Natural Language Processing
Course code: CSE3015

Module 5
Web Crawling and Social Media Mining

Prepared by
Dr. Venkata Rami Reddy Ch
SCOPE

Syllabus

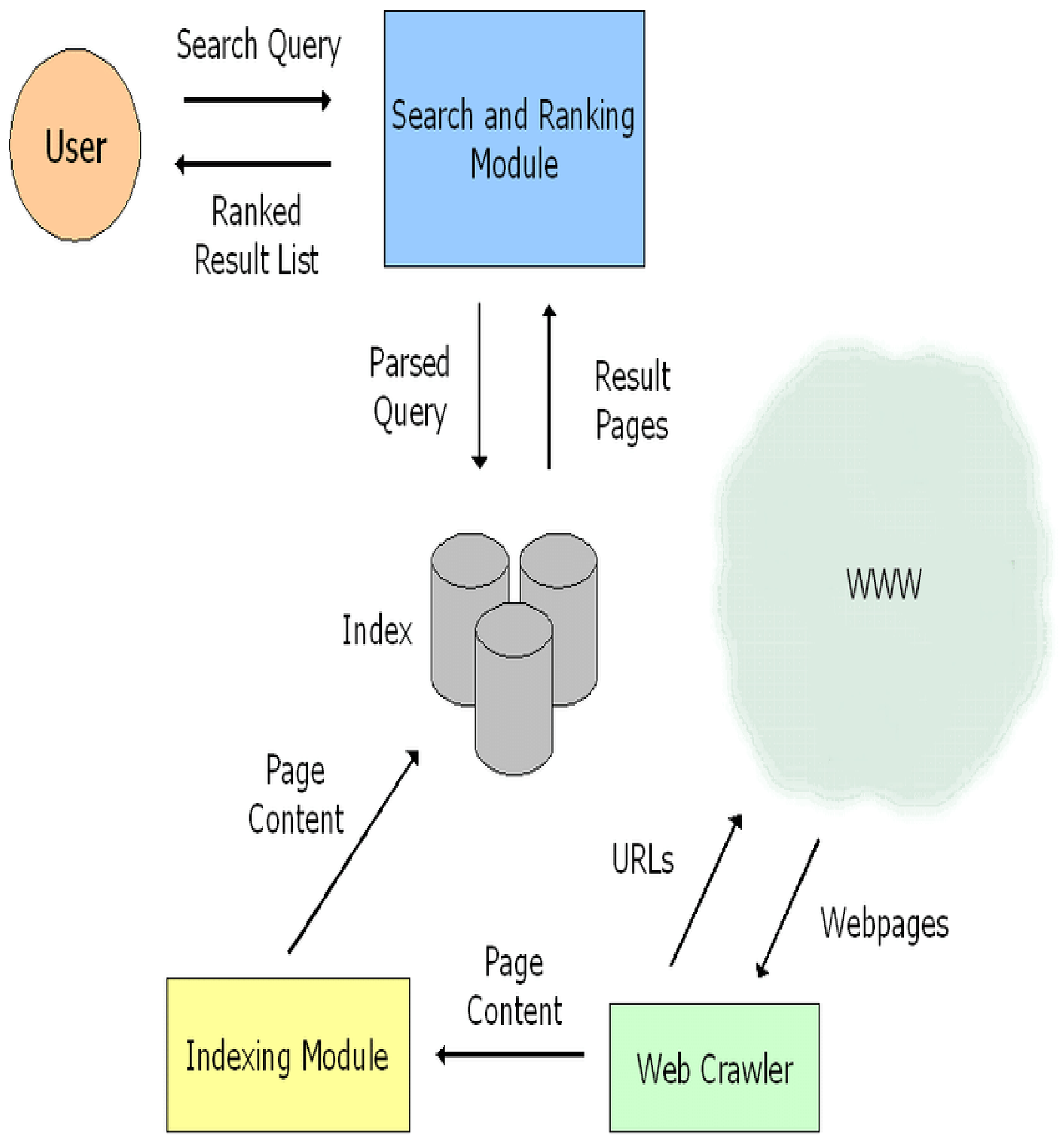
- Web crawler
    - Writing first crawler
- Data flow in Scrapy
- Scrapy shell.
- Social Media Mining
    - Data Collections
    - Data Extraction
    - Geo visualization.

search engine

- A **search engine** is a software system that helps users **find information** on the internet by **searching** through indexed content and returning **relevant results**.
- A **search engine** takes a query (what you type), **searches its index**, and shows the **most relevant web pages** (or documents).

