Information Services

**Core Objective:** Set up an AI-backed news portal network that is narrative intelligent, self-learning, self-sustaining, scalable, multilingual, and fully automated. We are building a **strategic media machine** with intelligence, memory, editorial strength, and counter-narrative capabilities.

# Phase-Wise Goals

## 

## Phase 1: Learning Context

* The Model requires to start learning from self before feeding RSS feeds.
* The system needs to be able to understand whether an article is relevant, if it has narrative potential & if it is a threat, opportunity or neutral for the client
* The system should generate semantic embeddings (AI memory) from the uploaded articles
* The system needs to be able to classify any article based on demographic segmentation and analyse its story as well as sentiment.
* The system should classify and cluster the input articles into multiple folders and tag narrative docs to articles.
* The system should auto-load at a predefined time every day to learn the articles uploaded in the last 24 hours. (Which can be smaller durations as well)

| **Input** | **Purpose** |
| --- | --- |
| News Article (Headline + Body in .txt format) | Raw content to analyze |
| List of Broad Demographic Segments | Predefined voter groups for classification |
| List of Sentiments to Activate | Report Leaning towards (Left, Right, Neutral) |
| Definition of Relevance | A simple guideline or logic for what is "politically relevant" to your client |
| Examples of Threat / Opportunity / Neutral | Examples to help the AI recognize the polarity of articles |
| Feed Narrative Documents | Train the model to do the sentiment analysis based on the narrative docs |
| Add Name Entity Recognition | Tag People, Places, Political Parties, Policies, in the articles while feeding |
| Add Metadata details | Add timestamp, source, language, source bias to each article |

*\*All the input articles and narratives can be in .txt or .md file format. Direct links can be pasted from a few news portals that follow WebP standard.*

**Phase 2: Learning the Reasoning**

* The system needs to be able to pick whether it should move forward on the story based on segmentation & core context
* The system should cluster the articles and reason whether it fits the narrative to write or not.
* Define Opportunity and Threat for the system. If the story is political and relates to INC and is tagged ‘opportunity’ → learn more about it from narrative documents.”
* Add a YAML or Json config to define reasoning rules of a Threat, Opportunity or Neutral and define Action for each category.

**Phase 3: Learning the Tone**

* The system should start writing based on the tone of sample articles and human-like language,
* Understand the sentence style, structure, tone, and keywords from each input
* The system should perfect the many narrative spins on articles.
* The system should be able to create multiple results for each input article/ prompt if required. One standard tone, one aggressive, one neutral to see which one works better and which one needs more work.
* Adding tone templates and adding more to it from the output file, marking which tones are preferable. Further segmentation can be added (e.g., “questioning tone”, “hopeful tone”, “exposing tone”)
* The system should show output article options to choose from (Which later will be automated as we select specific formats)
* The system should learn from the inputs of our preferred articles + from biased competitors
* The system should mimic - analyse - counter with preferred tone & patterns
* The system will evolve into a thinking editorial agent which will write intelligent counter-articles (attack, expose, reframe)

**Phase 4: Setting Target Audience/ Rich Category based storage (Enhanced Memory)**

* The system should learn various target groups and tag the output articles according to the target audience. Adding more demographic and Psychographic tags like Urban middle class, rural youth, first-time voters, religious conservatives, etc.
* The system should classify each output into certain narratives it has already learned
* The system should capture narrative impact, which stories resonated with which segments and why.
* Add human-like vocabulary and spin.
* Add goals to each article to be produced, which will be picked by the system as per the RSS feed and target website later. Influence opinion / generate doubt / promote alignment / set agenda.

**Phase 5: News Article Automation**

* The system should connect with an RSS feed and read inputs and cluster them accordingly.
* The system should pick the clusters as per the narrative prompts of the day or the total narrative docs fed regularly
* The system should pick the narrative tone (Negative, Positive, Neutral) to rewrite articles
* The system should pick context-worthy keywords and include them in the article
* Writing SEO rich articles and adding relevant images from a predefined gallery, Wikimedia Commons, or scraping images from the source article. (This section needs to be connected to an instance that can handle image optimisation)
* Connecting to WordPress - identifying relevant categories - relevant tags - uploading articles in approved format.
* Updating the links of articles with context in a Google sheet for some manual checkups on quality.

\*an optional **human-in-the-loop checkpoint** for early versions before full automation

\*Create publish rules based on the target website on the basis of language, tone and demography.   
This will be used for automation funnels for each website as a template so system can auto select for future websites

**Phase 6: Translation + Regional Tuning**

* The system should become multilingual using regional language models from IndicTrans2, M2M100 or similar models.
* The system should be able to manage being connected to multiple websites with target languages
* In this phase, we will also start feeding narrative docs and sample articles in local languages.
* **tone loss in translation** → may need localized tone templates in future.

**Phase 7: Self-Improving Narrative fine-tuning**

* The system should start populating article metrics to the output and learn which type of articles work
* Adapting to prompts that fit on an ad-hoc basis.
* The system should start generating and testing narratives on its own (based on new articles from rss feeds or prompts).
* narrative drift detection - Weekly manual input for narrative articles to keep the system working with feedback.
* To be connected with Google Analytics and Social media APIs to understand the impact of articles.

**The system will work on the following infrastructure:**

1. LLM engine (Llama 2.0 on local at first - Use Mistral in production server)
2. A streamlit/FastAPI dashboard for the admin
3. Python 3.13 as interpreter
4. Regional language support through IndicTrans2 or M2M100
5. Vector Database feeding the brain of the Model
6. Pandas for file reading, logging and later storing output logs in a vector Database. With larger volume easy transition to SQLite.
7. In future, with more volume, we can consider Haystack Pipelines for better automation
8. **Docker Compose** for local - VPS transition (easier scaling later)

**We can set category rules to keep the model in check.:**

| **Category** | **Behaviour** |
| --- | --- |
| Politics | Apply a strong narrative, take a position  Fact-checking if required (Model will pick up the claims/ prompt based) |
| Culture | Align with values, promote selectively |
| Apolitical | Reframe gently, write neutral/filler tone |
| Breaking or Prompt-based | Push fast, but within narrative bounds |

Content Flow:

RSS → Article (raw) → Narrative Vector Match (auto-selected context, no prompt) → LLM (LLaMA/Mistral) writes article → Image + SEO + Tags → WordPress (Published) → Weekly Feedback (Good/Bad + Comments) → Tone Learner Memory Updated

Metrics Flow:

Published Article → Google Analytics → Metrics Loading in DB → YAML/JSON config to read the Analytics → Generate weekly report on views/reaction