Raw Data ← CSVs, Logs, DB

↓

Preprocessing Layer ← Clean, Normalize, Timestamp, GeoTag

↓

AI Modules

─────────────

Sentiment Analysis

NER

Topic Detection

Semantic Embeddings

Post-Incident Summary

Geo-Clustering

────────────

↓

Vector DB / SQL

↓

Frontend UI ← Streamlit or Next.js

Dashboards / Search Tools

**User Profile Summary (AI)**

**Goal:**

***User Metadata***

user\_id: Unique identifier for the user

username: Telegram handle (if available)

display\_name: Optional, user’s name in contact

chat\_ids: List of chats/groups user is in

***Activity Features***

message\_count: Total number of messages sent

active\_days: Number of days user has sent messages

messages\_per\_day: Avg. messages per active day

first\_message\_time, last\_message\_time: Range of activity (can track account age/activity gap)

time\_distribution: Histogram of message times (morning/evening/night person)

active\_chat\_count: No. of distinct chats/groups the user participates in

reply\_count No. of messages that are replies (shows interaction)

forward\_count Number of forwarded messages

***Sentiment & Emotion Features***

avg\_sentiment\_score: Based on sentiment model (positive/negative/neutral score)

dominant\_emotion: Most frequent emotion (anger, joy, fear, etc.) if using emotion classifiers

sentiment\_variance: Range of sentiment (emotional volatility)

***Suggested Tools & Libraries***

NLP: spaCy, NLTK, TextBlob, Hugging Face

Sentiment: VADER, RoBERTa, GoEmotions

Embeddings: SentenceTransformers

Clustering: scikit-learn (KMeans, DBSCAN), UMAP

Visualization: Seaborn, Matplotlib, Plotly, Streamlit

Topic Modeling: BERTopic, Gensim

Toxicity: Perspective API, Detoxify, HateBERT

**Build comprehensive behavioral and linguistic profiles of users to understand:**

* Engagement patterns
* Tone and language style
* User influence or roles
* Communication interests/topics

**AI Techniques:**

* Linguistic Analysis: Vocabulary richness, average sentence length, use of emojis, sentiment trends.
* Behavioral Features: Frequency of messages, peak activity hours, media usage, reply-to patterns.
* User Clustering: Use message embeddings + KMeans or DBSCAN to group similar users.

**Tools:**

* Python, Pandas, Scikit-learn
* Sentence-Transformers (BERT)
* Matplotlib/Seaborn/Plotly for visual insights

**Example Insights:**

* "User A is highly active late at night, uses formal tone, and often discusses politics."
* "User B is a community influencer with frequent replies and high message counts."

**Natural Language Processing (NLP)**

| **Task** | **Technique** | **Purpose** |
| --- | --- | --- |
| **Tokenization & Parsing** | spaCy, nltk | Break down text into words, sentences, and parts of speech. |
| **POS tagging / Dependency parsing** | spaCy | Understand the grammar and structure of user messages. |
| **TF-IDF / KeyBERT** | Keyword extraction | Identify important or unique words per user. |
| **NER (Named Entity Recognition)** | spaCy, BERT-based NER | Extract people, places, dates, etc. |
| **Emotion classification** | GoEmotions, DistilBERT fine-tuned | Detect emotions like joy, sadness, anger in messages. |

**Sentiment Analysis (AI)**

**Goal:** To determine whether a message expresses a positive, negative, or neutral sentiment. We can also go deeper to detect emotions like joy, anger, sadness, fear, etc. Classify each message as **Positive**, **Negative**, or **Neutral**, helping track user moods or community sentiment over time.

**AI Techniques:**

* **Pretrained Sentiment Models**:
  + VADER (for short/informal messages)
  + HuggingFace models like cardiffnlp/twitter-roberta-base-sentiment
* Fine-tuning possible on Telegram-specific text if slang/abbreviations dominate.

**Tools:**

* NLTK (for VADER)
* Hugging Face Transformers (RoBERTa, BERT)
* TextBlob (for basic analysis)

**Example Insights:**

* Detect surges in negative sentiment after a controversial event.
* Identify overly aggressive or toxic users for moderation.

**Named Entity Recognition (NER) (AI)**

**Goal:**

Extract named entities from messages such as:

* **People**, **Organizations**, **Locations**, **Dates**, **Events**

**AI Techniques:**

* Pre-trained NER models:
  + spaCy (en\_core\_web\_trf)
  + Transformers (e.g., dslim/bert-base-NER)
* Fine-tuning for better results with casual/sparse text formats

**Tools:**

* spaCy
* Hugging Face Transformers
* Stanza (Stanford NLP)

**Example Insights:**

* "User X mentioned 'Tesla' and 'Elon Musk' in political conversations."
* Location entity spikes after travel or crisis-related messages.

**Topic Detection (AI)**

**Goal:**

Automatically identify and group **dominant themes** or topics discussed in chats.

**AI Techniques:**

* **LDA (Latent Dirichlet Allocation)** – Identifies underlying topics in a collection of documents by assuming each document is a mixture of topics, and each topic is a distribution of words.
* **BERTopic** – combines BERT embeddings + clustering + class-based TF-IDF (Term Frequency-Inverse Document Frequency).
* **Zero-shot classification** using NLI models for predefined topic labels.

**Tools:**

* Gensim (LDA {Latent Dirichlet Allocation}): It is a technique to extract the hidden topics from large volumes of text.
* BERTopic (Highly recommended): Topic modeling technique that leverages, transformers and c-TF-IDF to create dense clusters allowing for easily interpretable topics whilst keeping important words in the topic descriptions.
* Scikit-learn for dimensionality reduction (e.g., UMAP {Uniform Manifold Approximation and Projection})

**Example Insights:**

* Detect trending topics like "Elections", "Crypto", "Exam results"
* Compare topic diversity between group chats vs private conversations