🌐 Topical Focus Analyzer - User Guide

💡 Understanding Site Focus Score & Site Radius Score

Before diving into the tool, let's clarify the key metrics it calculates:

- **Site Focus Score (0-100):** This score measures how thematically similar the URLs (representing pages) are to each other across the site. **A higher score** indicates the website's content is tightly clustered around specific topics, showing strong thematic cohesion and specialization. A lower score suggests broader or more scattered topics.
- **Site Radius Score (0-100):** This score reflects how tightly the page representations cluster around a central theme for the site (the calculated "centroid" or site embedding concept). **A higher score** means the content is closely grouped (a smaller "radius"), indicating that individual pages don't deviate much from the overall site theme. A lower score signifies that content themes are more spread out or diverse relative to the site's center.

Why is this important? Analyzing these scores helps you:

- **Improve SEO:** Search engines favor websites with clear topical authority. High Site Focus and Radius scores can signal expertise and relevance in specific areas.
- **Refine Content Strategy:** Understand if your content aligns with your intended niche or if it's becoming too broad. Identify gaps or areas needing more depth to reinforce the site's core themes.
- **Enhance User Experience:** A well-focused site is often easier for users to navigate and understand its purpose.
- **Competitor Analysis:** Compare your site's focus and thematic consistency against competitors.

1. 🚀 Introduction

Welcome to the Topical Focus Analyzer! This tool helps you understand a website's structure and content themes by analyzing its sitemaps and, optionally, the content of its pages. It provides insights into how focused the site's topics are (using the **Site Focus Score** and **Site Radius Score** explained above), visualizes content relationships, and identifies potential content duplication (cannibalization).

This guide covers setting up the application (based on the `multi_sitemap_app.py` version) and using its features.

2. 📋 Prerequisites

Before you begin, ensure you have the following:

- **Python:** Version 3.9 or higher installed. You can download it from python.org.
- **Google AI API Key (Optional):** Needed only if you want to use the AI-powered summary feature. Get one from Google AI Studio.

3. 🛠️ Installation & Setup

Follow these steps precisely in your terminal or command prompt.

1. **Create Project Directory:**

   Make a folder for the application and navigate into it:

   ```
   mkdir topical-focus-analyzer
   cd topical-focus-analyzer
   ```

2. **Create & Activate Virtual Environment:**

   This isolates the app's dependencies.

   Create:

   ```
   python -m venv venv
   ```

   Activate:

   On Windows (Command Prompt/PowerShell):

   ```
   venv\Scripts\activate
   ```

   On macOS/Linux (Bash/Zsh):

   ```
   source venv/bin/activate
   ```

   > You should see `(venv)` prefixed to your terminal prompt.

3. **Create `requirements.txt` File:**

   Create a file named exactly `requirements.txt` in the `topical-focus-analyzer` directory. Paste the following content into it:

   ```
   # Core Libraries
   requests
   beautifulsoup4
   lxml
   pandas
   numpy<=1.26.4
   scikit-learn<=1.6.2
   plotly
   streamlit
   python-dotenv

   # AI Summarization
   google-generativeai

   # Content Extraction (Simplified)
   trafilatura
   regex
   ```

   > Specific versions for `numpy` and `scikit-learn` are included for better compatibility as discussed during testing. `trafilatura` is recommended but optional; the app has a fallback if it fails to install.

4. **Install Dependencies:**

   Run this command while your virtual environment is active:

   ```
   pip install -r requirements.txt
   ```

5. **Create Project Files & Structure:**

   Create the necessary subdirectory and empty Python files. The actual code for these files should be obtained based on the development conversation (this guide focuses on setup).

