Automating the Creation of 700 Blogs Monthly: AI-Driven Approach

Author: Gregorio Capelli

Satispay – Junior Data Scientist – Business Case

Contents

Introduction

In today's digital landscape, content marketing plays a critical role in the success of FinTech companies. However, producing high-quality, SEO-optimized blog content at scale presents significant challenges. This document outlines an automated solution designed to streamline the end-to-end process of blog content creation—from keyword extraction to content generation, SEO optimization, and publishing.

Objective

The goal of this project is to develop an automated workflow that:

- Retrieves relevant keywords for a topic connected to the FinTech ecosystem;
- Generates high-quality textual content using AI models;
- Optimizes the content for SEO;
- Manages internal linking between the newly created content and existing content;
- Automates the publication of content to a website CMS.

Workflow Overview

Key Steps

The proposed automation workflow consists of five key steps:

Keyword Extraction and Generation:

The starting point is either a pre-made set of lists of semantically linked keywords or a single generic topic about which we want to create various blogs.

If we have pre-made keywords, the program just skips to point 2 of this workflow. On the other hand, if we only have the topic, the program proceeds with the generation of the missing relevant keywords.

To do so, it follows two complementary routes:

(1) it scrapes the top links of a google search of the topic, retrieves the most common relevant words in those links, and uses them as context for an AI model to generate a series of lists of keywords that depend on current information and are currently relevant;
(2) it generates a series of lists of keywords from scratch using an AI model thus using the general knowledge of LLMs.

Finally, it combines the two series of lists into one big list of lists of the type:

$$final\_list = [\,[keyword11, keyword12, \dots], [keyword21, keyword22, \dots], \dots].$$

Where the sublists are formed by semantically close keywords and are going to be the basis for the generation of each different blog post.

As a note, to improve the speed of the web scraping process, we implemented parallel processing of the different scraped links.

Content Generation:

Then, via an API call to OpenAI the program generates a first draft of a blog post, a title, and a meta description based on each group of keywords. Such drafts are required to respect correct Heading

Structure, current SEO best practices and be engaging, relevant for a FinTech startup, and well written with call to actions, bullet lists and so on. The system message and prompt for AI when creating the body of the post can be found below:

```
system = (
            "You are an expert SEO copywriter specializing in FinTech content."
            "You excel at creating engaging, well-researched articles that rank highly in search engines."
            "Your writing style combines technical accuracy with clear explanations for both beginners and experts."
            "You follow modern SEO best practices, including proper heading hierarchy, optimal keyword placement, "
            "and reader-friendly formatting with short paragraphs and bullet points."
            f"Your task is to write a comprehensive blog post for a FinTech audience about the keywords in this list: '{keywords}'."
        )

prompt = (
            f"Write a comprehensive blog post for a FinTech audience about the keywords in this list: '{keywords}'. The post should include:\n"
            f"1. An engaging introduction that hooks the reader\n"
            f"2. 3-5 main sections with H2 headings\n"
            f"3. Relevant subsections with H3 headings where appropriate\n"
            f"4. A clear conclusion summarizing key points\n"
            f"5. A practical bullet list of actionable takeaways\n"
            f"6. A strong call-to-action\n\n"
            f"Maintain natural keywords density (1-2%), use transition words for flow, "
            f"and keep paragraphs under 3-4 sentences for readability. "
            f"Balance professional expertise with an approachable, conversational tone."
            f"The post must not include the title nor the meta description."
        )
```

To make this step faster, we implemented parallel processing of the requests when OpenAI APIs permits it.

SEO Optimization:

Afterwards, each of the generated content is checked with respect to various readability and SEO benchmarks and reports with the necessary corrections are created.

Such reports are then passed to an AI model together with the initial drafts of the content. Final and SEO optimized versions of the blog posts are thus generated. Again, to make this step faster, we implemented parallel processing of the requests when OpenAI APIs permits it.

In case it appears necessary, or we receive the suggestion from marketing, to perform a new SEO or content quality check it will be easy to add it to the workflow by just implementing it and making the program append the results to the report.

Internal Linking Management:

To further improve the new posts’ SEO, the program proceeds with a simple internal linking management process, that is to be considered as a starting point for further development. For each newly created post, it computes the cosine similarity with each of the already existing blog posts, checks if such metric is bigger than a threshold and, in this case, adds links to the old blog posts into the new one. It will be easy to also apply more advanced Machine Learning and NLP techniques if necessary.

Publishing Automation:

Finally, we implemented a simple process to automate the upload of our SEO optimized content into a Content Management System, particularly WordPress, but with a structure that is easily extendable to other CMS if needed. This step is also to be considered as a starting point for further development and improvement can be easily implemented if necessary.

