TASK 2

AUTOMATIC NEWS SCRAPING WITH PYTHON

INTRODUCTION:

In this project, we will build an automatic news scraping tool using Python, leveraging the newspaper3k and feedparser libraries. The goal is to automate the extraction of relevant information from news articles, such as titles, authors, publish dates, and content. This project is designed to make the process of collecting and organizing news data from multiple sources more efficient, eliminating the need for manual extraction. The feedparser library will handle the parsing of RSS feeds to gather URLs of articles, while newspaper3k will download and parse the content of these articles. This combination allows us to build a robust news aggregation tool, which can be useful for journalists, researchers, or anyone interested in monitoring news trends across different platforms.

REQUIREMENTS:

- **Python Packages Used**:
    - **newspaper3k**: A library for extracting and parsing newspaper articles.
    - **feedparser**: A library for parsing RSS feeds.
- **Installation Instructions**: Step-by-step guide on how to install the required packages using pip:

  pip install newspaper3k feedparser

SYSTEM DESIGN:

- **Architecture of the News Scraping System**: Overview of the system's design, including how different components interact.

- **Functionality Overview**: Description of the system's core functionalities, including parsing RSS feeds, extracting URLs, and downloading articles.

- **Data Flow Diagram**: A visual representation of how data moves through the system from RSS feed input to extracted news data.

IMPLEMENTATION:

Parsing RSS Feeds:

- **Explanation of RSS Feeds:** Overview of RSS feeds and their role in news aggregation.

- **Code for Parsing RSS Feeds**: Sample Python code using feedparser to parse RSS feeds and extract article URLs.

```
import feedparser
def parse_rss_feed(feed_url):
    feed = feedparser.parse(feed_url)
    return [entry.link for entry in feed.entries]
```

Extracting Article URLs:

- **How URLs are Extracted**: Process of obtaining article URLs from parsed RSS feeds.
- **Code for Extracting URLs:** Python code snippet to extract URLs from the parsed RSS feed data.

Downloading and Parsing Articles:

- **Process of Downloading Articles:** Method for downloading articles using URLs.
- **Code for Parsing Articles:** Example code using newspaper3k to parse and extract content from downloaded articles.

```
from newspaper import Article

def parse_article(url):
    article = Article(url)
    article.download()
    article.parse()
    return {
```

```
    'title': article.title,
    'author': article.authors,
    'publish_date': article.publish_date,
    'content': article.text
}
```

Extracting Relevant Information:

- **Information to be Extracted**: Explanation of the key details to extract, such as title, author, publish date, and content.
- **Code for Extracting Information**: Sample code demonstrating how to extract and store these details.

Testing

- **Testing Methodology**: Outline of the approach to testing the scraping system, including test scenarios and expected outcomes.
- **Test Cases**:
  - **Multiple RSS Feed URLs**: Description of the different RSS feed URLs used for testing and the results obtained.
- **Results and Observations**: Summary of the testing results, including any issues encountered and their resolutions.

CODE:

[9]:
```
import newspaper
import feedparser
```

[10]:
```
def scrape_news_from_feed(feed_url):
    articles = []
    feed = feedparser.parse(feed_url)
    for entry in feed.entries:
        # create a newspaper article object
        article = newspaper.Article(entry.link)
        # download and parse the article
        article.download()
        article.parse()
        # extract relevant information
        articles.append({
            'title': article.title,
            'author': article.authors,
            'publish_date': article.publish_date,
            'content': article.text
        })
    return articles
```

[11]:
```
feed_url = 'http://feeds.bbci.co.uk/news/rss.xml'
articles = scrape_news_from_feed(feed_url)
```

[12]:
```
# print the extracted articles
for article in articles:
    print('Title:', article['title'])
    print('Author:', article['author'])
    print('Publish Date:', article['publish_date'])
    print('Content:', article['content'])
    print()
```

OUTPUT:

CONCLUSION:

The development of an automatic news scraping system using Python has demonstrated the efficiency and effectiveness of leveraging libraries like newspaper3k and feedparser for news aggregation. This project successfully automated the extraction of critical information from news articles, including titles, authors, publish dates, and main content, streamlining the process of collecting and analyzing news data. By automating these tasks, the system not only saves valuable time but also enhances the accuracy of information retrieval. The successful implementation of this project underscores the potential for automation to significantly improve data collection workflows in various domains. Future enhancements could further expand the system's capabilities, such as incorporating advanced features for sentiment analysis or supporting a wider range of news sources, thereby adding even greater value to news aggregation efforts.