How It Works:

1. **Crawling** – Collects web pages.
2. **Indexing** – Organizes and stores content.
3. **Searching** – Matches your query to relevant pages.
4. **Ranking** – Sorts results based on relevance and quality.
5. **Displaying** – Shows you results with snippets, titles, links, etc.

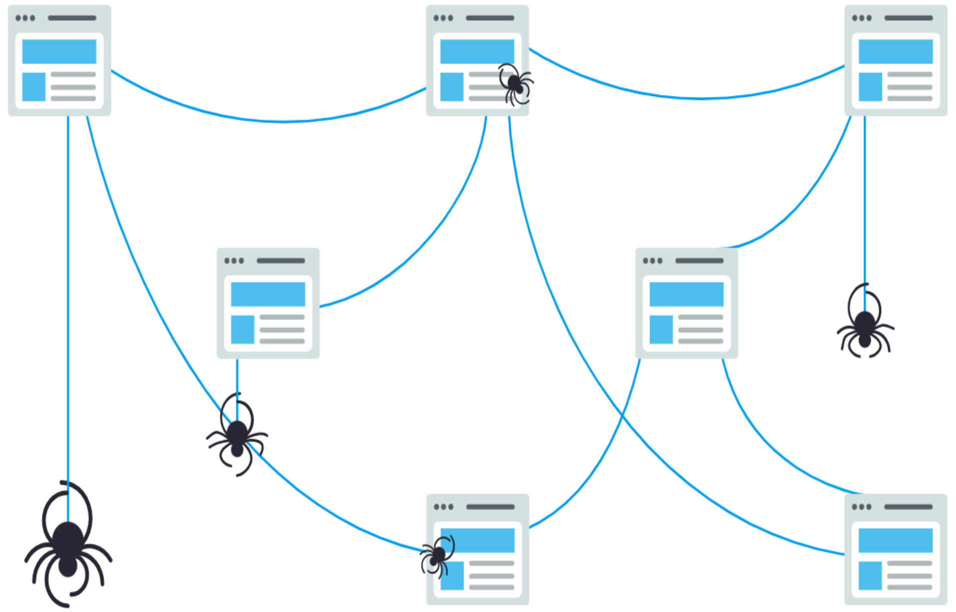


web crawler

- A **web crawler** also known as a **spider** or **bot** is a program that automatically browses the web, discovers new or updated web pages(links), and collects data from them.
- A **web crawler** is a digital bot that crawls the World Wide Web to find and index pages for search engines.
- Web Crawler, also known as Spider or a Robot, is a program that downloads web pages associated with the given URLs, extracts the hyperlinks contained in them and downloads the web pages continuously that are found by these hyperlinks.

What Does a Web Crawler Do?

1.**Starts with a list of URLs** (called *seed URLs*)
2.Visits each URL and downloads the web page
3.**Extracts all the links** on that page
4.Adds new, unseen links to a list (called the *frontier*)
5.Repeats the process for each new URL



How Web Crawlers Work

Start with a list of URLs:

- The web crawler starts with a list of URLs to visit.

Visit each URL:

