ChatGPT

Handoff for auto-video-generator topic source enhancements

Purpose

This document outlines changes needed to extend the **auto-video-generator** project so that topic generation does not rely solely on Google Trends. The goal is to allow the project to continue generating videos even when trend RSS feeds fail and to support AI-generated monthly topics and evergreen lists. This handoff provides enough context and tasks for Codex to generate a TODO list and progress the work without further guidance.

Background

The current implementation includes a script `scripts/auto_trend_pipeline.py` that can fetch trending keywords via Google Trends RSS (`--source youtube`) and optionally a limited LLM-based mode (`--source llm`). The script then uses `generate_script_from_brief.py` to turn a topic brief into a YAML script and uses ffmpeg to build a video. It supports parameters like `--interval-minutes` to run periodically.

Past work has shown that trends RSS can be unreliable, and the project requires an alternate source of topics. You suggested introducing additional topic sources and a pre-processing step that produces “seed phrases” before script generation to improve quality. You also want to maintain backwards compatibility, avoid widespread refactoring and keep the modifications within the existing script when possible.

Proposed Enhancements

Topic Sources and Fallback

Introduce multiple topic supply mechanisms, with a prioritized fallback. A new configuration parameter `topic_source` should allow specifying a primary source and fallback sources. Sources to support:

- **trends**: Existing Google Trends RSS / YouTube trending retrieval (current default).
- **ai_monthly**: A new mode where an LLM generates a list of topics that feel like trending topics from the last month. The LLM should return JSON containing topics. This mode does not use the RSS feed at all.
- **evergreen**: A locally stored dictionary of evergreen topics categorized by theme (e.g., “人体”, “宇宙”, “歴史”) that can be used when both trends and ai_monthly fail.

Add simple duplicate filtering to avoid repeating the same topic within a configurable window (e.g., 30 days). For Phase 1, use exact string matches; a more sophisticated similarity filter can be added later.

Implement fallback logic: attempt the primary source; if it returns an empty list or errors, try each fallback in order. This allows the pipeline to continue running unattended.

AI Monthly Topics Implementation

Within `--source llm` handling in `auto_trend_pipeline.py`, skip RSS retrieval and call the LLM directly to produce a list of topics. The LLM should be instructed to output a JSON object with a `topics` array. Keep the current brief templating approach so that the generated topics can be fed into `generate_script_from_brief.py`.

Prompts should ensure the topics are short (around 12–18 characters), safe (no medical advice, hate, violence, etc.) and diverse. It is acceptable to let the LLM produce both the keyword and a short brief.

Seed Phrases (Optional Phase 2)

The quality of scripts can be improved by generating "seed phrases"—short factual fragments—before the final script generation. This can be implemented by adjusting the prompt in `generate_script_from_brief.py` to instruct the LLM to produce short factual fragments as part of the brief. In Phase 1, do not add new fields to the YAML schema; instead adjust the prompt only. If deeper integration is needed later, we can introduce a `seed_phrases` field and modify the script generator to handle both direct and seed modes.

Configuration Example

Users should be able to specify configuration via YAML or command-line flags similar to:

```
topic_source:
  primary: trends
  fallback:
    - ai_monthly
    - evergreen
topic_count: 20
dedupe:
  enabled: true
  window_days: 30
script_mode: direct  # or "seed"
```

For now, integrate the new logic into the existing CLI arguments for `auto_trend_pipeline.py` (e.g., by extending `--source llm` internally) rather than adding a new CLI flag.

Branch and Compatibility

To avoid disrupting current operations, implement these changes on a feature branch (e.g., `feature/topic-source-fallback`). Ensure that `--source youtube` retains exactly the same behaviour as before. The new functionality should be triggered only when using `--source llm` or by an explicit configuration file. After thorough testing, the branch can be merged.

Deliverables for Codex

Codex should focus on the following tasks and break them down into a TODO list:

1. **Analyse the current repository** to understand how `scripts/auto_trend_pipeline.py` and `generate_script_from_brief.py` are structured. Pay attention to the existing `--source llm` path and where topics are passed through the pipeline.
2. **Implement the topic source fallback** inside `auto_trend_pipeline.py` for `--source llm`. The fallback should first call the existing LLM prompt to generate monthly topics; if it returns no topics (or errors), load evergreen topics from a local YAML file; and only if both fail should it default to the existing trends logic.
3. **Add dedupe logic**: maintain a simple JSON file (e.g., under `work/` or `outputs/`) that records previously used topics and skip duplicates within a 30-day window. Phase 1 can use exact string matches.
4. **Adjust the script generation prompt** (Phase 2) in `generate_script_from_brief.py` to incorporate seed phrases by describing the structure of the brief: include short factual fragments that the script generator can rearrange. A future phase may add a `seed_phrases` field to the YAML schema.
5. **Update README.md** to document the new source modes and configuration options, providing examples on how to run the tool without Google Trends.
6. **Create or update tests** if they exist, or manually validate that:
7. `python scripts/auto_trend_pipeline.py --source youtube` still fetches and processes Google Trends correctly.
8. `python scripts/auto_trend_pipeline.py --source llm` runs entirely without trends, using AI monthly topics and fallback.
9. Duplicate topics are avoided when configured.
10. **Use a branch** for the implementation and ensure that a pull request summarises the changes. Make sure not to modify the behaviour of existing modes unless explicitly intended.

Feel free to break down these tasks into a TODO list with smaller steps (e.g., reading YAML configuration, loading evergreen topics, implementing dedupe functions, writing new prompts, adding tests). This handoff document should give you enough context to proceed without additional clarification.