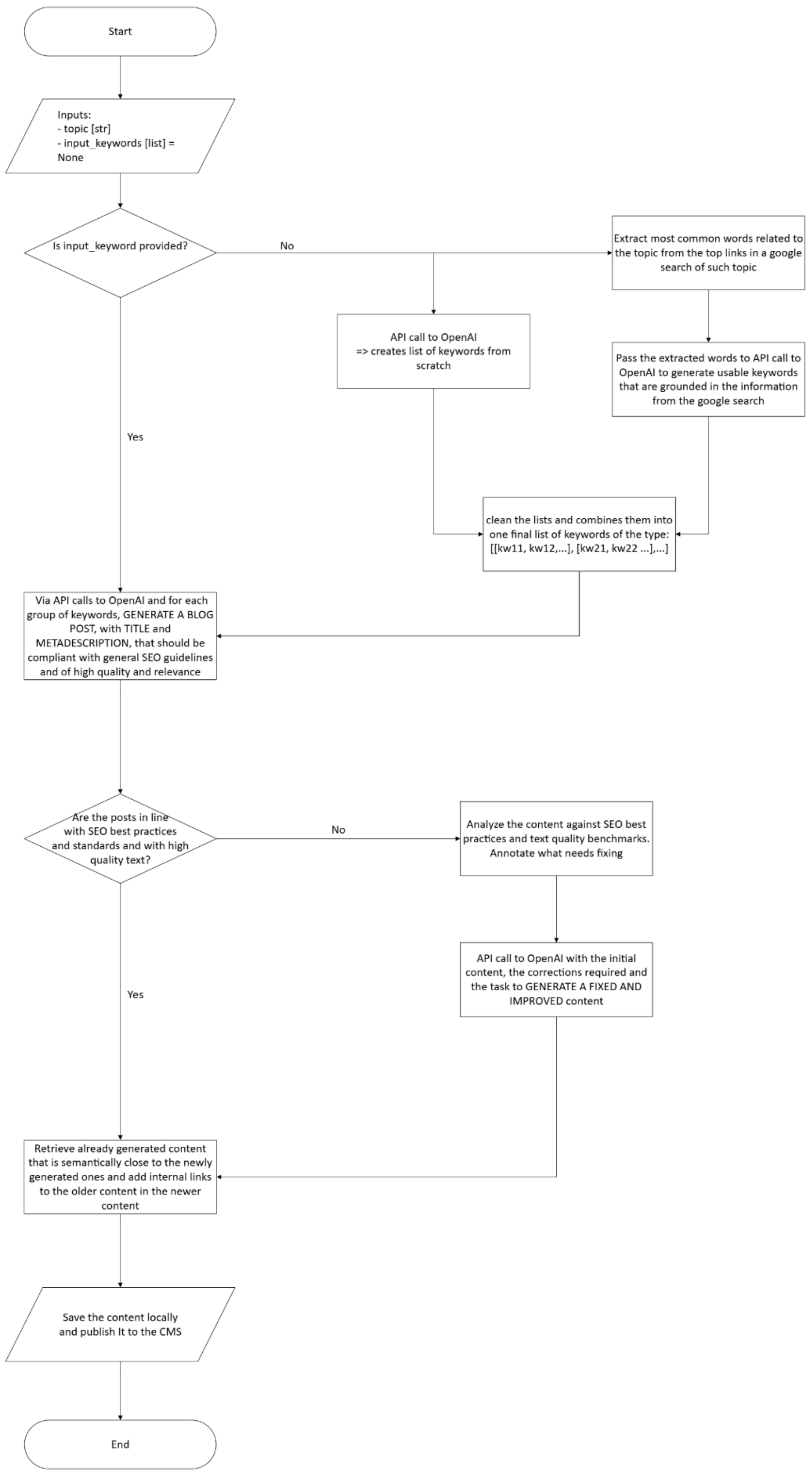
Technical Overview of Each Step

Steps	Descriptions	Tools/Techniques Used
1. Keyword Extraction and Generation	Identify high-value keywords related to FinTech topics using AI and web scraping.	• *OpenAI API;* • *Google Search Scraping:* ○ *BeautifulSoup;* ○ *NLTK;* • *Parallel Processing.*
2. Content Generation	Automatically generate blog posts, titles, and meta descriptions based on the keywords	• *OpenAI API;* • *Parallel Processing.*
3. SEO Optimization	Analyze and enhance content for SEO best practices.	• *Python:* ○ *TextBlob;* ○ *Textstat;* ○ *NLP models;* • *Parallel Processing.*
4. Internal Linking	Suggest internal links to improve site navigation and SEO.	• *TF-IDF;* • *Cosine Similarity (scikit-learn);* • *NLP techniques (future implementation).*
5. Publishing Automation	Upload content to a CMS (e.g., WordPress).	• *Requests API;* • *CMS REST APIs.*

