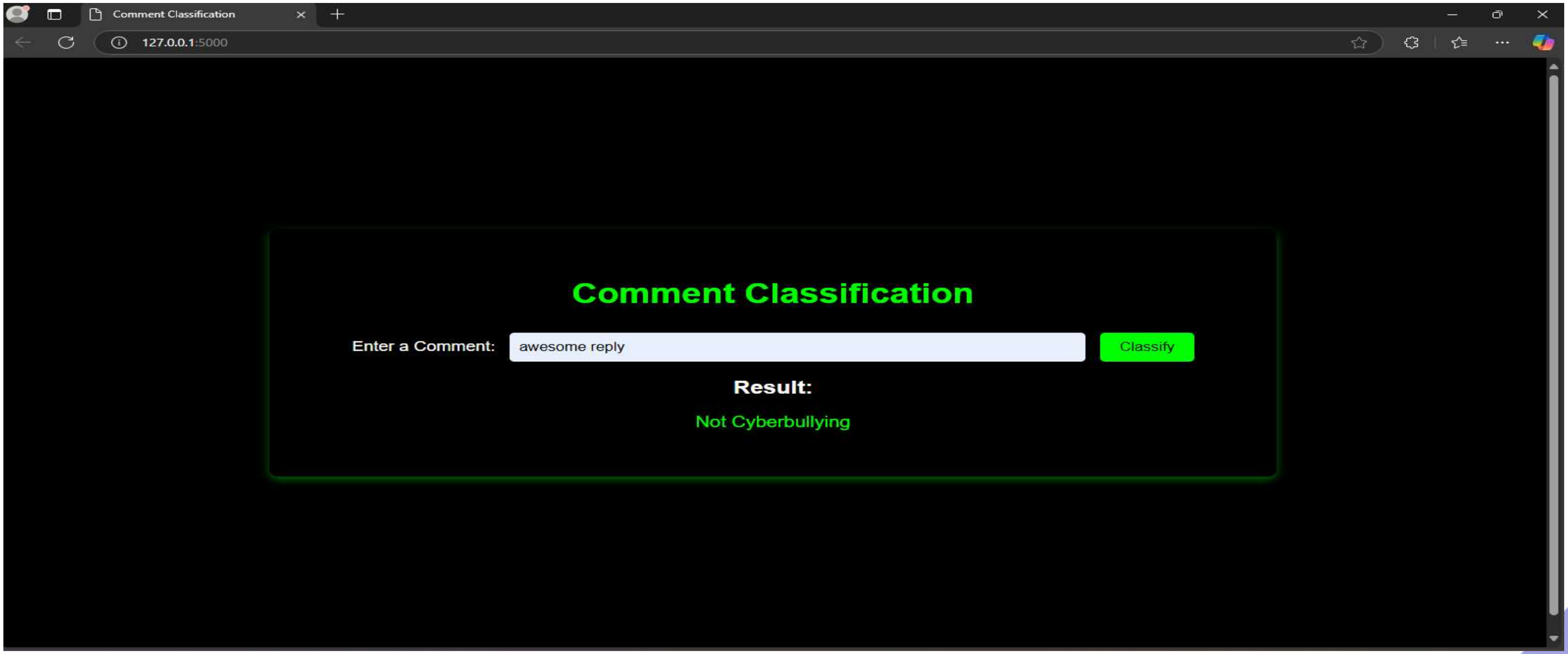
Accuracy For RNN

Test Size	PRE	POST
0.2	0.83	0.641
0.25	0.828	0.656
0.3	0.831	0.647
0.33	0.821	0.651