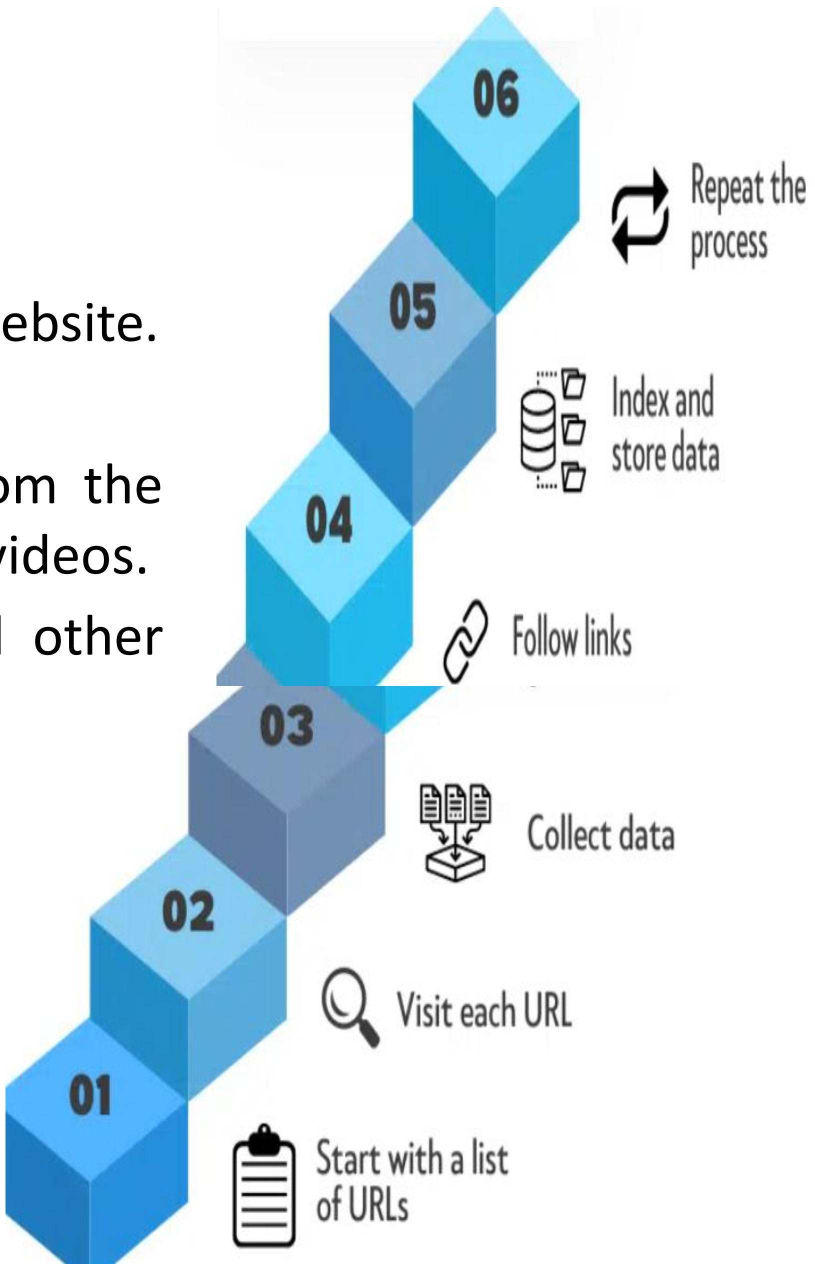
- The web crawler then begins to visit each URL on the list.
- It does this by sending an HTTP request to the server that hosts the website.

Collect data:

- Once the web crawler has accessed a website, it collects data from the HTML code and other resources such as CSS, JavaScript, images, and videos.
- The data collected can include text, images, links, metadata, and other information about the web page.

Follow links:

- The web crawler follows the links found on the web page to discover new pages to crawl.
- It repeats this process for each new page it finds until it has visited all the pages in the original list, or until it reaches a specified depth or limit.



How Web Crawlers Work

Indexing

- **Indexing** is the process of storing and organizing the data collected by the crawler so it can be searched quickly and efficiently.

How it works:

1. **Text Extraction** – Extract raw content (ignoring tags and scripts).
2. **Tokenization** – Break the text into individual words or tokens.
3. **Stop-word Removal** – Remove common words like "the", "is", etc.
4. **Stemming/Lemmatization** – Reduce words to their root form.
5. **Inverted Index Creation** – Create a data structure that maps each word to the pages it appears on.
6. **Metadata Storage** – Store things like page titles, keywords, and frequency of terms.

