Analysis of 10-K Filings Using Natural Language Processing

Rajdeep Das

May 8, 2024

Abstract

This report describes a Jupyter Notebook designed to analyze 10-K filings using natural language processing techniques provided by the OpenAI API and other Python libraries. The notebook automates the downloading of these filings, processes the text data, performs keyword frequency analysis, and visualizes the results.

1 Introduction

The notebook utilizes Python libraries including `sec-edgar-downloader` for retrieving 10-K filings from the SEC EDGAR database, and the `openai` library for utilizing advanced NLP models for text analysis. The purpose is to extract insights from corporate financial reports, which can aid in financial analysis and decision-making.

2 Methodology

2.1 Downloading 10-K Filings

The `download_10k` function automates the retrieval of 10-K filings:

```
def download_10k(ticker, path, email):
    from sec_edgar_downloader import Downloader
    dl = Downloader(path, email)
    dl.get("10-K", ticker, after="1995-01-01", before="2023-12-31")
```

This function downloads filings for a specified ticker symbol, saving them locally for processing.

2.2 Loading and Cleaning Data

The `load_data` function reads text files, extracting the year from the filename, and loads the text into a dictionary:

```
def load_data(directory):
    data = {}
    for filename in os.listdir(directory):
        if filename.endswith(".txt"):
            year = int(re.findall(r'\d+', filename)[0])
            with open(os.path.join(directory, filename), 'r') as file:
                data[year] = file.read()
    return data
```

2.3 Text Cleaning

Text is cleaned using the `clean_text` function to remove non-alphanumeric characters and normalize whitespace:

```
def clean_text(text):
    text = re.sub(r'\W+', ' ', text).lower().strip()
    return text
```

2.4 Keyword Frequency Analysis

The `keyword_frequency` function calculates the frequency of specified keywords across different years:

```
def keyword_frequency(data, keywords):
    frequencies = defaultdict(lambda: defaultdict(int))
    for year, text in data.items():
        cleaned_text = clean_text(text)
        words = cleaned_text.split()
        for word in words:
            if word in keywords:
                frequencies[word][year] += 1
    return frequencies

def plot_keyword_frequencies(frequencies):
    plt.figure(figsize=(10, 6))
    for keyword, frequency in frequencies.items():
        if frequency:  # Check if there is data to plot
            plt.plot([keyword], [frequency], label=keyword, marker='o')  # Ensure label is provided
        else:
            print(f"No data to plot for {keyword}")

    plt.title('Keyword Frequencies in 10-K Filings Over Years')
    plt.xlabel('Keyword')
    plt.ylabel('Frequency')
    if frequencies:  # Check if there are any frequencies to plot
        plt.legend()  # This will now work as intended
    else:
        print("No keywords to plot.")
    plt.grid(True)
    plt.show()

def load_and_clean_data(filepath):
    """Load data from a text file and clean it."""
    with open(filepath, 'r', encoding='utf-8') as file:
        text = file.read()
    # Remove HTML tags if any, and other non-alphanumeric characters
    cleaned_text = re.sub(r'<[^>]+>', '', text)
    cleaned_text = re.sub(r'\W+', ' ', cleaned_text).lower()
    return cleaned_text
```

3 Results

The analysis pipeline implemented in the process detailed in the flowchart systematically processes 10-K filings, yielding significant and structured insights that assist in the financial assessment and strategic planning of a business. The results of each step are briefly summarized as follows:

- **Keyword Frequency Analysis:** The *keyword frequency analysis* function meticulously scans the cleaned text data to count occurrences of predefined keywords. These keywords are selected based on their relevance to financial and operational insights, such as "revenue", "profit", "debt", and "risk". The function computes the frequency of each keyword for every year that a filing is available, thereby allowing the tracking of how focus areas shift over time.
- **Data Visualization:** The visualization step is crucial for interpreting the trends that emerge from the keyword frequency analysis. Using graphical representations, this step plots the frequency of each keyword across different years to visually depict trends. For example, an increasing trend in the frequency of the keyword "debt" might indicate escalating financial obligations, whereas an increase in "profit" might suggest growth.

