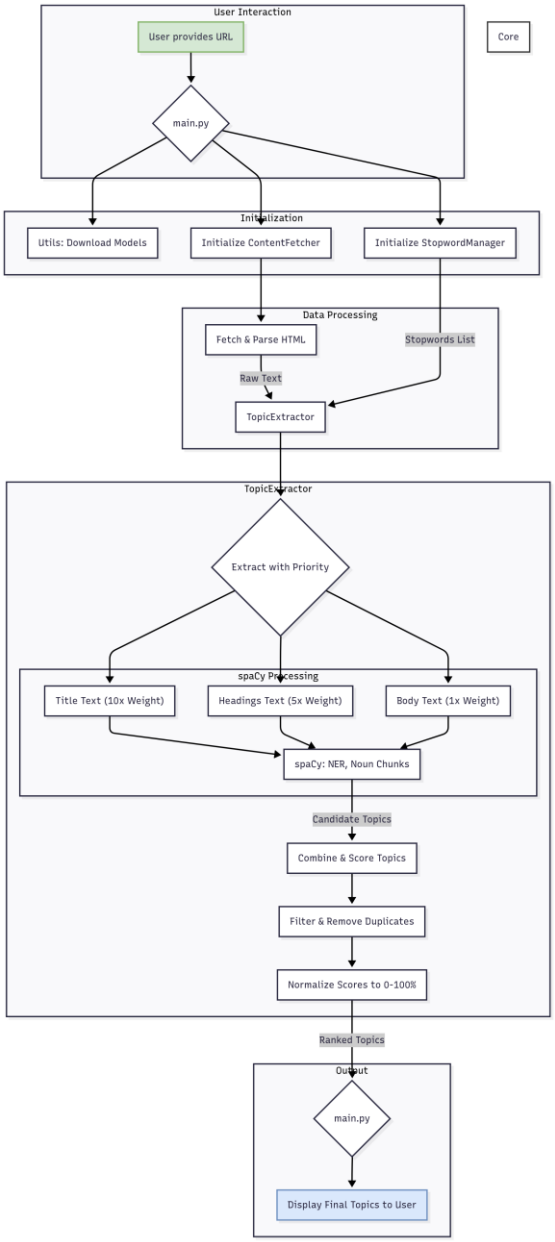
Design of the Current System

This section details the architecture and techniques used in the provided Python code.



A. Architecture Overview

The system is designed with a modular structure, separating responsibilities into distinct classes, each in its own file:

- **main.py**: The entry point of the application. It orchestrates the entire workflow, from initializing dependencies to running the analysis and printing the results.
- **utils.py**: A helper module responsible for downloading and loading the required NLTK and spaCy models, with error handling.

- **content_fetcher.py**: Handles all network interactions. It is responsible for fetching the HTML content from a URL and parsing it to extract relevant text sections.
- **stopword_manager.py**: Manages all aspects of stop word filtering. It creates a comprehensive set of words to ignore during analysis.
- **topic_extractor.py**: The core of the system. It contains all the NLP logic for identifying, scoring, and ranking topics from the text provided by the ContentFetcher.

B. Core Techniques and Logic

1. Stop-word Management (Stopword_Manager)

The system creates a robust filter list by combining three sources of stopwords:

- **NLTK Standard List:** It uses the default English stopword list from nltk.corpus.stopwords as a baseline (e.g., "a", "the", "in").
- **Custom Generic List:** A manually curated list of common web and action-oriented words (e.g., "click", "download", "website", "product") is added to prevent them from appearing as topics.
- **Dynamic URL Keywords:** The script parses the input URL and extracts the domain name (e.g., "amazon" from amazon.com). This ensures the website's own name is not listed as a key topic.

2. Priority-Based Weighting (Topic_Extractor)

A key feature of the system is its understanding that not all text is equally important. It assigns a **priority multiplier** to topics based on where they are found in the HTML document:

- **Title (<title>)**: Topics found here receive a **10.0x** score multiplier. The title is the strongest indicator of a page's content.
- **Headings (<h1>, <h2>, etc.)**: Topics from headings get a **5.0x** multiplier, as they signify important sections.
- **Body Text**: Topics in the main content serve as the baseline with a **1.0x** multiplier.

3. NLP with spaCy (Topic_Extractor)

The core topic identification is powered by spaCy's pre-trained en_core_web_sm model, which provides several layers of linguistic analysis:

- **Named Entity Recognition (NER):** This is used to identify proper nouns that fall into predefined categories like PRODUCT, ORG (Organization), and GPE

(Geo-Political Entity). These are treated as high-value topics. For example, in a product page, NER would identify "Cuisinart" as an ORG.

- **Part-of-Speech (POS) Tagging and Noun Chunking:** The system analyzes sentences to identify their grammatical structure. It specifically extracts "noun chunks"—phrases that represent a person, place, or thing (e.g., "compact 2-slice toaster"). These chunks are ideal topic candidates. It also identifies individual important words tagged as nouns (NOUN) or proper nouns (PROPN).

4. Scoring and Normalization

After all candidate topics are extracted and scored with their priority weight, they undergo a final processing stage:

1. **Duplicate Removal:** The list is sorted by score and length. The code then iterates through it, removing any topic that is a substring of a higher-scoring, longer topic. For example, if "Toaster" and "Compact 2-Slice Toaster" are both found, "Toaster" is removed as it's redundant.
2. **Score Normalization:** The raw scores are scaled to a user-friendly **0-100 confidence percentage**. This makes the final output easy to interpret, regardless of the document's length or initial scores.