   Create the `modules` subdirectory:

   ```
   mkdir modules
   ```

   (Use `md modules` on Windows Command Prompt if `mkdir` fails)

   Create the following empty files:

   On macOS/Linux:

   ```
   touch modules/__init__.py
   touch modules/sitemap_finder.py
   touch modules/sitemap_parser.py
   touch modules/content_extractor.py
   touch modules/simple_vectorizer.py
   touch modules/dimensionality_reducer.py
   touch modules/analyzer.py
   touch modules/llm_summarizer.py
   touch multi_sitemap_app.py
   touch .env
   ```

   On Windows (PowerShell):

   ```
   New-Item -ItemType File -Path "modules\__init__.py" -Force
   New-Item -ItemType File -Path "modules\sitemap_finder.py" -Force
   New-Item -ItemType File -Path "modules\sitemap_parser.py" -Force
   New-Item -ItemType File -Path "modules\content_extractor.py" -Force
   New-Item -ItemType File -Path "modules\simple_vectorizer.py" -Force
   New-Item -ItemType File -Path "modules\dimensionality_reducer.py" -Force
   New-Item -ItemType File -Path "modules\analyzer.py" -Force
   New-Item -ItemType File -Path "modules\llm_summarizer.py" -Force
   New-Item -ItemType File -Path "multi_sitemap_app.py" -Force
   New-Item -ItemType File -Path ".env" -Force
   ```

   > You will need to populate these files (especially `multi_sitemap_app.py` and the files inside `modules/`) with the Python code developed previously.

4. ⚙️ Configuration

Configure the optional AI Summary feature:

1. Open the `.env` file located in your `topical-focus-analyzer` directory.
2. Add your Google AI API Key like this (replace `your_google_api_key_here` with your actual key):

   ```
   GOOGLE_API_KEY=your_google_api_key_here
   ```

3. Save and close the file.

> If you don't add a key or the key is invalid, the "Generate AI Summary" option in the app will be disabled.

5. ▶️ Running the Application

1. **Ensure Virtual Environment is Active:** If you closed your terminal, navigate back to the project directory and reactivate it (see Step 3.2).
2. **Run the Streamlit App:** Execute this command:

   ```
   streamlit run multi_sitemap_app.py
   ```

3. **Access the App:** Your default web browser should open automatically to the application (usually `http://localhost:8501`). If not, manually navigate to that address in your browser.

6. 🖱️ Using the Application

The application interface is divided into a sidebar for configuration and a main area for results.

1. **Enter Domain:** In the sidebar, type the target domain (e.g., `streamlit.io`).
2. **Find Sitemaps:** Click the "Find Sitemaps" button.
3. **Select Sitemaps:** Check the boxes for the sitemap(s) you wish to analyze. You can use the "Select All Sitemaps" checkbox.
4. **Configure Filters (Optional):** Expand the "URL Include/Exclude Filters" section to add keywords that must (or must not) be present in the URLs you want to analyze. Choose the logic (AND/OR) for include filters.
5. **Set Analysis Options:**
   - Decide whether to **Analyze Page Content**. Enabling this is slower but much more accurate.
   - If analyzing content, choose the **Vectorization Mode** (Content Only, URL Path Only, or Combined). Adjust the **Content Weight** if using Combined mode.
   - Set the **Max URLs to Process** (note: this slider might be outside the "Advanced" section in some versions).
   - Adjust **Advanced Analysis Options** (TF-IDF, t-SNE, Metrics, Cannibalization) if needed. Defaults are generally reasonable, but see the explanation below for details on Perplexity, k1, and k2.