Python Implementation

Simple Flowchart

The logic behind the Python implementation of the proposed workflow can be seen in the flowchart below:


Start
Inputs:
- topic [str]
- input_keywords [list] = None
Is input_keyword provided?
No
Yes
Extract most common words related to the topic from the top links in a google search of such topic
API call to OpenAI
=> creates list of keywords from scratch
Pass the extracted words to API call to OpenAI to generate usable keywords that are grounded in the information from the google search
clean the lists and combines them into one final list of keywords of the type: [[kw11, kw12,...], [kw21, kw22 ...],...]
Via API calls to OpenAI and for each group of keywords, GENERATE A BLOG POST, with TITLE and METADESCRIPTION, that should be compliant with general SEO guidelines and of high quality and relevance
Are the posts in line with SEO best practices and standards and with high quality text?
No
Yes
Analyze the content against SEO best practices and text quality benchmarks. Annotate what needs fixing
API call to OpenAI with the initial content, the corrections required and the task to GENERATE A FIXED AND IMPROVED content
Retrieve already generated content that is semantically close to the newly generated ones and add internal links to the older content in the newer content
Save the content locally and publish It to the CMS
End

Code Structure Overview

The repository for the code can be found in GitHub (https://github.com/GregC03/AutomatingBlogPosts) and is structured as follows:

```
project_folder/
 |-- README.md                # Instructions
 |-- requirements.txt         # Required python libraries
 |-- main.py                  # Main script to execute the automation pipeline
 |-- keyword_extraction.py    # Extract keywords using AI and web scraping
 |-- oai_content_generation.py # AI-based blog content generation
 |-- seo_optimization.py      # SEO analysis and fixes
 |-- internal_linking.py      # Internal linking strategy
 |-- cms_integration.py       # Publishing to CMS
 |-- helpers.py               # Utility functions for saving outputs
 |-- .env                     # Environment variables for credentials
 |-- .gitignore               # Instructions for git
 |-- outputs/                 # Generated blog content
 |-- data/                    # Folder for previously created blog posts
 |-- utils/
 |      └── helpers.py        # Helper functions
```

Implemented Steps

Following the automation workflow outlined before, we implemented all the proposed steps in Python. It is to note that the last two steps are implemented in a simple, introductory way that serves as a base for future, more complex, developments.

We decided to organize the code in a modular fashion, with different *.py* files containing the classes and methods necessary to address one task. The file *main.py*, instead, integrates the modules together and unifies the workflow.

Here is a brief description of the tasks implemented in such modules:

1. **Keyword Extraction and Generation** *in keyword_extraction.py*

   In this module we implemented two classes:

   - *GoogleScraper* with methods for addressing web scraping from Google Searches;
   - *KeywordGeneration* with methods for addressing the AI generation of keywords.

   We also used parallelized processing for a more efficient implementation.

2. **Content Generation** *in oai_content_generation.py*

   In this module we implemented three classes:

   - *OaiContentGenerator* with different methods for generating content (blog posts, titles, and meta descriptions), from the keywords, using OpenAI API;
   - *ContentGeneration* with a method that combines all methods from the previous class, and outputs together the blog posts, titles and meta descriptions while parallelizing the generation process;
   - *OaiSeoSpecialist* with methods for fixing SEO problems with AI in already generated content given some instructions and generating improved content.

3. **SEO Optimization** *in seo_optimization.py*

   In this module we implemented two classes:

   - *SEOAnalyzer* with methods for analyzing content with respect to SEO benchmarks and creating a final comprehensive SEO report to be used later on by the methods from the class *OaiSeoSpecialist*.
   - *SEOFixer* with a method that generates the SEO report from the previous class, and uses methods from the *OaiSeoSpecialist* class to generate SEO optimized posts that can then be published;

4. **Internal Linking (Basic Implementation)** *in internal_linking.py*

   In this module we implemented a class, *InternalLinking*, with methods for implementing a simple similarity-based approach used to suggest and generate internal links for the newly generated blog posts.

5. **CMS Integration (Basic Implementation)** *in cms_integration.py*

   In this module we implemented a class, *CMSIntegration*, with a simple method to publish the generated content to WordPress and the possibility of adding compatibility with other CMSs.

