### Data Preprocessing Steps

The process begins with the extract\_articles function, which is responsible for pulling data from both the Dawn and BBC websites. This function uses the scrape\_dawn\_articles and scrape\_bbc\_articles functions, each tailored to navigate the specific HTML structure of their respective news sites. These functions fetch the web pages using the requests library, which is robust enough to handle network issues and HTTP errors. The fetched pages are then parsed with BeautifulSoup, a library that facilitates the process of pulling specific content from HTML by allowing us to search for particular tags and attributes.

For BBC articles, the code identifies div elements that match the "dundee-card" and "manchester-card" test IDs, which are likely used to mark individual news articles on the BBC website. Each article's link, title, and description are extracted and stored. Similarly, for Dawn, the articles are found by searching for article tags directly, with further details fetched from nested elements.

The raw data extracted consists of titles, links, and descriptions of articles, which are returned as a list of lists. This list includes data from both websites, merged into a single dataset which then undergoes cleaning in the clean\_data function. This function filters out any entries where the description is missing, ensuring the quality and completeness of our data before it is saved.

### Data Version Control (DVC) Setup

Once the data is cleaned, it is written to a CSV file in the save\_to\_csv function. This CSV file serves as the final output of our preprocessing and is saved at a predetermined path. Following the creation of this CSV file, the push\_to\_dvc function takes over to handle data version control using DVC. This function ensures that every version of the dataset is tracked and reproducible.

Initially, if not already initialized, DVC is set up within the project directory using dvc init. The CSV file is then added to DVC tracking with dvc add, which not only tracks the file but also prepares it for version control by creating a corresponding .dvc file. This file acts as a pointer to the actual data in the DVC cache, allowing DVC to track changes to the data without storing multiple copies of the data itself in the git repository. The changes are then pushed to a configured DVC remote, ensuring that the data is backed up externally and can be retrieved or reverted to a previous state as needed.

To integrate with Git, the .dvc file and any changes to the data preprocessing scripts are added to the repository using git add. These changes are then committed with a descriptive message and pushed to the remote Git repository, providing a comprehensive history of both the code and the data changes. This integration of DVC with Git allows for a robust version control system that handles both the datasets and the codebase, ensuring that each part of our ETL pipeline and its outputs are fully reproducible and traceable.