Accuracy For LSTM

Test Size	PRE	POST
0.2	0.83	0.83
0.25	0.81	0.82
0.3	0.81	0.76
0.33	0.80	0.71

Frontend



Comment Classification

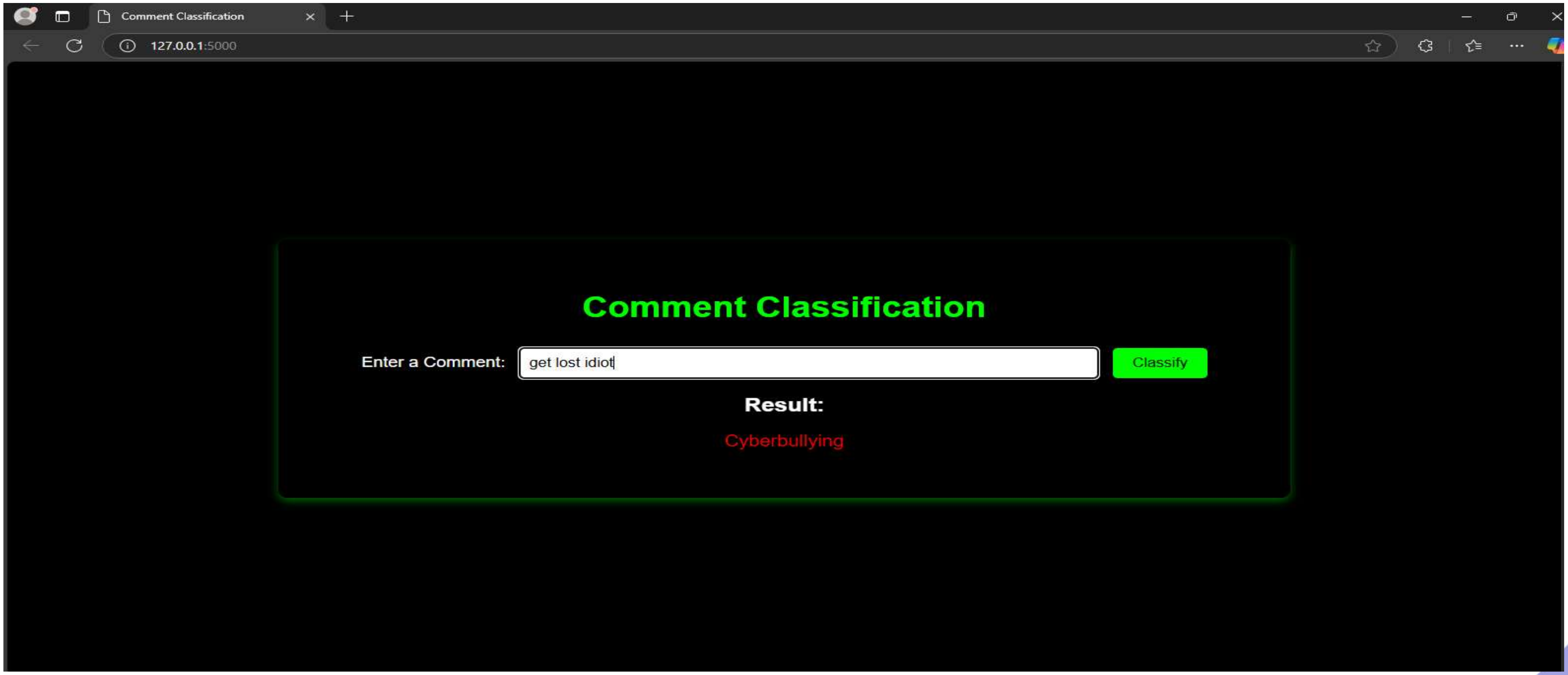
Enter a Comment: awesome reply

Result:

