# Engineering Automated Trend Detection Pipelines for the Japanese Market (2026 Edition)

## 1. Executive Summary

In the hyper-competitive landscape of automated content generation, the ability to identify, validate, and act upon emerging trends before they reach saturation is the defining variable of success. For "Ovi English School," a platform designed to produce timely, trend-based English learning podcasts, the technical challenge is twofold: traversing an increasingly hostile data acquisition environment and interpreting noisy, high-velocity signals with precision. This report provides an exhaustive technical analysis and strategic roadmap for engineering a robust trend detection system specifically tailored for the Japanese market in 2026.

The operational environment for scraping and data harvesting has shifted dramatically over the last two years. The era of permissive, open-source data collection via libraries like pytrends has effectively ended, casualties of Google's sophisticated anti-bot countermeasures and the aggressive monetization of API access. As of early 2026, the official Google Trends API remains a restricted Alpha product, inaccessible for general commercial deployment. Consequently, the optimal path forward necessitates a "Build vs. Buy" decision that heavily favors commercial scraping infrastructure—specifically SerpApi and ScrapingBee—which offer the necessary resilience against IP blocking and browser fingerprinting.

For the Japanese market, reliance on Google alone is insufficient. The digital ecosystem in Japan is bifurcated, with Yahoo! Japan retaining significant cultural gravity as a real-time news hub. A dual-source strategy that ingests signals from both Google and Yahoo! Japan, validated by secondary metrics from YouTube and news APIs, is essential for high-fidelity trend detection. Furthermore, the linguistic complexity of the market—requiring the simultaneous tracking of Kanji, Hiragana, and Katakana scripts—demands a sophisticated keyword expansion strategy to capture the true volume of a trend.

The recommended architecture leverages the computational efficiency of the Apple Silicon M5 stack for local statistical processing. By integrating Python’s advanced data science libraries (Pandas, SciPy) for algorithmic scoring—utilizing metrics such as Robust Z-Scores, Velocity, and Acceleration—within an n8n orchestration layer, Ovi English School can achieve a fully automated, resilient pipeline. This report details the specific pricing models, fallback chains, and algorithmic implementations required to transform raw search data into actionable content insights.

## 2. The State of Google Trends Access in 2026

The capability to programmatically access Google Trends data is the foundational requirement for the Ovi English School pipeline. However, the mechanism for this access has undergone a radical transformation, moving from community-maintained open-source scripts to a paid, adversarial ecosystem.

### 2.1. The Official Google Trends API: Status and Limitations

For over a decade, the developer community anticipated a public, documented API for Google Trends. By early 2026, Google has technically released this product, but its availability remains severely restricted, rendering it non-viable for most independent developers and startups.

#### 2.1.1. Availability and Access Control

The official Google Trends API is currently in an **Alpha testing phase**.1 Access is not granted through the standard Google Cloud Console self-service workflows found for services like the YouTube Data API or Google Maps Platform. Instead, it is gated behind an application process reserved primarily for large-scale enterprise partners, academic institutions with verified research credentials, and major publishers.3 Reports from the developer community indicate that even legitimate business applications are frequently waitlisted indefinitely. Google’s roadmap suggests a broader rollout may occur later in 2026, potentially coinciding with major industry events, but relying on this timeline presents an unacceptable critical path risk for immediate development.4

#### 2.1.2. Data Characteristics and Constraints

For those few with access, the API offers distinct advantages but also significant limitations compared to web-scraped data. The API provides "consistently scaled" data, which differs from the relative 0-100 scaling found on the public web interface.3

* **Consistently Scaled Data:** This feature allows for the comparison of search interest across different time periods and regions without needing to perform complex normalization or "stitching" of multiple requests. It addresses the "normalization problem" where a query for "iPhone" might appear as 100 in a weak month and 100 in a strong month if queried separately.
* **Quota Limits:** Early testers report strict rate limits (quotas), designed to prevent the extraction of high-granularity data at scale. The API is optimized for broad trend analysis rather than the minute-by-minute tracking required for identifying viral spikes in real-time.
* **Authentication:** Integration requires full OAuth 2.0 flows and Google Cloud project linkage, adding administrative overhead compared to simple API key authentication.5

### 2.2. The Obsolescence of Open Source: The Fall of pytrends

The pytrends Python library was, for many years, the industry standard for unofficial Google Trends access. As of 2026, it is considered **deprecated and unreliable** for production environments.6 Its decline is not merely a lack of maintenance but a result of fundamental changes in Google's defensive posture.

#### 2.2.1. The Mechanism of Failure

The primary failure mode for pytrends and similar requests-based scrapers is the immediate triggering of **HTTP 429 (Too Many Requests)** errors.7 This is not a simple rate limit; it is a bot detection trigger.