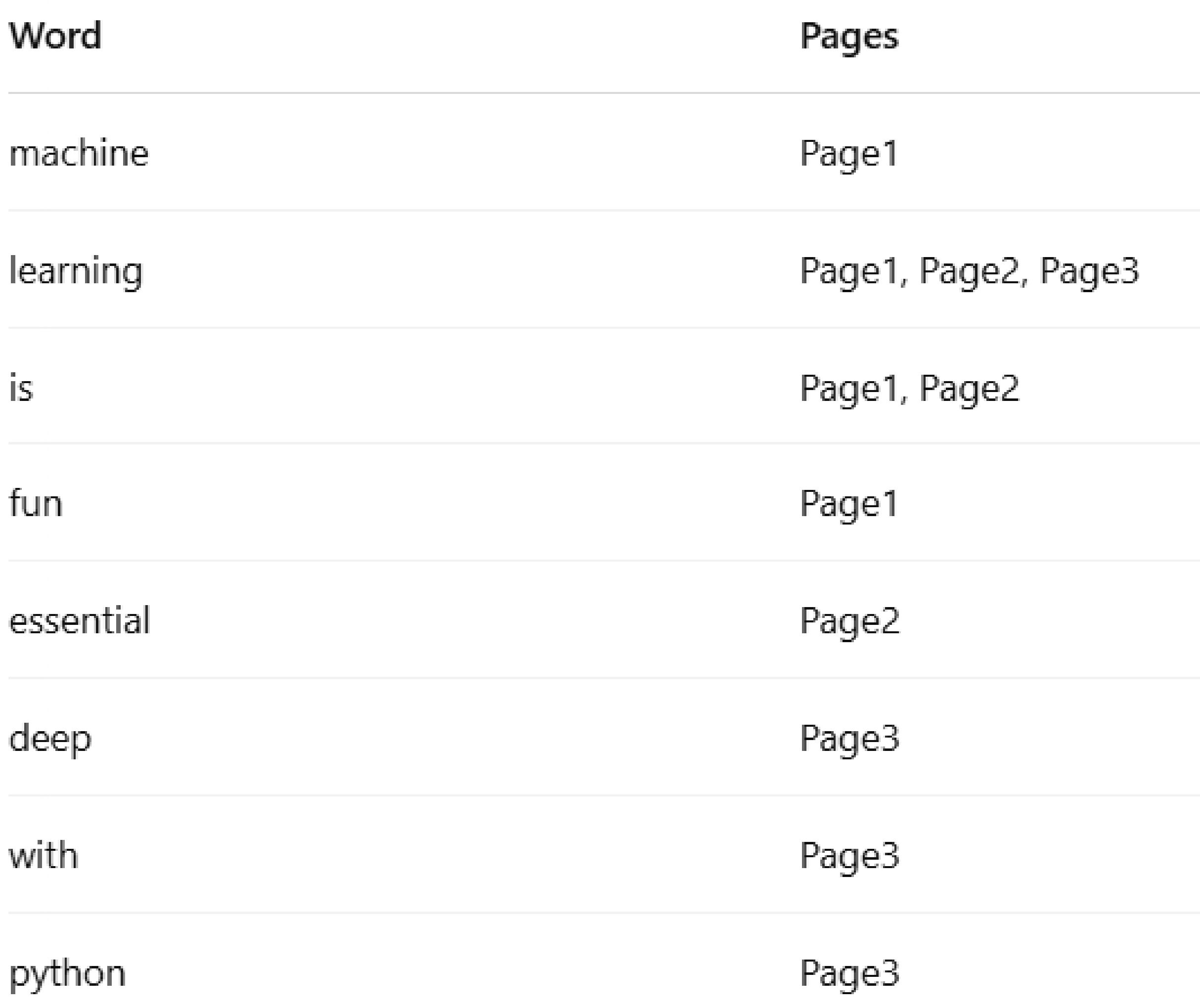
How Web Crawlers Work

Inverted Index Example

Let’s say you crawl 3 pages:

•Page1: "Machine learning is fun"
•Page2: "Learning is essential"
•Page3: "Deep learning with Python“

•The **inverted index** will look like:



Word	Pages
machine	Page1
learning	Page1, Page2, Page3
is	Page1, Page2
fun	Page1
essential	Page2
deep	Page3
with	Page3
python	Page3

Web crawling vs. web scraping

- Web scraping is the process of extracting specific data from web pages, often using automated software or tools.
- Web scraping usually involves targeting specific websites or pages, and extracting only the relevant data, such as product prices or customer reviews.
- Web crawling, on the other hand, is the process of systematically browsing the internet to collect information from websites, without necessarily targeting specific pages or data.
- Web crawlers follow links from one web page to another, and collect information on a wide range of topics, such as web page content, metadata, and URLs.
- While web scraping is often used to extract data for specific purposes, such as market research or data analysis, web crawling is usually used for more general purposes, such as search engine indexing or website optimization.

Web crawling applications

- **Search engine indexing:** Web crawling is essential for search engines like Google, Bing, and Yahoo to index web pages and make them searchable.
- **Website optimization**: Web crawling can help website owners optimize their sites for search engines and improve their overall online presence.
- **Market research:** Web crawling can be used to collect data on competitors, prices, and market trends, helping businesses make informed decisions about their products and services.
- **Social media monitoring:** Web crawling can be used to gather data on social media platforms, such as user activity, engagement rates, and sentiment analysis.
- **News and media:**Web crawling is important for news and media organizations that need to stay up-to-date on the latest events and trends.
- **E-commerce:**Web crawling is crucial for e-commerce businesses that need to gather product information, prices, and other data from competitor websites to stay competitive.
- **Intellectual property protection**: Web crawling can be used to identify and prevent intellectual property violations, such as copyright infringement and trademark infringement.

Examples of web crawlers

- Most popular search engines have their own web crawlers that use a specific algorithm to gather information about webpages.
- Web crawler tools can be desktop- or cloud-based.
- Some examples of web crawlers used for search engine indexing include the following:
- Amazonbot is the Amazon web crawler.
- Bingbot is Microsoft's search engine crawler for Bing.
- DuckDuckBot is the crawler for the search engine DuckDuckGo.
- Googlebot is the crawler for Google's search engine.
- Yahoo Slurp is the crawler for Yahoo's search engine.
- Yandex Bot is the crawler for the Yandex search engine.

