MODUL 12: WEB CRAWLING DAN SCRAPING

12.1 Deskripsi Singkat

Web crawling adalah teknik pengumpulan data yang digunakan untuk mengindeks informasi pada halaman menggunakan URL (Uniform Resource Locator) dengan menyertakan API (Application Programming Interface) untuk melakukan penambangan dataset yang lebih besar. Sedangkan web scraping adalah teknik mengekstraksi data dari suatu halaman web. Web scraping merupakan cara yang powerful untuk mengumpulkan data ketika website tidak menyediakan data API.

12.2 Tujuan Praktikum

Setelah praktikum pada modul 12 ini diharapkan mahasiswa mempunyai kompetensi sebagai berikut:

1) Dapat mempraktekkan Teknik crawling menggunakan API Twitter.
2) Dapat mempraktekkan tekni scraping untuk mengekstraksi elemen tertentu dari halaman web.

12.3 Material Praktikum

Tidak ada

12.4 Kegiatan Praktikum

A. Crawling menggunakan API Twitter

Sebelum melakukan crawling, Anda perlu membuat project terlebih dahulu di halaman developer Twitter untuk mendapatkan Twitter API key. Silakan login Twitter dengan akun masing-masing, lalu akses halaman developer Twitter (https://developer.twitter.com/en). Kemudian klik menu Developer Portal (https://developer.twitter.com/en/portal/petition/use-case), lalu pilih jenis project, misalnya Academic Research -> Student. Isi form yang dibutuhkan untuk membuat project. Harap perhatikan Terms and Condition penggunaan datanya. Setelah membuat project di halaman Developer Twitter, catat consumer key, consumer_secret, access_token, dan access_token_secret yang akan digunakan dalam autentikasi penggunaan AI Twitter.

Kemudian, Anda akan menggunakan library Tweepy dalam melakukan crawling data Twitter menggunakan API Twitter. Oleh karena itu, silakan instal Tweepy terlebih dahulu.

```
pip install tweepy
```

Import semua library yang dibutuhkan sebagai berikut.

```
import tweepy # library untuk akses data twitter
import datetime
import json, time, sys
import csv    # library untuk membuat file CSV
```

Lalu masukkan API key yang sudah didapatkan. Kemudian autentikasi menggunakan key tersebut.

```
consumer_key = 'xxx'
consumer_secret = 'xxx'
access_token = 'xxx'
access_token_secret = 'xxx'

# Autentikasi API Twitter menggunakan Tweepy
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth,wait_on_rate_limit=True)
```

Akses terlebih dahulu google drive Anda, untuk kemudian digunakan sebagai lokasi penyimpanan data yang di-crawling.

```
from google.colab import drive
drive.mount("/content/drive", force_remount=True)
csvFile = open('/content/drive/MyDrive/kuliah/IR/data/gempa.csv', 'a')
csvWriter = csv.writer(csvFile))
```

Selanjutnya Anda akan melakukan crawling dengan memanfaatkan fitur Search.

```
max_tweets = 20
for tweet in tweepy.Cursor(api.search,q="#gempa",count=10,
                           lang="id",
                           since="2022-11-21").items(max_tweets):
# cari tweet dengan hastagh gempa yang berbahasa indonesia dengan max
10 tweet dari tanggal 21 november 2022
    print (tweet.created_at, tweet.text)    # ambil data dari response
berupa tanggal dan isi tweet
    csvWriter.writerow([tweet.created_at, tweet.text.encode('utf-8')])
# masukkan kedalam csv dengan isi tweet dilakukan proses encoding
```

Anda dapat menggunakan atribut objek Tweet lainnya. Silakan melakukan eksplorasi, setelah membaca dokumentasi berikut.

https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/tweet

Selain itu Anda juga dapat secara realtime melakukan crawling dengan fitur Stream.

```
from tweepy.streaming import StreamListener

class StdOutListener(StreamListener):
    def __init__(self, api=None):
        super(StdOutListener, self).__init__()
        self.num_tweets = 0     # buat variabel inisialisasi jumlah
tweet
```

```
    def on_status(self, status):
        record = {'Text': status.text, 'Created At': status.created_at}
# simpan hasil response kedalam variabel (contoh: ambil text dan
created_at)
        print(record)
        self.num_tweets += 1        # setiap satu tweet yang berhasil
didapat dihitung dengan menambahkan satu
        if status.lang == 'id':     # filter tweet yang hanya berbahasa
indonesia
          if self.num_tweets < 20:  # batasi jumlah tweet yang
dicrawling sebanyak 20
              return True
          else:
              return False

    def on_error(self, status_code):
        if status_code == 420:
            print('Error on status', status)
            return False

    def on_limit(self, status):
        print('Limit threshold exceeded', status)

    def on_timeout(self, status):
        print('Stream disconnected; continuing...')


stream = tweepy.Stream(auth, StdOutListener())
stream.filter(track=['gempa'])   # menggunakan fungsi stream filter
untuk mencari kata kunci gempa
```

Silakan melakukan eksplorasi setelah membaca dokumentasi berikut.

https://docs.tweepy.org/en/stable/streaming.html

B. Scraping menggunakan Request dan BeautifulSoup

Selanjutnya Anda akan melakukan scraping suatu halaman website menggunakan library Request dan BeautifulSoup. Import terlebih dahulu library yang diperlukan.

```
from bs4 import BeautifulSoup as Soup
import os
import re
import requests
```

Buat fungsi berikut untuk mengekstraksi konten dari suatu halaman website.

```
def downloader(link):
    req = requests.get(link)
    req.encoding = "utf8"
    return req.text
```

Panggil fungsi tersebut, misalnya untuk mengakses halaman berikut.

```
https://jurnal.stis.ac.id/index.php/jurnalasks/
```

```
contents =
downloader("https://jurnal.stis.ac.id/index.php/jurnalasks/")

print(contents)
```

Perhatikan isi dari variabel contents di atas. Kemudian bandingkan dengan membuka halaman website lalu klik `View Page Source`.

Anda dapat merapikan tampilan output menggunakan kode berikut.

```
soup = Soup(contents, "lxml")
print(soup.prettify())
```

Selanjutnya Anda akan mencoba mengakses elemen html dari halaman web tersebut. Misalnya Anda akan mengakses tag html title.

```
soup.title
```

Anda juga dapat mengakses elemen html dengan atribut tertentu. Misalnya Anda akan mengakses elemen link dengan id = article-361.

```
soup.find_all("a", attrs={"id": "article-361"})
```

Selain itu, Anda juga dapat menerapkan regular expression untuk memfilter isi atribut yang diakses. Misalnya Anda akan mengakses elemen link dengan id yang terdapat string ”article”.

```
soup.find_all("a", attrs={"id": re.compile(r"(article)")})
```

Kemudian ekstraksi judul dari setiap link jurnal tersebut.

```
for u in urls:
  content_u = downloader(u['href'])
  soup_u = Soup(content_u, "lxml")
  print(soup_u.find("h1", attrs={"class": "page_title"}).text)
```

Eksplorasi data lain yang dapat diekstraksi, kemudian simpan ke dalam csv.

12.5 Penugasan

1. Lakukan duplikasi dari latihan di modul ini dengan Google Colab masing-masing.