* **TLS Fingerprinting:** Google's edge servers now analyze the Transport Layer Security (TLS) Client Hello packet. Standard Python libraries like requests or urllib have distinct TLS fingerprints (JA3 signatures) that differ from legitimate web browsers (Chrome, Safari, Firefox). When Google detects a mismatch between the declared User-Agent and the actual TLS fingerprint, it flags the request as automated.
* **Cookie and Token Validation:** The Trends backend increasingly relies on complex, obfuscated tokens generated via JavaScript execution on the client side. pytrends, being a simple HTTP client, does not execute JavaScript and thus fails to generate the necessary proof-of-work tokens.9
* **Data Poisoning:** A more subtle defense mechanism observed by developers is the serving of "fake" or "flat" data. Instead of blocking the request, Google returns a valid JSON response containing zeros or flat lines. This "data poisoning" is particularly dangerous for automated systems like Ovi English School, as it may not trigger error logs but will result in missed trends and irrelevant content.1

#### 2.2.2. Maintenance Status

The pytrends repository on GitHub has seen minimal activity, with the last significant commits dating back nearly two years. The issue tracker is filled with unaddressed reports of blocking and authentication failures. The project has effectively been archived by its maintainers, signaling that the "cat and mouse" game with Google has become too resource-intensive to sustain as a free open-source project.6

### 2.3. Anti-Scraping Defenses in 2026

Understanding the adversary is crucial for engineering a bypass. In 2026, Google employs a multi-layered defense-in-depth strategy to protect its data:

1. **IP Reputation Filtering:** The most basic layer blocks requests originating from known data center IP ranges (AWS, Google Cloud, Azure, DigitalOcean). Trends data can effectively only be accessed via **Residential IPs**—IP addresses assigned by ISPs to home users.
2. **Browser Consistency Checks:** For requests that pass the IP filter, Google performs consistency checks. Does the request include the correct Sec-Ch-Ua headers? Does the TCP window size match the OS claimed in the User-Agent?
3. **Behavioral Analysis:** For high-volume scrapers, Google analyzes query patterns. A distinct lack of "human" behavior—such as mouse movements, random timing jitter, or navigation through the UI—can trigger a CAPTCHA or a "soft ban" (temporary IP block).

### 2.4. Strategic Implication for Ovi English School

Given the unavailability of the official API and the unreliability of pytrends, Ovi English School must adopt a **commercial scraping strategy**. Attempting to maintain a custom scraping infrastructure (managing headless browsers, rotating residential proxies, and updating parsing logic) would require a dedicated full-time engineer. Using a specialized API provider offloads this complexity, transforming an engineering problem into a predictable operational expense.

## 3. Commercial Data Acquisition Infrastructure

To reliably feed the trend detection algorithms, Ovi English School requires a data provider capable of navigating Google's defenses. The market for these services is mature, with several major players offering distinct advantages. The analysis below specifically evaluates providers based on the requirement for **Japanese market coverage** and a daily volume of **100-1,000 requests** (approx. 30,000 requests/month).

### 3.1. Provider Capability Analysis

The following table summarizes the key metrics for the leading candidates: SerpApi, ScrapingBee, Bright Data, and Oxylabs.

| **Feature / Metric** | **SerpApi** | **ScrapingBee** | **Bright Data** | **Oxylabs** |
| --- | --- | --- | --- | --- |
| **Primary Mechanism** | Structured API (Middleware) | Headless Browser (SaaS) | SERP API & Proxy Infra | SERP Scraper API |
| **Google Trends Support** | **Native Endpoint** (engine=google\_trends) | Via General Web Scraping | Via SERP API | Via SERP Scraper |
| **Japan Reliability** | **High** (gl=jp, hl=ja) | High (Premium Proxies) | **Very High** (Top Tier IPs) | High |
| **Pricing Model** | Search Volume (Per Request) | Credit Based (Resource Usage) | Pay-As-You-Go / CPM | Per Request / Tiered |
| **Est. Cost (30k req/mo)** | **$275/mo** (Big Data Plan) | **~$249/mo** (Business Plan) | ~$100 - $300 (Depends on CPM) | ~$200 - $500 |
| **Response Format** | Clean, Normalized JSON | Raw HTML (requires parsing) | JSON / HTML | JSON / HTML |
| **Latency** | Low (Optimized Parsing) | High (Browser Rendering) | Low | Low |
| **Caching** | **Free (1-hour cache)** | No (Charged per call) | No | No |

### 3.2. Deep Dive: SerpApi

SerpApi is the strongest candidate for this specific use case due to its developer-centric design that abstracts the scraping process entirely.