Challenges of web crawling

- **Server load:** Frequent crawling can strain your servers with numerous requests, potentially degrading performance for site visitors if left unchecked.
- **Bandwidth consumption:** The transmission of crawl data eats up your capacity and resources.
- **Privacy concerns:** If crawlers collect and distribute sensitive personal information from your site, they raise issues around data protection.
- **Intellectual property issues:** These bots sometimes infringe on copyright by copying and sharing proprietary images, text, or code.
- **Duplicate content:** Identical or near-identical pages on across the web present challenges for crawlers to properly distinguish.
- **International and legal considerations:** Regulations differ globally, so crawlers need policies respecting local jurisdiction over data practices and ownership rights.

Writing first crawler

```
import requests
from bs4 import BeautifulSoup

def simple_crawler(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    links = set()
    for link in soup.find_all('a'):
        links.add(link.get('href'))

    print(f"Found {len(links)} links:")
    for link in links:
        print(link)

simple_crawler('https://www.geeksforgeeks.org/')
```

Code Explanation
- Fetches a page with requests
- Parses it using 'html.parser of BeautifulSoup
- Collects all <a> tags and grabs their href
- Stores unique links in a set
- Prints them nicely

Scrapy

- Scrapy is a fast high-level web crawling and web scraping framework, used to crawl websites and extract structured data from their pages.
- It's widely used for extracting data from websites and can handle more complex tasks compared to basic libraries like BeautifulSoup.
- Scrapy is written in Python

Architecture of Scrapy

Components

Scrapy Engine

- The engine is responsible for controlling the data flow between all components of the system.

Scheduler

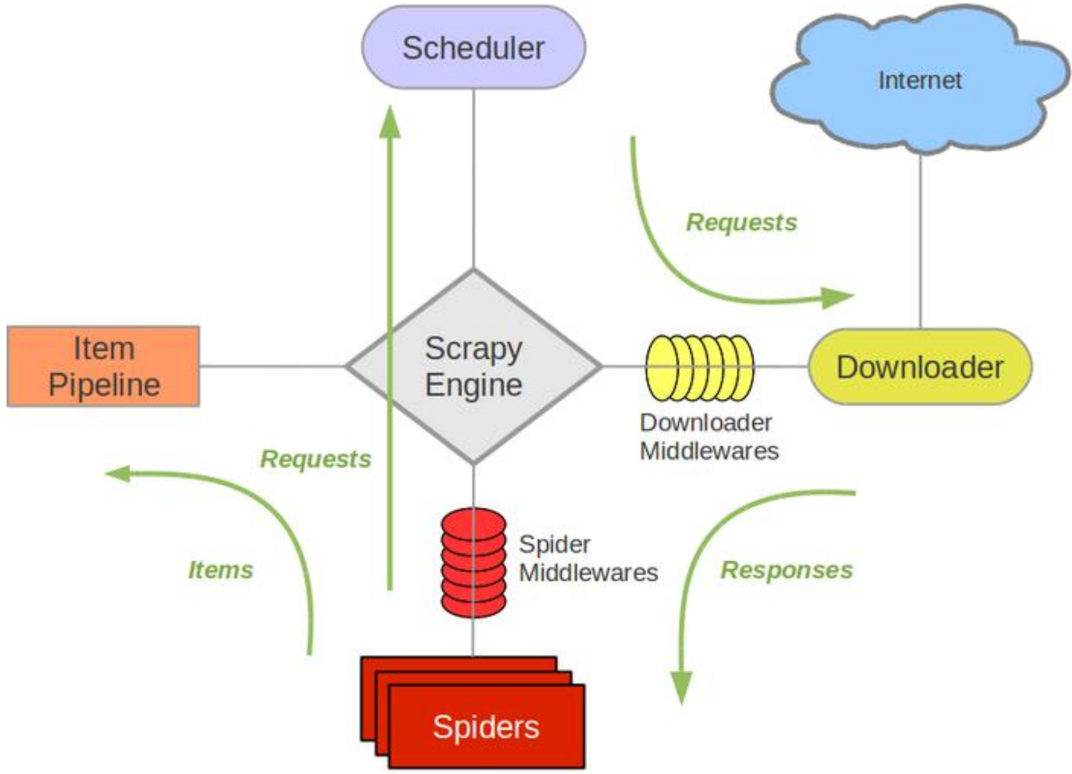
- The Scheduler receives requests from the engine and enqueues them’
- Later on, when the engine requests them, the scheduler is responsible for feeding them

Downloader

- The Downloader is responsible for fetching web pages and feeding them to the engine which, in turn, feeds them to the spiders.

Spiders

- Spiders are custom classes written by users to extract relevant information from the fetched web pages.



