TrendStory: Real-Time AI Story Generator Using gRPC and Ollama

# Abstract

TrendStory is a full-stack AI-powered NLP application designed to extract real-time trending topics from platforms like YouTube and Google News, analyze them, and convert them into coherent stories using a locally hosted LLM (LLaMA3 via Ollama). It leverages a client-server gRPC architecture for fast communication, includes a user-friendly Gradio interface, and uses advanced NLP techniques for scoring, filtering, and visualizing data.

# Objectives

- Extract trending topics from online platforms.  
- Analyze trends using sentiment, keyword density, topic classification.  
- Generate creative stories with customizable tone, theme, and style.  
- Visualize relationships using statistical and logical graphs.  
- Maintain full control over LLM execution with local deployment via Ollama.

# System Architecture

The system consists of the following components:  
1. Frontend (PC1): Gradio-based UI for user input and displaying the generated story and graphs.  
2. gRPC Client: Sends user-selected preferences to the server using a TrendStoryRequest.  
3. gRPC Server (PC0): Receives and validates requests. Calls the story\_maker() function to process trends, build graphs, and craft story prompts.  
4. StoryMaker Module: Filters and analyzes trends. Builds semantic graphs using TF-IDF, keyword similarity, and temporal logic. Calls the local LLM (LLaMA3 via Ollama) to generate a story.  
5. TrendExtrAnalyzer.py: A scheduled module (via cron) that fetches trends hourly from YouTube and Google News. Performs preprocessing: translation, language detection, TF-IDF, sentiment scoring, and topic classification.  
6. TrendCleaner.py: Keeps only the top 3 trends in trends.json for testing purposes.

# Data Flow

User Preferences (tone, theme, style, region, category)  
↓  
Gradio UI → gRPC Client → gRPC Server → StoryMaker  
↓  
Trends + Graphs + Prompts → Ollama (LLaMA3) → Story  
↓  
gRPC Response → Gradio UI

# Technology Stack

Backend:  
- Python 3.9+  
- gRPC & Protocol Buffers: Server-client communication  
- Ollama (LLaMA3): Local LLM generation  
- NetworkX + Matplotlib: Graph construction and visualization  
- NLTK, YAKE, TF-IDF: Sentiment & keyword analysis  
  
Frontend:  
- Gradio: Web UI for interaction  
  
Infrastructure:  
- Docker: Containerization for reproducible environments  
- Cron: Scheduling periodic trend updates

# Key Features

- gRPC-powered concurrency: Supports both synchronous and asynchronous servers  
- Trend Mining: YouTube Data API, Google News RSS Feeds  
- Semantic Graphs: Statistical (TF-IDF), Logical (Sentiment, keyword)  
- Multilingual Support: Roman Urdu and English with translation fallback  
- Modular Story Customization: Themes, Tones, Styles

# gRPC Protocol Design

service TrendStoryService {  
 rpc GetStory (TrendStoryRequest) returns (TrendStoryResponse);  
}  
  
message TrendStoryRequest {  
 string tones = 1;  
 string themes = 2;  
 string styles = 3;  
 string language = 4;  
 string category = 5;  
 string region = 6;  
}  
  
message TrendStoryResponse {  
 string response = 1;  
 bytes image\_data\_clean = 2;  
 bytes image\_data\_messy = 3;  
}

# Sample Cron Setup

# Runs TrendExtrAnalyzer.py every hour  
0 \* \* \* \* /usr/bin/python /home/app/TrendExtrAnalyzer.py >> /home/app/cron\_output.log 2>&1

# Testing and Validation

- Validated correct functioning of async gRPC vs. threading-based server.  
- Verified hourly trend extraction via cron and correct file writes to trends.json.  
- Debugged gRPC field mismatches and prompt overflows to ensure prompt length limits for LLaMA3 were respected.

# Conclusion

TrendStory successfully integrates real-time data collection, NLP-driven analysis, and LLM-based narrative generation in a modular architecture. By using gRPC for low-latency communication and Ollama for offline LLM inference, it offers a scalable and private way to convert global trends into human-like stories.