* **Architecture:** SerpApi does not just provide a proxy; it acts as a normalization layer. You send a clean HTTP request to SerpApi, and it handles the complex interaction with Google, parsing the resulting DOM or hidden API responses into a predictable JSON format.
* **Google Trends Specifics:** Unlike general scraper APIs, SerpApi has a dedicated google\_trends engine. This means Ovi English School's codebase does not need to maintain CSS selectors or parsing logic. If Google changes its frontend code, SerpApi updates its parser, ensuring zero downtime for your application.10
* **Japan Configuration:** The API supports the gl (Country) and hl (Language) parameters natively. Setting gl=jp and hl=ja ensures that the trend data returned is specifically for users in Japan, filtering out global English searches that might originate from US military bases or tourists.12
* **Cost Efficiency:** The "Big Data" plan ($275/mo for 30,000 searches) fits the upper bound of the requirement (1,000/day). Crucially, SerpApi offers **free cached searches**. If the system queries for "trending topics in Japan" every 10 minutes, but the data hasn't changed on Google's end (or within SerpApi's 1-hour cache window), the request does not count toward the quota. This allows for high-frequency polling without budget overruns.10

### 3.3. Deep Dive: ScrapingBee

ScrapingBee takes a different approach, offering a "Headless Browser as a Service."

* **Architecture:** It allows you to execute Puppeteer scripts in the cloud. You send a URL and a set of instructions, and it returns the HTML or data.
* **Pros:** It is incredibly flexible. It can be used not just for Google Trends, but also to scrape Yahoo! Japan, news articles, and other sources that don't have APIs. This makes it a vital fallback or secondary tool.5
* **Cons:** Google Trends is a "hard" target. Scraping it with ScrapingBee often requires enabling "Premium Proxies" and "Stealth Mode," which increases the credit cost per request significantly (often 10-25 credits per single page load). This makes cost estimation difficult and potentially higher than SerpApi for high-volume structured data.5

### 3.4. Deep Dive: Bright Data & Oxylabs

These providers are the infrastructure giants, focusing on the underlying proxy networks.

* **Reliability:** Bright Data and Oxylabs own the world's largest residential IP networks. Their "SERP APIs" are built on top of this massive infrastructure, offering virtually 100% success rates because they can rotate through millions of Japanese residential IPs.15
* **Complexity:** They are often geared towards enterprise clients with complex compliance and integration needs. The setup can be more involved than SerpApi's "drop-in" replacement. However, for a "set it and forget it" reliable pipeline, they are the gold standard.
* **Pricing:** Pricing is often usage-based (CPM - Cost Per Mille). For 1,000 requests/day, the costs are competitive, but the lack of a simple "flat rate" for API calls can lead to variable monthly bills.18

### 3.5. Recommended Selection

**Primary Provider: SerpApi.**

The combination of a dedicated Google Trends endpoint, predictable flat-rate pricing, and free caching makes it the optimal choice for a startup-scale engineering team. The operational simplicity of receiving pre-parsed JSON outweighs the raw power of the larger proxy networks.

**Secondary/Fallback Provider: ScrapingBee.**

ScrapingBee should be retained for two purposes:

1. **Redundancy:** If SerpApi experiences an outage.
2. **Breadth:** To scrape Yahoo! Japan Realtime Search and the full text of news articles (for the content generation phase), tasks for which SerpApi is not designed.

## 4. Algorithmic Signal Detection

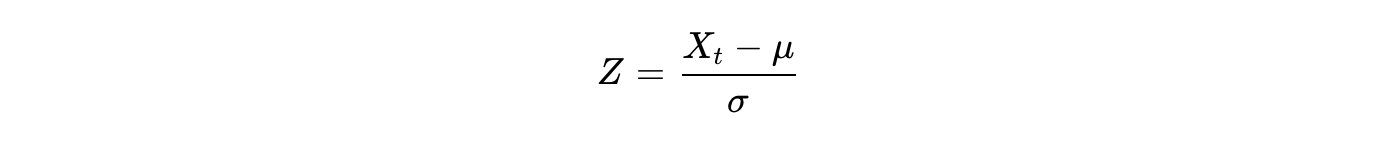
Collecting raw data is only the first step. The core value of Ovi English School lies in its ability to filter signal from noise—identifying which "trending" topics are relevant, rising, and sustainable versus those that are fleeting viral spikes or irrelevant noise.

### 4.1. Theoretical Framework for Trend Detection

Social and search data does not follow a normal (Gaussian) distribution; it typically follows a **Power Law** or **Poisson** distribution. This implies that "average" behavior is very low, and "trending" behavior is exponentially higher. Standard statistical tools must be adapted to handle this "fat-tailed" reality.