Architecture of Scrapy

Item Pipeline

- The Item Pipeline is responsible for processing the items once they have been extracted (or scraped) by the spiders.

•cleansing HTML data, validating scraped data, checking for duplicates ,storing the scraped item in a database

Downloader middlewares

- Downloader middlewares are specific hooks that sit between the Engine and the Downloader.

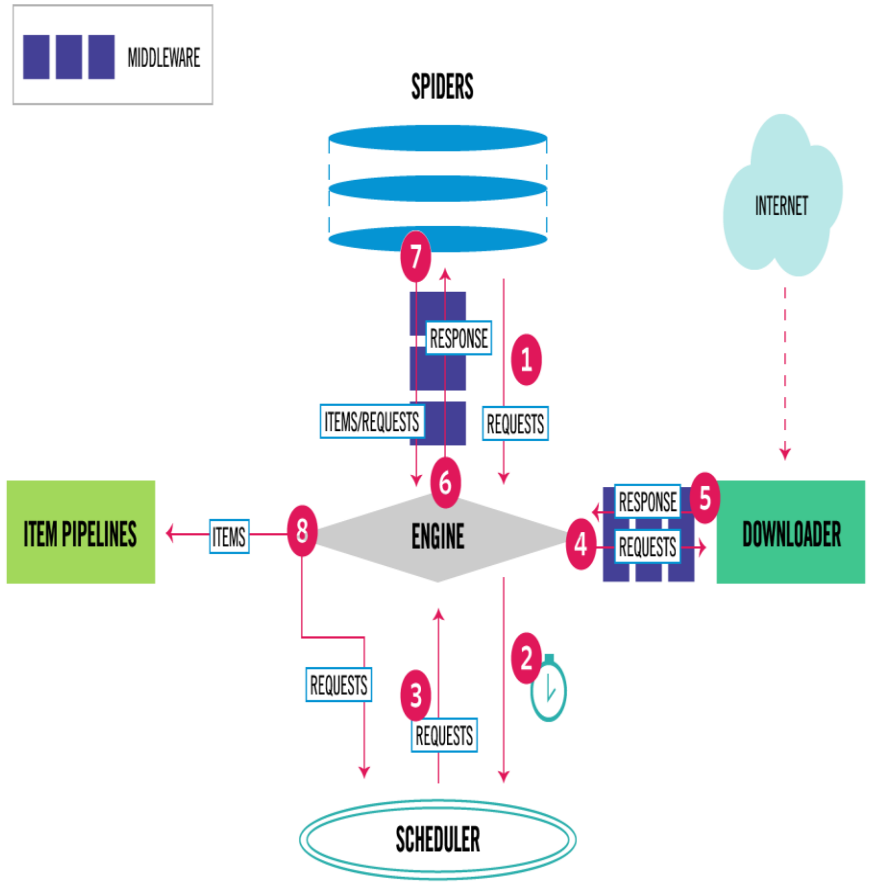
Spider middlewares

- Spider middlewares are specific hooks that sit between the Engine and the Spiders and are able to process spider input (responses) and output (items and requests).

Data flow in Scrapy

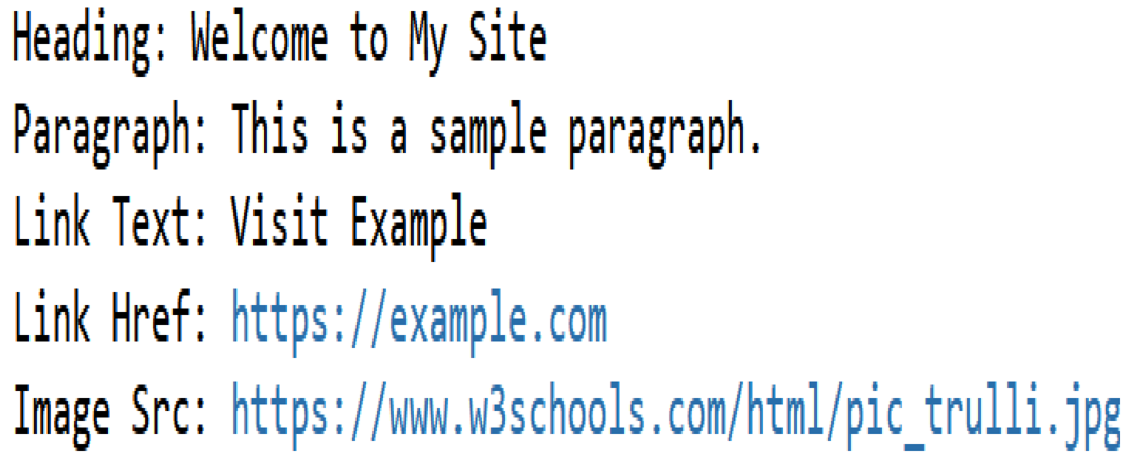
- The data flow in Scrapy is controlled by the execution engine

