MA3831 Assignment 3 - Capstone Project

Hunter Kruger-Ilingworth - 14198489

Introduction

Predicting the stock market is a topic of great intrest to many people. News and consumer sentiment play a crucial role in influencing stock prices, as traders continuously monitor financial news and social media to assess supply, demand, and market trends. This process requires constant tracking, interpretation, and decision-making to respond effectively to market fluctuations. It is therefore evident that an NLP system could be of great use, being able to (1) summarise articles and identify key information and (2) provide sentiment variables as input features for a stock price prediction model.

This report presents a pipeline that collects news articles using web scraping techniques and processes them through NLP systems to extract meaningful insights.

Web Crawler

Website Selection

Web Scraping

To begin this analysis, we scrape the sitemap of `smallcaps.com.au`. This offers the benefit of having a consolidated list of all article URLs, eliminating the need to emulate human navigation through pages of website search results using `Selenium`. The `lxml` parser is used for this task due to its fast performance and capability to handle both XML and HTML scraping. table 1 shows the distribution of the last modified date of the urls in the XML sitemap.

Article Year	Number of Observations
2023	10,468
2024	2,374
2025	323
Total	**13,165**

Table 1: Number of Observations per Year

Now, individual article content needs to be scraped. The following listing uses `BeautifulSoup` and the `requests` library to extract the contents of a given URL, which was sourced from the sitemap. The `lxml` parser was chosen again for its speed, and since the necessary external C dependency had already been installed with the `lxml-xml` parser during the XML sitemap scraping, no additional setup was required [1]

The contents of the HTML were analysed and parsed into a dictionary capturing the title, author, date, sector, stock codes, and the article body, as demonstrated in the listing below. `BeautifulSoup` was used to extract data from the relevant tags. Below is an example output from the parsing process:

Listing 1: Example HTML extraction code use and dictionary response

```
soup_test = extract_html_from_url('https://smallcaps.com.au/galan-lithium-directors-show-confidence-share-purchase/')
parsed_dict = parse_html_to_dict(soup_test)
parsed_dict

{'title': 'Galan Lithium directors show confidence with $1.4 million share purchase',
 'author': 'Colin Hay',
 'date_string': 'May 23, 2024',
 'sector': 'Mining',
 'stock_codes': ['ASX:GLN'],
 'document': 'Galan Lithium (ASX: GLN) managing director Juan Pablo....}
```

References

[1] L. Richardson, *Beautiful Soup Documentation*, 2004–2025, retrieved April 27, 2025. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/#installing-a-parser