#### 4.1.1. Z-Score (Standardized Anomaly Detection)

The Z-score is a measure of how many standard deviations a data point is from the mean. It is the primary tool for detecting "shocks" in the system.



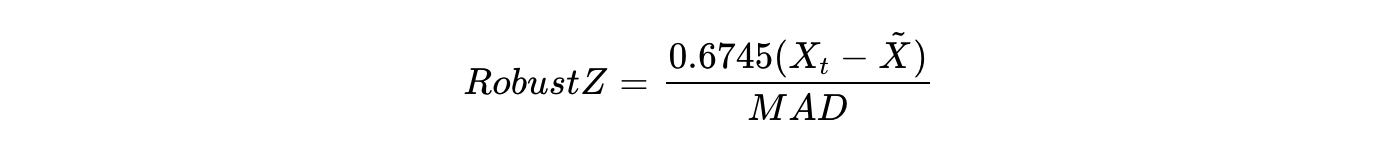
Where:

*  is the search interest at time .
*  is the rolling mean (e.g., over the last 7 or 14 days).
*  is the rolling standard deviation.

**Interpretation:**

* ****: Significant outlier (95th percentile).
* : Extreme anomaly (99.7th percentile) - Likely a viral spike.

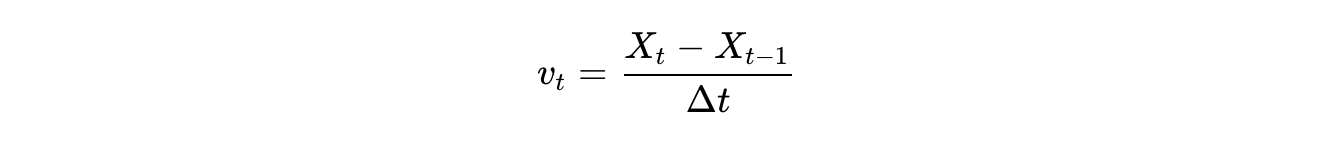
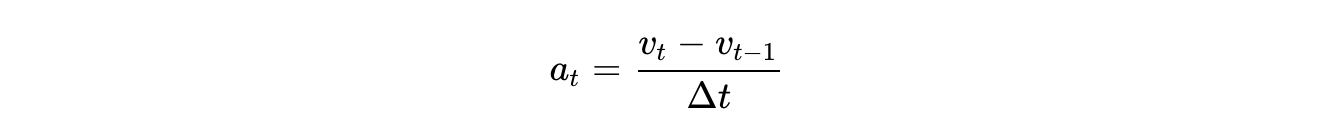
**The Robust Z-Score (Modified Z):** Because search data often contains outliers that skew the mean and standard deviation, a standard Z-score can be unreliable. A **Robust Z-Score** uses the Median and the Median Absolute Deviation (MAD), which are resistant to outliers.20



Where  is the median. For Ovi English School, implementing the Robust Z-score is recommended to prevent one massive viral day from skewing the baseline for the subsequent week.

#### 4.1.2. Kinematics of Trends: Velocity and Acceleration

To detect a trend *before* it peaks, we must treat search volume as a physical object in motion.

* **Velocity ():** The speed at which interest is growing.  
    
  High positive velocity indicates a rising topic.
* **Acceleration ():** The rate of change of velocity.  
  

**Strategic Signal Detection:**

* **The "Early Riser" Signal:**  and . The trend is growing, and the *rate* of growth is increasing. This is the optimal time to create content.
* **The "Peaking" Signal:**  and . The trend is still growing, but momentum is slowing. The peak is imminent.
* **The "Crashing" Signal:** . Interest is declining.

#### 4.1.3. Statistical Significance: The Mann-Kendall Test

To distinguish between a "viral spike" (one-day wonder) and a "sustained trend" (useful for educational content), we employ the **Mann-Kendall Test**. This is a non-parametric test used to statistically assess if there is a monotonic upward or downward trend in the variable over time.22

* **Application:** Run the Mann-Kendall test on the last 14 days of data.
* **Result:** If the -value  and the trend is positive, it confirms a statistically significant sustained trend. This is perfect for selecting topics for "Deep Dive" episodes rather than "Breaking News."

### 4.2. Python Implementation Module

The following Python module is designed to run on the M5 architecture. It utilizes pandas for vectorized calculations and scipy for statistical functions. It is structured to be callable from an n8n workflow.