1. The Engine gets the initial Requests to crawl from the Spider.
2. The Engine schedules the Requests in the Scheduler and asks for the next Requests to crawl.
3. The Scheduler returns the next Requests to the Engine.
4. The Engine sends the Requests to the Downloader.
5. Once the page finishes downloading the Downloader generates a Response (with that page) and sends it to the Engine.
6. The Engine receives the Response from the Downloader and sends it to the Spider for processing.
7. The Spider processes the Response and returns scraped items and new Requests to the Engine.
8. The Engine sends processed items to Item Pipelines, then send processed Requests to the Scheduler and asks for possible next Requests to crawl.
9. The process repeats (from step 3) until there are no more requests from the Scheduler.



Retrieve data from webpage(html code) using scrapy

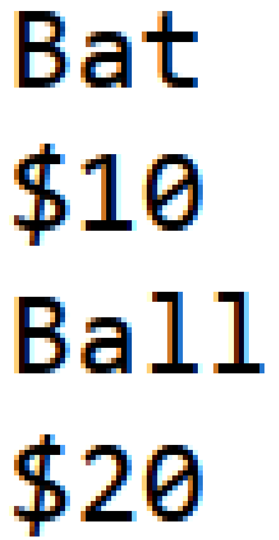
```python
from scrapy.selector import Selector

html = """
<html>
  <body>
    <h1>Welcome to My Site</h1>
    <p>This is a sample paragraph.</p>
    <a href="https://example.com">Visit Example</a>
    <img src="https://www.w3schools.com/html/pic_trulli.jpg" alt="Example Image">
  </body>
</html>
"""
selector = Selector(text=html)
# Extract elements
heading = selector.xpath('//h1/text()').get()
paragraph = selector.css('p::text').get()
link_text = selector.css('a::text').get()
link_href = selector.css('a::attr(href)').get()
image_src = selector.css('img::attr(src)').get()

print("Heading:", heading)
print("Paragraph:", paragraph)
print("Link Text:", link_text)
print("Link Href:", link_href)
print("Image Src:", image_src)
```



```
Heading: Welcome to My Site
Paragraph: This is a sample paragraph.
Link Text: Visit Example
Link Href: https://example.com
Image Src: https://www.w3schools.com/html/pic_trulli.jpg
```

Retrieve data from webpage(html code) using scrapy

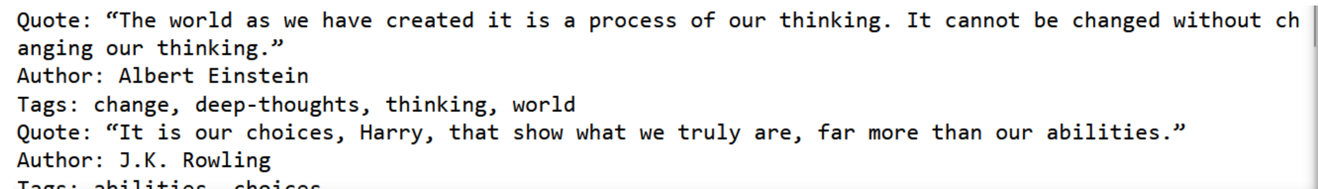
```
html = """
<html>
  <body>
    <div class="product">
      <h2 class="title">Bat</h2>
      <span class="price">$10</span>
    </div>
    <div class="product">
      <h2 class="title">Ball</h2>
      <span class="price">$20</span>
    </div>
  </body>
</html>
"""


selector = Selector(text=html)

for product in selector.css('div.product'):
    print(product.css('h2.title::text').get())
    print(product.css('span.price::text').get())
```



```
Bat
$10
Ball
$20
```

Retrieve data from webpage using scrapy

```python
import scrapy
from scrapy.crawler import CrawlerProcess

class QuotesSpider(scrapy.Spider):
    name = "quotes"

    def start_requests(self):
        yield scrapy.Request(url='http://quotes.toscrape.com', callback=self.parse)

    def parse(self, response):
        for quote in response.css('div.quote'):
            text = quote.css('span.text::text').get()
            author = quote.css('small.author::text').get()
            tags = quote.css('div.tags a.tag::text').getall()

            print(f"Quote: {text}")
            print(f"Author: {author}")
            print(f"Tags: {', '.join(tags)}")

# Run the spider in-process
process = CrawlerProcess()
process.crawl(QuotesSpider)
process.start()
```



```
Quote: “The world as we have created it is a process of our thinking. It cannot be changed without ch
anging our thinking.”
Author: Albert Einstein
Tags: change, deep-thoughts, thinking, world
Quote: “It is our choices, Harry, that show what we truly are, far more than our abilities.”
Author: J.K. Rowling
Tags: abilities, choices
```

Scrapy Shell

- **Scrapy Shell** is an interactive command-line environment where you can:
  - Explore web pages.
  - Try out CSS/XPath selectors.
  - Fetch and inspect HTML content

How to Start Scrapy Shell

```
>scrapy shell http://quotes.toscrape.com
```

- This fetches the page and drops you into an interactive shell with a response object ready to use.