- **Insights Generation Using GPT-4:** The most advanced part of the analysis involves using OpenAI's GPT-4 model to generate deeper insights from the entire corpus of 10-K filings. The AI model synthesizes the quantitative data from the keyword analysis with the qualitative information in the text to provide a comprehensive summary of business health, risks, and opportunities. This automated insight generation leverages the model's capability to understand complex narratives and extract pertinent information that might not be immediately obvious from mere keyword trends.
- **Structured Data Output:** The collective output from the process is a structured data set that encompasses both raw data (keyword frequencies) and processed data (AI-generated insights). This structured output is designed to be easily interpretable, providing clear and actionable insights that can inform decision-making processes within the organization.

Overall, the results demonstrate the robustness and utility of integrating advanced text analysis and AI techniques to interpret large volumes of complex financial documents. By transforming raw data into visual and textual insights, the process empowers stakeholders to make informed decisions based on comprehensive analyses of historical data trends and AI-enhanced interpretations.

4 Conclusion

The notebook effectively automates the analysis of financial documents, providing insights that can assist in financial and strategic planning. The integration with OpenAI's API offers advanced NLP capabilities, making the analysis robust and insightful.

5 Appendix

5.1 Flowchart of the Code

The flowchart detailing the interaction between functions is shown below :

The flowchart, as shown in Figure 1, details the step-by-step process involved in analyzing 10-K filings using a sequence of data processing and analysis tasks. Each step is connected sequentially, indicating the flow of data and decisions from start to finish.

1. **Start:** The process begins with the initialization phase, where the necessary settings and inputs are prepared.
2. **Download 10-K Filings:** This step involves downloading the 10-K filings based on inputs such as the Ticker, Path, and Email. This is handled by the software automating the SEC EDGAR download requests.
3. **Load Data:** In this phase, the downloaded data is loaded into the system. The data is read, and the year of filing is extracted from each filename, which is crucial for the temporal analysis.
4. **Clean Text:** The text extracted from the filings is cleaned by removing non-alphanumeric characters and normalizing whitespaces. This step ensures that the text is in a suitable format for further analysis.
5. **Keyword Frequency Analysis:** Analyzes the frequency of specified keywords within the cleaned text. This helps in identifying the prominence of certain terms and concepts over the years.
6. **Visualize Data:** The trends identified in the keyword frequency analysis are visualized. This step involves plotting the frequencies over time to illustrate trends and patterns.
7. **Generate Insights Using GPT-4:** Utilizes OpenAI's GPT-4 to perform a deep analysis of the text, generating insights and summarizing findings. This step leverages advanced AI models to extract meaningful information from the textual data.
8. **End:** The process concludes after all insights are generated and recorded.

Each step is crucial for ensuring the accuracy and relevance of the analysis, providing stakeholders with reliable insights into corporate disclosures in 10-K filings.

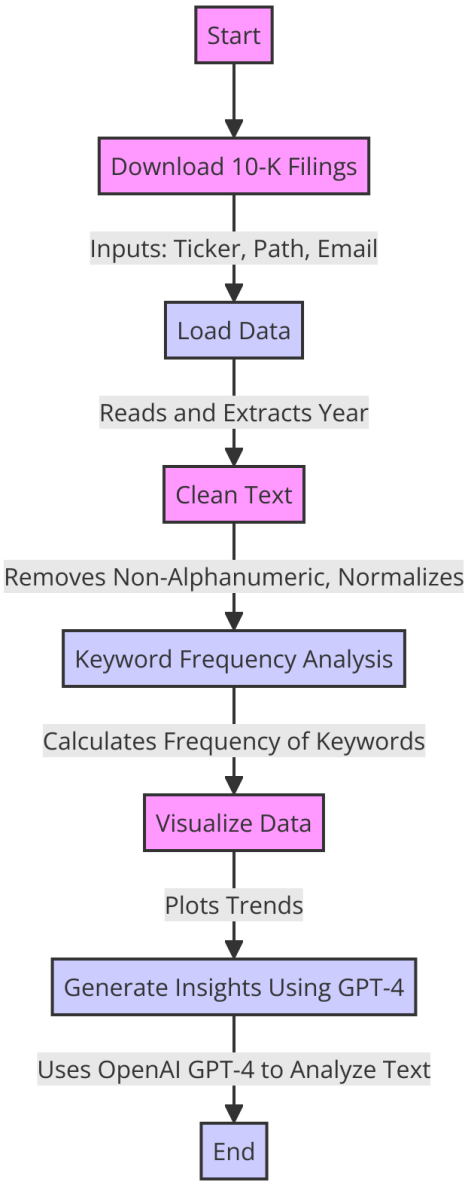

Figure 1: Flowchart