Python

import sys  
import json  
import pandas as pd  
import numpy as np  
from scipy.stats import norm  
import pymannkendall as mk # Requires: pip install pymannkendall  
  
class TrendAnalyzer:  
 def \_\_init\_\_(self, history\_data):  
 """  
 Initialize with a list of dictionaries: [{'date': '2026-01-01', 'value': 50},...]  
 """  
 self.df = pd.DataFrame(history\_data)  
 self.df['date'] = pd.to\_datetime(self.df['date'])  
 self.df = self.df.sort\_values('date').reset\_index(drop=True)  
 self.df['value'] = pd.to\_numeric(self.df['value'])  
  
 def calculate\_kinematics(self):  
 """Calculate Velocity and Acceleration"""  
 # First derivative (Velocity)  
 self.df['velocity'] = self.df['value'].diff()  
   
 # Second derivative (Acceleration)  
 self.df['acceleration'] = self.df['velocity'].diff()  
   
 return self.df.iloc[-1][['velocity', 'acceleration']].to\_dict()  
  
 def calculate\_robust\_z\_score(self, window=7):  
 """Calculate Modified Z-Score based on Median Absolute Deviation"""  
 roll = self.df['value'].rolling(window=window)  
 median = roll.median()  
 # MAD = median(|x - median|)  
 mad = roll.apply(lambda x: np.median(np.abs(x - np.median(x))), raw=True)  
   
 # Avoid division by zero  
 mad = mad.replace(0, 1e-6)  
   
 self.df['robust\_z'] = 0.6745 \* (self.df['value'] - median) / mad  
 return self.df.iloc[-1]['robust\_z']  
  
 def check\_sustained\_trend(self, window=14):  
 """Run Mann-Kendall Test for monotonic trend"""  
 if len(self.df) < window:  
 return {"trend": "insufficient\_data", "p\_value": 1.0}  
   
 recent\_data = self.df['value'].tail(window).values  
 result = mk.original\_test(recent\_data)  
   
 return {  
 "trend": result.trend, # 'increasing', 'decreasing', 'no trend'  
 "p\_value": result.p,  
 "slope": result.slope  
 }  
  
 def analyze(self):  
 """Main analysis pipeline"""  
 if self.df.empty:  
 return None  
  
 kinematics = self.calculate\_kinematics()  
 z\_score = self.calculate\_robust\_z\_score()  
 mk\_test = self.check\_sustained\_trend()  
   
 current\_value = self.df.iloc[-1]['value']  
   
 # Decision Logic  
 status = "STAGNANT"  
 if z\_score > 2.0 and kinematics['acceleration'] > 0:  
 status = "VIRAL\_BREAKOUT"  
 elif mk\_test['trend'] == 'increasing' and mk\_test['p\_value'] < 0.05:  
 status = "SUSTAINED\_GROWTH"  
 elif kinematics['velocity'] < 0:  
 status = "DECLINING"  
   
 return {  
 "current\_interest": float(current\_value),  
 "velocity": float(kinematics['velocity']),  
 "acceleration": float(kinematics['acceleration']),  
 "robust\_z\_score": float(z\_score),  
 "mann\_kendall": mk\_test,  
 "classification": status  
 }  
  
# Example usage for n8n integration  
if \_\_name\_\_ == "\_\_main\_\_":  
 # n8n passes JSON string as first argument  
 try:  
 input\_json = sys.argv  
 data = json.loads(input\_json)  
 analyzer = TrendAnalyzer(data['history'])  
 result = analyzer.analyze()  
 print(json.dumps(result))  
 except Exception as e:  
 print(json.dumps({"error": str(e)}))

## 5. Alternative Trend Sources and Cross-Validation

Relying on a single data source is a single point of failure. A robust system triangulates trends from multiple platforms to confirm their validity.

### 5.1. Yahoo! Japan: The Cultural Hub

Unlike in the West, Yahoo! Japan remains a dominant web portal and is culturally distinct from Google.

* **Realtime Search:** Yahoo! Japan's "Realtime Search" (searches combined with Twitter data) is often faster than Google Trends for breaking news and pop culture. It is the definitive source for "what is happening right now" in Japan.25
* **Access Method:** There is no cheap, public API for Yahoo! Japan's Realtime ranking. The strategy here relies on **ScrapingBee**. The URL https://search.yahoo.co.jp/realtime contains a ranked list of the top 20 trending terms.
* **Integration:** Scrape this list every hour. If a term appears in both Yahoo! Realtime (Top 10) and Google Trends (Rising), it is a high-confidence candidate for Ovi English School.

### 5.2. Twitter / X: The Cost Barrier

Twitter (X) is massively popular in Japan. However, the API pricing model in 2026 is prohibitive for small-scale automated monitoring.