View HTML

```
In [6]: print(response.text)
```

Scrapy Shell

Extract data

```
In [7]: print(response.css('span.text::text').get())
“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.”

In [8]: print(response.css('small.author::text').get())
Albert Einstein

In [9]: print(response.css('div.tags a.tag::text').getall())
['change', 'deep-thoughts', 'thinking', 'world']
```

Social Media Mining

- Social Media Mining is the process of representing, analyzing, and extracting actionable patterns from social media data.
- Social media mining is the process of **extracting useful information, patterns, and insights** from social media platforms like Twitter, Facebook, Instagram, Reddit, etc.
- It combines techniques from: **Data mining, Machine learning, Natural language processing (NLP) and Network analysis**

Applications include:

- Sentiment analysis
- Trend detection
- Influencer identification
- Crisis monitoring
- Market analysis

Social Media Mining

Data Collection

- In social media mining, **data collection** refers to gathering raw social media data, often using:

Web Scraping: Using automated scripts to gather data from social media websites.

APIs: Many social media platforms offer application programming interfaces (APIs) that allow developers to access user data and perform queries.

- **APIs** (e.g., Twitter API, Facebook Graph API)

Streaming services (e.g., Twitter streaming API for real-time data)

Common data collected:

- Posts/tweets
- User profiles
- Likes, shares, and comments
- Hashtags and mentions
- Timestamps and locations (if available)

Social Media Mining

Data Preprocessing

- Remove URLs, mentions, hashtags, emojis, etc.
- Tokenization
- Stop-word removal
- Stemming or lemmatization
- Handling missing values or duplicates

Data Extraction

- Data extraction is the process of extracting specific elements from collected data to make it usable.

It involves:

- **Parsing content** (extracting text, metadata, hashtags)
- **Entity recognition** (users, organizations, places)
- **Language processing** (POS tagging, etc.)

Social Media Mining

Exploratory Data Analysis (EDA)

Understand the dataset:

- Word frequency analysis
- Sentiment distribution
- Hashtag trends
- Most active users or locations

Geo Visualization

- It’s all about **mapping** and **analyzing location-based data**, like geotagged tweets, check-ins, or location mentions in posts..
- **Geo Visualization** refers to the visual representation of data with a **spatial or geographic component**.

Geo-tagged tweets or posts are mapped to visualize:

- Event hotspots
- Regional sentiment
- Crisis areas

Tools: **GIS software** (ArcGIS, QGIS),**Python libraries** (folium, geopandas, plotly, basemap)

Social Media Mining-Use Cases

Brand Sentiment Analysis

Example: Coca-Cola

- Coca-Cola has employed social media mining to monitor brand sentiment across various platforms.
- By analyzing tweets and posts, Coca-Cola can gauge consumer opinions about promotional campaigns, new product launches, and overall brand perception.
- Sentiment analysis has revealed how different demographics react to specific marketing messages, allowing them to optimize their advertising strategies accordingly.

Crisis Management

Example: United Airlines

- In 2017, United Airlines faced a significant public relations crisis when a passenger was forcibly removed from an overbooked flight.
- Social media mining played a critical role in their response strategy.
- By monitoring Twitter and Facebook, tracked the rapid spread of negative sentiment and adjusted their communication to address customer concerns, ultimately working to mitigate damage to their reputation.

Social Media Mining-Use Cases

Health Monitoring

Example: CDC's Flu Tracking

- The Centers for Disease Control and Prevention (CDC) has integrated social media mining into tracking flu outbreaks.
- By analyzing tweets containing keywords related to flu symptoms, they can gauge locations experiencing spikes in flu activity.
- This predictive modeling enables public health authorities to allocate resources more efficiently during peak flu seasons.

Product Development Feedback

Example: LEGO Ideas

- LEGO uses social media mining to gather consumer feedback on new product ideas.
- By analyzing posts and interactions on platforms like Reddit and Facebook, they determine which themes, sets, or minifigures would resonate well with their audience.
- This insight isn't merely about existing products; it informs the developmental process for future releases.