6. **Workflow unification** *in main.py*

   In the main file of the program, we import all the necessary classes from the previous modules and implement the end-to-end automatization of the process, from keyword generation to publishing of the generated and SEO optimized blog posts.

   The final program then takes in input one or two values, the topic on which we want to write the post about, and, possibly a pre-made list of keywords to be directly used for content generation instead of letting the program extract and generate them automatically. It finally outputs the generated and optimized posts and publishes them on the CMS of choice if this is enabled via an environment variable.

Code's Key Features

- **Parallel Processing:**

  Improves efficiency by processing multiple blog posts and keyword extractions simultaneously.

- **Environment Variable Management:**

  Credentials and sensitive information are securely handled using .env files.

- **Multiple Export Options:**

  In addition to the publication to the desired CMS, content is saved in CSV, Excel, and Markdown formats for flexibility.

- **Modularity:**

  Given the code's structure it is very easy to implement new features or improve existing ones, especially for Internal Linking and CMS Integration which are still implemented in a simple, introductory way

Limitations & Next Steps

- **Basic Testing Coverage:**

  Current implementation focuses on functional correctness. Advanced testing (unit/integration) can be added for production readiness.

- **Content Quality Validation:**

  Additional machine learning techniques could be implemented to assess and rank content quality automatically.

- **Improve the basic implementation of Internal Linking and CMS Integration**

Applying Machine Learning

While the current solution does not implement machine learning for content validation or personalization, several ML techniques could enhance the process in the future. Here we propose a few that we believe are most relevant.

Proposed ML Approaches

1. **Content Quality Assessment**
   - Train a classification ML model (e.g., logistic regression, random forest classifier, XGBoost) to detect low-quality content based on factors like readability, engagement scores, and SEO compliance.
   - This assessment can then be used to decide whether or not to directly publish a post or if it is better to revise it, improve it, and publish it later on.
2. **Topic Personalization**
   - Use clustering algorithms (e.g., K-Means, LDA, Bayesian Non-Parametrics) to cluster users and their interests, identify emerging trends and personalize the suggestion of blog posts for each user.
3. **Feedback Loop Integration**
   - Incorporate user engagement data (click-through rates, bounce rates) to refine content generation models over time, for instance by pushing the generation of keywords that are connected with content that usually generates more engagement.
   - Alternatively, construct machine learning algorithms to predict topics that will generate good engagement in the future, using the company data and Google Analytics data.

Roadmap for ML Integration

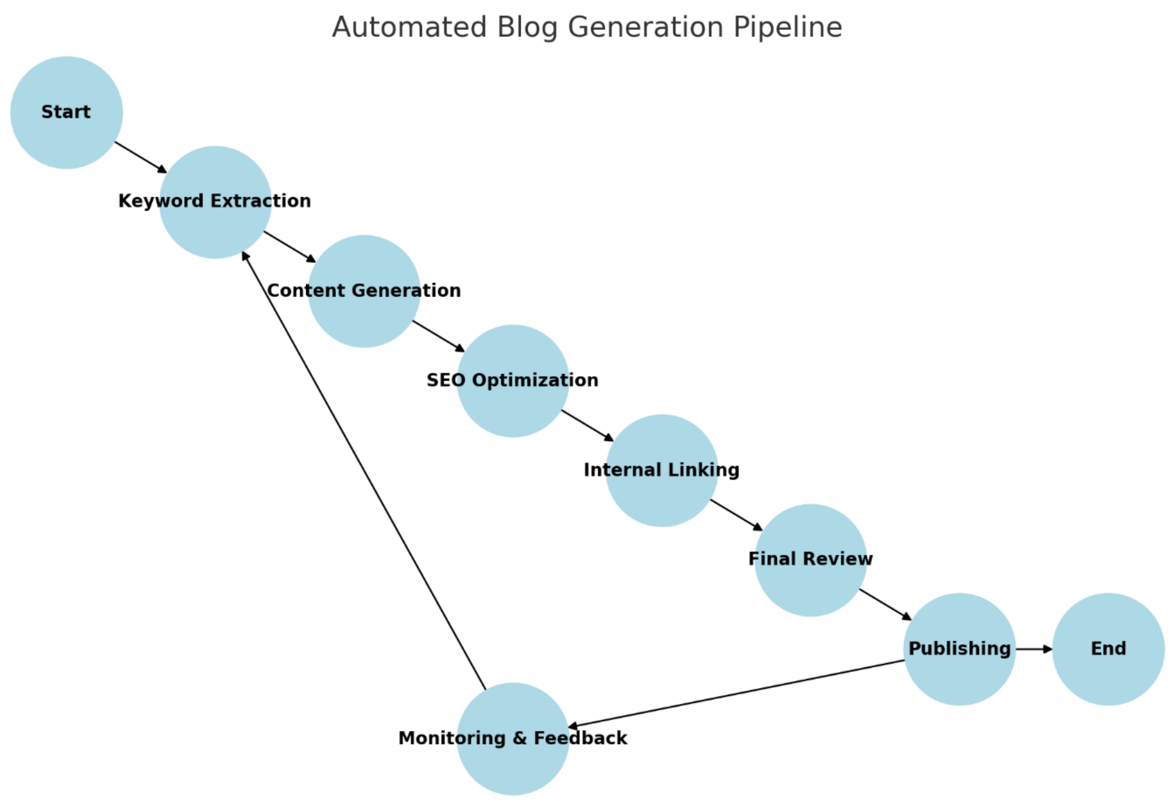
Phase	**Task**	**Expected Outcome**
Phase 1 (Current)	Rule-based SEO and AI content generation	Foundational system in place
Phase 2 (Future)	Implement content quality classifiers	Improved content quality
Phase 3 (Future)	Add personalization and trend analysis	More relevant content for users
Phase 4 (Future)	Develop adaptive learning models	Continuous improvement based on feedback