* **Pricing:** The "Basic" tier ($100/mo) offers only 10,000 reads per month. The "Pro" tier ($5,000/mo) is required for full stream access.27
* **Strategy:** Use Yahoo! Japan Realtime Search as a proxy for Twitter data. Since Yahoo! Japan natively indexes Japanese tweets for its realtime ranking, scraping Yahoo effectively gives you Twitter trend insights without paying the X API fees.

### 5.3. YouTube Data API v3

Japan has a thriving YouTube ecosystem. Trends here are often different—focused on entertainment, VTubers, and hobbies—providing a great source for "lifestyle" English content.

* **Access:** The API is free (quota-based). The default quota of 10,000 units/day is sufficient for checking trends 4-6 times per day.29
* **Endpoint:** videos.list with chart='mostPopular' and regionCode='JP'.
* **Cost:** A list request costs 1 quota unit. Fetching the top 50 videos every 4 hours consumes only ~6 units/day, leaving ample quota for deep dives.

### 5.4. News Sources: NewsData.io and GNews

Once a keyword is identified, you need the actual content (articles) to feed the podcast generation script.

* **NewsData.io:** Offers excellent coverage of Japanese local sources (Asahi Shimbun, Mainichi, etc.). It supports the country=jp parameter and provides sentiment analysis metadata.31
* **Pricing:** Free tier is limited; the "Standard" tier (~$100/mo) is recommended for production.
* **NewsAPI.org:** Historically popular, but often has gaps in Japanese local media coverage compared to NewsData.io.

## 6. Japan-Specific Considerations

Engineering for the Japanese market requires handling unique linguistic and temporal challenges.

### 6.1. The "Three-Script" Keyword Strategy

A single trend can be represented in three different ways in Japan:

1. **Kanji (漢字):** Formal, often used in newspapers (e.g., 人工知能 for AI).
2. **Katakana (カタカナ):** Used for loan words and trending tech/pop-culture terms (e.g., AIツール or チャットGPT).
3. **Hiragana (ひらがな):** Used for grammatical particles or softer, native terms.
4. **Romaji (Alphabet):** Occasionally used for brand names (e.g., iPhone).

**Impact on Trend Detection:** A trend might be "spiking" in Katakana but "flat" in Kanji. If you only track one script, you miss the volume.

* **Strategy:** Implement a **Keyword Expansion Step** in Python. When a candidate trend is found (e.g., "Election"), use a translation library or simple mapping to generate its variants. Query SerpApi for the **aggregate volume** of all variants combined, or select the variant with the highest Velocity.

### 6.2. Temporal Dynamics and Time Zones

* **JST (Japan Standard Time):** UTC+9.
* **Commute Windows:** The highest value engagement windows for podcasts in Japan are the morning commute (7:00 AM - 9:00 AM JST) and the evening return (6:00 PM - 8:00 PM JST).
* **Engineering Consequence:** The n8n automation must run at **4:00 AM JST** (19:00 UTC previous day). This allows 3 hours for data collection, script generation, audio synthesis, and publishing before the user steps onto the train at 7:00 AM.
* **Day Boundaries:** Google Trends daily data often aggregates based on PST (Pacific Standard Time). You must request timezone=-540 (min) to ensure the data aligns with the Japanese day, preventing "yesterday's news" from bleeding into "today's trend."

## 7. Technical Implementation: The M5/n8n Architecture

This section details the concrete architecture for the user's Apple MacBook Pro M5 stack.

### 7.1. Technology Stack Selection

* **Hardware:** Apple MacBook Pro M5 (ARM64). The Neural Engine and high-bandwidth memory are excellent for running local NLP models (if needed for summarization) and rapid vector calculations.
* **Orchestration:** **n8n** (Self-hosted). n8n is superior to Zapier for this use case because it allows the execution of local Python scripts and has no per-step pricing for complex logic loops.
* **Language:** Python 3.12+. Use uv or poetry for dependency management to ensure reproducible environments.
* **Database:** **SQLite** (Local). A lightweight SQL database is necessary to store the *history* of trends. To calculate a 7-day Z-score, you need 7 days of data stored locally; you cannot rely on fetching 7 days of history from the API every single time (it is slow and expensive).

### 7.2. Detailed n8n Workflow Design

The workflow is a Directed Acyclic Graph (DAG) triggered daily.

**1. Trigger Node:**

* **Schedule:** Cron 0 4 \* \* \* (Runs at 4:00 AM JST).

**2. Data Ingestion (Parallel Execution):**

* **Branch A (SerpApi):** HTTP Request to https://serpapi.com/search.
  + Params: engine=google\_trends\_trending\_now, geo=JP.
  + Returns: List of top 20 rising queries.
* **Branch B (ScrapingBee):** HTTP Request to https://search.yahoo.co.jp/realtime via ScrapingBee proxy.
  + Returns: HTML of ranking list.
* **Branch C (YouTube):** HTTP Request to YouTube Data API.
  + Params: chart=mostPopular, regionCode=JP.