   > ### Understanding Advanced Analysis Options
   >
   > These settings, found in the "Advanced Analysis Options" expander, allow fine-tuning the analysis process. Adjusting them is optional and typically only needed if default results seem off or if you want to experiment.
   >
   > - #### t-SNE Perplexity
   >
   > **What it does:** This parameter influences the t-SNE algorithm used for the "Visual Map". Conceptually, it relates to the number of nearest neighbors considered for each point when creating the low-dimensional map.
   > **Impact:** Lower values emphasize local structure (potentially showing more small, tight clusters). Higher values focus more on the global structure (potentially merging smaller clusters or showing broader relationships).
   > **Guidance:** The default (**15**) is a reasonable starting point, especially for smaller to moderate datasets. For very small datasets (< 50 URLs), try values closer to **5**. For larger datasets (> 1000 URLs), you might experiment with values up to **50**. If the map looks like a single dense ball, perplexity might be too high. If it looks too fragmented with no clear groupings, it might be too low.
   >
   > - #### Site Focus Score Scaling (k1)
   >
   > **What it does:** This slider (default **5.0**) adjusts the sensitivity of the **Site Focus Score** calculation. It scales how the average similarity between URL vectors translates into the final 0-100 score.
   > **Impact:** A higher `k1` value makes the score increase more rapidly as average similarity goes up. This means the score becomes more sensitive, requiring higher internal similarity to achieve top scores. A lower `k1` value makes the score increase more gradually.
   > **Guidance:** Adjust this if you find scores across different sites are consistently too high (lower `k1`) or too low (increase `k1`) for your interpretation. The default of 5.0 provides moderate sensitivity. Experiment if needed to calibrate the score range.
   >
   > - #### Site Radius Score Scaling (k2)
   >
   > **What it does:** This slider (default **5.0**) adjusts the sensitivity of the **Site Radius Score** calculation, specifically how the maximum distance of any URL vector from the site's central theme (centroid) affects the score via the logarithmic formula.
   > **Impact:** A higher `k2` value makes the score decrease more rapidly as the maximum distance increases. This makes the score more sensitive to outliers or content spread (a tighter site gets a higher score more easily). A lower `k2` value makes the score decrease more gradually, meaning the site needs to be significantly more spread out for the score to drop substantially.
   > **Guidance:** Adjust this if the Radius Scores seem counter-intuitive. If seemingly focused sites get low scores (consider slightly increasing `k2` to make it more sensitive to small deviations), or if diverse sites get very high scores (consider slightly decreasing `k2` to make it less sensitive). The default of 5.0 offers balanced sensitivity.

6. **Configure AI Summary (Optional):** Enable the toggle if your API key is configured in `.env`.
7. **Process Sitemaps:** Click the "Process Selected Sitemaps" button to start the analysis. The app will show status updates.
8. **Explore Results:** Once processing is complete, explore the results using the tabs:
   - 📊 **Overview:** Key metrics (**Site Focus Score** & **Site Radius Score**), AI summary, page type/source distribution, key URLs.
   - 📄 **URL Details:** Searchable table of all processed URLs and their data.
   - 🗺️ **Visual Map:** Interactive t-SNE plot showing URL relationships.
   - 👯 **Cannibalization:** Table of potentially duplicate content pairs based on similarity.
   - 🔍 **Content Inspector:** View the extracted text used for analysis (if content analysis was enabled).
   - 📋 **Processing Log:** Detailed log messages from the analysis run.

7. 🧩 Understanding the Components (Main Files)

The application relies on several files working together:

`multi_sitemap_app.py`
The main application file you run. It creates the web interface (using Streamlit), handles user input, calls the different modules in sequence, and displays the final results.

`.env`
A simple text file (not Python) to securely store your Google API Key outside the main code.

`requirements.txt`
Lists all the external Python libraries the application needs to function.

`modules/` **(Directory)**
Contains the core logic, broken down for organization:

`sitemap_finder.py`
Finds potential sitemap URLs for a given domain (checking `robots.txt` and common paths).

`sitemap_parser.py`
Parses sitemap files (XML, TXT, GZipped, Index files) and extracts URLs, applying include/exclude filters.

`content_extractor.py`
(Simplified version) If content analysis is enabled, fetches HTML content from URLs and attempts to extract the main textual content using `trafilatura` (if available) or basic `beautifulsoup` heuristics. Cleans the extracted text.

`simple_vectorizer.py`
(Simplified version) Preprocesses URL paths and/or cleaned page content. Converts the text data into numerical TF-IDF vectors (page embeddings conceptually) using `scikit-learn`. Handles URL-only, Content-only, and Combined modes.

`dimensionality_reducer.py`
Reduces the high-dimensional vectors to 2 dimensions using t-SNE for visualization. Also calculates the geometric centroid (conceptual site embedding) of vectors and distances from it.

`analyzer.py`
Calculates high-level metrics like **Site Focus Score** and **Site Radius Score** (using updated log scale). Identifies potential content cannibalization by finding pairs of URLs with high vector similarity.

`llm_summarizer.py`
Formats analysis data into a prompt and uses the Google Gemini API (via `google-generativeai` library) to generate a natural language summary of the findings, if enabled and configured.