Workflow Design and Automation

To ensure seamless execution of the pipeline, the workflow can be automated using the following approach:

Proposed Automation Pipeline

1. **Input:**
    - Scheduled keyword extraction and generation via API and web scraping;
    - Parallel processing to improve efficiency.
    - Logging system to track failures in API calls.
    - Periodic execution, possibly daily to keep the website and blog active and reach the goal of 700 blog posts a month.
2. **Processing:**
    - Dynamic AI model selection based on cost and task complexity;
    - Content generation and optimization based on the keywords using OpenAI and ML;
    - SEO optimization using NLP, AI, and ML techniques;
    - Internal linking generation using TF-IDF, cosine similarity or other NLP techniques;
    - Batch processing of API calls to control costs.
    - Parallel execution to handle large volumes efficiently;
    - Error handling & retry logic for failed API requests.
3. **Output:**
    - Save content locally in CSV, Excel, and Markdown formats;
    - Automated CMS publishing (WordPress, but designed to be extensible);
    - Feedback loop (user engagement data is used to improve keyword selection).
4. **Scalability considerations**
    - With daily executions of the pipeline, it means an expected number of 24 posts a day to reach 700 posts a month;
    - This, even in the worst-case scenario, implies not more than 200 OpenAI API calls which is well below current rate limits for any type of account.



Potential Automation Tools

- **Apache Airflow / Prefect** → Job scheduling & dependency management.
- **Docker:** Containerization to ensure consistent execution across environments.
- **AWS Lambda:** For serverless deployment of content generation workflows.
- **Monitoring system** → Tracks API failures and SEO performance.

Deliverables

The following deliverables are included as part of this submission:

1. **Python Implementation:**
   - ZIP file containing the source code implementing the end-to-end blog post generation: keyword extraction, content generation, SEO optimization, internal linking and CMS integration.
2. **Documentation:**
   - This document describing the workflow, technical approach, future improvements and a roadmap for integrating Machine Learning into the workflow.
3. **PowerPoint presentation:**
   - A short slide deck presenting the most important aspects of the project.

Conclusion

This project presents a scalable, automated approach to generating SEO-optimized blogs using AI and Python. While the current implementation lays the foundation, further enhancements such as advanced ML techniques and workflow automation tools can significantly improve the system.

Key Takeaways:

- The solution addresses the core challenge of scaling blog production efficiently;
- It provides an automated 5-step process capable of generating and publishing 700 blogs/month;
- The implementation is modular, allowing easy enhancements over time and seamless integration of new features like ML-based content scoring, clustering of users, and so on;
- It discusses how to ensure scalability and cost-effectiveness with tools such as but not limited to dynamic AI model selection and batch processing;
- Future work can focus on integrating feedback loops, advanced analytics for performance optimization, advanced testing, and fully automating the solution using cloud and pipeline automation tools.

Next Steps

To take this project further, the following next steps are recommended:

1. Conduct exploratory analysis on the outputs to fine-tune keyword and content generation;
2. Implement and automate advanced testing;
3. Implement the proposed ML-based techniques such as but not limited to content quality scoring and validation, topic personalization, feedback loops, and so on;
4. Automate the pipeline using Apache Airflow or similar tools as detailed in the Workflow Design and Automation section;
5. Expand CMS integration beyond WordPress to include platforms like Joomla or Wix.
6. Implement real time dashboards to monitor key metrics such as generated posts, SEO, engagement, and API costs.