**3. Data Normalization & Filtering (Python Node):**

* **Input:** Merges JSON outputs from A, B, and C.
* **Process:**
  + Extract keywords.
  + **Filter Blocklist:** Remove "Earthquake" (地震), "Tsunami," "Weather," and specific "Sports" terms if the podcast is not about sports.
  + **Deduplicate:** Merge "iPhone 17" and "iPhone release" into a single topic.

**4. Scoring & Selection (Python Script via Execute Command):**

* **Input:** List of top 10 candidate topics.
* **Action:** For each candidate, check the local SQLite DB for historical data.
  + *If missing:* Call SerpApi (engine=google\_trends, q=keyword, date=today 1-m) to get 30-day history. Save to DB.
  + *If present:* Update DB with latest data point.
* **Algorithm:** Run TrendAnalyzer (from Section 4.2). Calculate Robust Z-Score, Velocity, Acceleration.
* **Ranking:** Rank candidates by a weighted score: Score = (Z\_Score \* 0.4) + (Velocity \* 0.4) + (Yahoo\_Presence \* 0.2).

**5. Content Retrieval (NewsData.io):**

* **Input:** Winner Topic.
* **Action:** Fetch top 5 articles for the topic from NewsData.io.
* **Output:** Article text and metadata passed to the Script Generation Module (LLM).

**6. Fallback Chain:**

* If **SerpApi** fails (Error 5xx):
  + Wait 60 seconds.
  + Retry.
  + If fail: Switch to **ScrapingBee** to scrape Google Trends UI directly.
* If **Yahoo** scrape fails:
  + Ignore and proceed with Google-only data (Soft failure).

### 7.3. System Resilience

* **Rate Limiting:** Implement a "Sleep" node in n8n between SerpApi history calls to avoid hitting the 20 QPS (Queries Per Second) limit of the plan.
* **Error Logging:** Use n8n's "Error Workflow" trigger to send a notification (Slack/Email) if the pipeline fails, ensuring the user can manually intervene before the morning deadline.

## 8. Conclusion and Strategic Recommendation

The construction of Ovi English School's automated trend detection engine in 2026 demands a shift from "hacker" tactics to robust engineering. The reliance on pytrends is no longer viable. The optimal path involves a modest investment in **SerpApi** to handle the heavy lifting of Google Trends interaction, supplemented by **ScrapingBee** for broader Japanese market signals like Yahoo! Japan.

By implementing the **Robust Z-Score** and **Kinematic (Velocity/Acceleration)** algorithms within a local Python environment orchestrated by **n8n**, the system can reliably differentiate between ephemeral noise and the substantial, rising topics that drive engagement. This architecture, specifically tuned for the M5 stack and the nuances of the Japanese time zone and script system, provides a sustainable competitive advantage: the ability to "see" the news before it becomes common knowledge.

### Summary Checklist for Deployment

1. **Procure API Keys:** SerpApi (Production), NewsData.io (Standard), YouTube Data API (Free).
2. **Setup Environment:** Install Docker/n8n on M5 Mac. Create Python virtual environment with pandas, scipy, pymannkendall.
3. **Initialize DB:** Set up a simple SQLite schema for trend\_history.
4. **Deploy Workflow:** Import the logic flow described in Section 7.2.
5. **Test:** Run the pipeline manually at 10:00 AM, 4:00 PM, and 10:00 PM to calibrate the Z-score thresholds for the specific behavior of Japanese traffic.