Not Cyberbullying

Model Predicting as Not CyberBullying

Frontend



Comment Classification

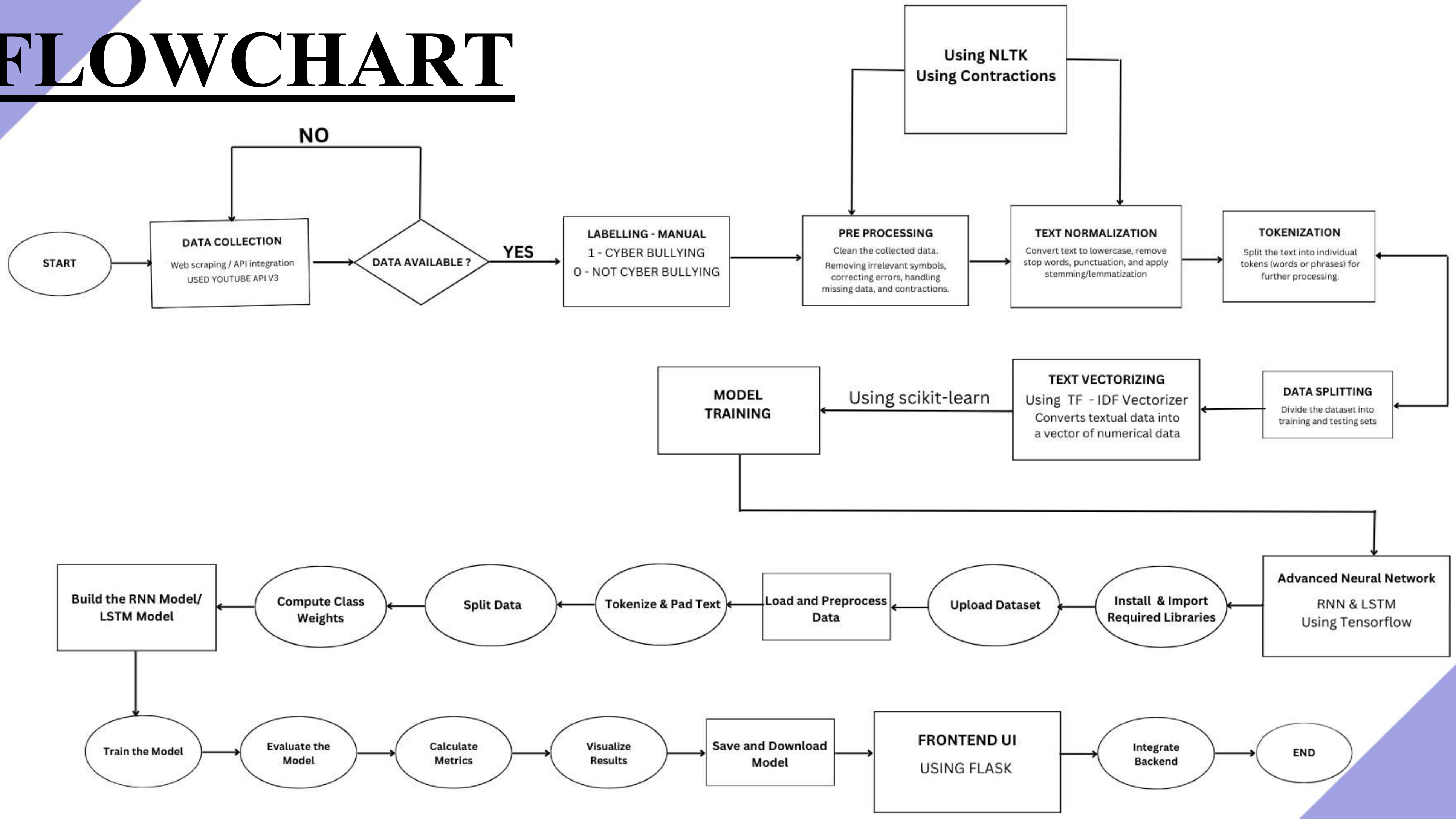
Enter a Comment: get lost idiot

Result:

Cyberbullying

Model Predicting as CyberBullying

FLOWCHART



START

DATA COLLECTION

Web scraping / API integration
USED YOUTUBE API V3

DATA AVAILABLE ?

NO

YES

LABELLING - MANUAL
1 - CYBER BULLYING
0 - NOT CYBER BULLYING

Using NLTK
Using Contractions

PRE PROCESSING

Clean the collected data.
Removing irrelevant symbols, correcting errors, handling missing data, and contractions.

TEXT NORMALIZATION

Convert text to lowercase, remove stop words, punctuation, and apply stemming/lemmatization

TOKENIZATION

Split the text into individual tokens (words or phrases) for further processing.

DATA SPLITTING

Divide the dataset into training and testing sets

TEXT VECTORIZING

Using TF - IDF Vectorizer
Converts textual data into a vector of numerical data

Using scikit-learn

MODEL TRAINING

Advanced Neural Network

RNN & LSTM
Using Tensorflow

Install & Import Required Libraries

Upload Dataset

Load and Preprocess Data

Tokenize & Pad Text

Split Data

Compute Class Weights

Build the RNN Model/ LSTM Model

Train the Model

Evaluate the Model

Calculate Metrics

Visualize Results

Save and Download Model

FRONTEND UI
USING FLASK

Integrate Backend

END

Conclusion

Cyberbullying, a pervasive issue in the digital age, poses significant threats to individuals, especially adolescents. Traditional methods often struggle to detect subtle forms of cyberbullying.

Deep learning, particularly Recurrent Neural Networks (RNNs) and hybrid models combining RNNs and Convolutional Neural Networks (CNNs), offer a promising solution. RNNs effectively capture long-term dependencies within text, while CNNs extract local features. These hybrid models can accurately identify cyberbullying patterns, even in complex and evolving online interactions.

By training these models on large datasets and fine-tuning their hyperparameters, we can achieve state-of-the-art performance in cyberbullying detection. This enables us to create safer online environments and protect individuals from the harmful effects of cyberbullying.

Team Members

- Janvi Singh
- Praneetha Sai
- Dona Krishna sai
- Ayush Srivastava
- Kalpesh Mahajan