#### Works cited

1. Google Trends' API isn't Public – Use This Instead - Glimpse, accessed February 1, 2026, <https://meetglimpse.com/google-trends-api/>
2. Google Releases Alpha Version of Trends API for Search Data, accessed February 1, 2026, <https://ecweb.ecer.com/topic/cn/detail-711795-google_releases_alpha_version_of_trends_api_for_search_data.html>
3. Introducing the Google Trends API (alpha): a new way to access Search Trends data, accessed February 1, 2026, <https://developers.google.com/search/blog/2025/07/trends-api>
4. The Holy Grail of Search Data Arrives in Budapest: Why the Google Trends API Session at Superweek 2026 is Unmissable - Adil Raseed | A Digital Marketer, accessed February 1, 2026, <https://adilraseed.com/superweek-google-trends-api/>
5. Best Google Trends Scraping APIs for 2025 | ScrapingBee, accessed February 1, 2026, <https://www.scrapingbee.com/blog/best-google-trends-api/>
6. Stepping out as a maintainer · Issue #636 · GeneralMills/pytrends - GitHub, accessed February 1, 2026, <https://github.com/GeneralMills/pytrends/issues/636>
7. Issues · GeneralMills/pytrends - GitHub, accessed February 1, 2026, <https://github.com/GeneralMills/pytrends/issues>
8. It broke again · Issue #602 · GeneralMills/pytrends - GitHub, accessed February 1, 2026, <https://github.com/GeneralMills/pytrends/issues/602>
9. Issue with pytrends and latest update of Google Trends website - Stack Overflow, accessed February 1, 2026, <https://stackoverflow.com/questions/75711314/issue-with-pytrends-and-latest-update-of-google-trends-website>
10. Google Trends API - SerpApi, accessed February 1, 2026, <https://serpapi.com/google-trends-api>
11. Google Trends Trending Now API - SerpApi, accessed February 1, 2026, <https://serpapi.com/google-trends-trending-now>
12. Yahoo! JAPAN Ads Search Ads API Reference v18, accessed February 1, 2026, <https://ads-developers.yahoo.co.jp/reference/ads-search-api?lang=en>
13. Google Trends Scraper - Free Credits, Simple Signup - ScrapingBee, accessed February 1, 2026, <https://www.scrapingbee.com/scrapers/google-trends-scraper/>
14. Pricing - ScrapingBee Web Scraping API, accessed February 1, 2026, <https://www.scrapingbee.com/pricing/>
15. How to Scrape Google Trends with Python - Bright Data, accessed February 1, 2026, <https://brightdata.com/blog/web-data/how-to-scrape-google-trends>
16. SERP API - SERP Scraper API - Free Trial - Bright Data, accessed February 1, 2026, <https://brightdata.com/products/serp-api>
17. Scrape Google Trends Data With Web Scraper API - Oxylabs, accessed February 1, 2026, <https://oxylabs.io/products/scraper-api/serp/google/trends>
18. SERP API Pricing Plans - Bright Data, accessed February 1, 2026, <https://brightdata.com/pricing/serp>
19. Google Search Scraper API - Free Trial - Oxylabs, accessed February 1, 2026, <https://oxylabs.io/products/scraper-api/serp/google>
20. Practical Guide for Anomaly Detection in Time Series with Python - Data Science With Marco, accessed February 1, 2026, <https://www.datasciencewithmarco.com/blog/practical-guide-for-anomaly-detection-in-time-series-with-python>
21. Modified Z-Score Explained (Python Outlier Detection) - YouTube, accessed February 1, 2026, <https://www.youtube.com/watch?v=wCXQo6OYbhg>
22. Performing a Mann-Kendall Trend Test for Temperature Data in Python | Code for Nepal, accessed February 1, 2026, <https://codefornepal.org/2022/08/29/mann-kendall>
23. How to Perform a Mann-Kendall Trend Test in Python - GeeksforGeeks, accessed February 1, 2026, <https://www.geeksforgeeks.org/machine-learning/how-to-perform-a-mann-kendall-trend-test-in-python/>
24. Mann-Kendall Trend Test Using Python - Analytics Vidhya, accessed February 1, 2026, <https://www.analyticsvidhya.com/blog/2023/07/mann-kendall-trend-test-using-python/>
25. Yahoo! JAPAN Ads API v19 release, accessed February 1, 2026, <https://ads-developers.yahoo.co.jp/en/ads-api/announcement/26012703.html>
26. How to Scrape Yahoo: Step-by-Step Tutorial | ScrapingBee, accessed February 1, 2026, <https://www.scrapingbee.com/blog/how-to-scrape-yahoo/>
27. How to Get X API Key: Complete 2026 Guide to Pricing, Setup & Optimization - Elfsight, accessed February 1, 2026, <https://elfsight.com/blog/how-to-get-x-twitter-api-key-in-2026/>
28. Twitter/X Data Access Options Compared (2026) | Xpoz Blog, accessed February 1, 2026, <https://www.xpoz.ai/blog/comparisons/twitter-x-data-access-options-compared-2026/>
29. Quota and Compliance Audits | YouTube Data API - Google for Developers, accessed February 1, 2026, <https://developers.google.com/youtube/v3/guides/quota_and_compliance_audits>
30. Quota Calculator | YouTube Data API - Google for Developers, accessed February 1, 2026, <https://developers.google.com/youtube/v3/determine_quota_cost>
31. The Only News API Comparison You Need In 2026, accessed February 1, 2026, <https://newsdata.io/blog/news-api-comparison/>
32. Japan News API - Latest News Headlines & Live Updates - NewsData.io, accessed February 1, 2026, <https://newsdata.io/news-sources/japan-food-news-api>