**Eksamen info 215**

**Pensum kronologisk:**

**Obliger:**

Webscraping with BS:

In this assignment you will extract information from the following wiki page:

https://en.wikipedia.org/wiki/Star\_Wars:\_The\_Rise\_of\_Skywalker

* 1. You have to extract all links from the page as well aswhere they point to (tip: look for the “href” attribute in “<a>” tags). You must print out absolute link reference.

A picture containing text, screenshot, font, software

Description automatically generated

* 1. You have to extract all images src attribute from the page. You must print out absolute link reference.

1. Task 2  
   for img in bs.find\_all('img'):  
    get\_source = img.get("src")  
    img\_full\_url = urljoin(main\_url, get\_source)  
    print(img\_full\_url)

2.a) Analysing code.

• The following python code is used to extract someinformation about the awards that the movie has won. With the following python code, we were able to select the table data that has 'Won' value and print out some related information by accessing its parent element. However, it is not printing the columns with Award, and Date of ceremony values for all selected items. Why it is not printing the column value for Award and Date of ceremony information for the selected awards? Analyse the table structure and python code and provide an answer in plain text (max length 100 words). Optional: You may provide a DOM tree with your answer.

Answer:  
When i try to run the code we get award and date for most columns. But for the for awards won in some awards  
the code does not print which award it is and what date it was won. And thats because date of cermony  
 and the award columns are nonexistent for some of the rows.

Task 2b

from urllib.request import urlopen  
from urllib.parse import urlparse  
from bs4 import BeautifulSoup  
import re  
  
#Task 2b  
html = urlopen('https://en.wikipedia.org/wiki/Star\_Wars:\_The\_Rise\_of\_Skywalker')  
bs = BeautifulSoup(html.read(), 'html.parser')  
  
tds = bs.find\_all('td', {'class':'yes table-yes2 notheme'})  
for td in tds:  
 if td.parent.get\_text().count('\n') > 6:  
 print(td.parent.get\_text())  
  
 elif td.parent.get\_text().count('\n') < 8:  
 find\_th = td.find\_previous('th').get\_text()  
 find\_date = td.find\_previous('th').find\_next\_sibling('td').get\_text()  
 print(f"{find\_th}{find\_date}{td.parent.get\_text()}")

Task 3: a. Start crawling from https://en.wikipedia.org/wiki/Web\_scraping

b. Using BeautifulSoup's find\_all() function get all the links from 'See also' section

c. for each link do the following        c.1 Load selected article page          c.2. Print out selected articles first paragraph

import requests  
from urllib.request import urlopen  
from urllib.parse import urljoin, urlparse  
from bs4 import BeautifulSoup  
  
html = urlopen("https://en.wikipedia.org/wiki/web\_scraping")  
bs = BeautifulSoup(html, 'html.parser')  
  
links = bs.find('div', {'class':'div-col'}).find\_all('a')  
for link in links:  
 main\_url = 'http://en.wikipedia.org'  
 full\_url = urljoin(main\_url, link.get('href'))  
  
 html = urlopen(full\_url)  
 see\_also\_links = BeautifulSoup(html, 'html.parser')  
  
 if len(see\_also\_links.find("p").text) == 1:  
 print(f" webpage: {full\_url} First paragraph is:\n {see\_also\_links.find\_all('p')[1].text}")  
 else:  
 print(f" webpage: {full\_url} First paragraph is: \n{see\_also\_links.find\_all('p')[0].text}")

**Oblig 3:**

Task spiders:

Fra items.py:

import scrapy  
  
  
class Scrapingoblig\_(scrapy.Item):  
 url = scrapy.Field()  
 title = scrapy.Field()  
 authorname = scrapy.Field()  
 author\_profile = scrapy.Field()

**scrapy spider:**

from scrapy.spiders import CrawlSpider, Rule  
from scrapingoblig.items import Scrapingoblig\_  
from scrapy.linkextractors import LinkExtractor  
from urllib.parse import urljoin  
  
#min crawler  
class Vergespider(CrawlSpider):  
 name = "my\_crawler"  
 allowed\_domains = ["theverge.com"]  
 start\_urls = ['https://www.theverge.com/reviews']  
  
 # regler for uthenting av linker. første for å finne linkene jeg ser etter. Regel nummer to for å utelukke andre linker  
 rules = [Rule(LinkExtractor(allow=(r'https://www\.theverge\.com/\d+/\S+',)), callback='parse\_items', follow=True, cb\_kwargs={'is\_article':True}),  
 Rule(LinkExtractor(allow=r'.\*'), callback='parse\_items', cb\_kwargs={'is\_article':False})]  
  
#funksjon som henter ut selve informasjonen  
 def parse\_items(self, response, is\_article):  
 base\_url = "https//:www.theverge.com"  
  
 if is\_article:  
 the\_verge = Scrapingoblig\_()  
 the\_verge['url'] = response.url  
 the\_verge['title'] = response.xpath('//h1/text()').extract\_first()  
 the\_verge['authorname'] = response.xpath('//a[contains(@href,"author")]/text()').get()  
 the\_verge['author\_profile'] = base\_url + response.xpath('//a[contains(@href,"author")]/@href').get()  
 yield the\_verge  
 else:  
 pass

csv output:

author\_profile,authorname,title,url  
https//:www.theverge.com/authors/monica-chin,Monica Chin,Dell’s Latitude 7330 convinced me that business laptops are too expensive,https://www.theverge.com/23630381/dell-latitude-7330-review-design-specs-price-features  
https//:www.theverge.com/authors/jennifer-tuohy,Jennifer Pattison Tuohy,How to stream your security cameras to your Echo Show,https://www.theverge.com/23627516/echo-show-security-cameras-stream-smart-display-how-to  
https//:www.theverge.com/authors/sheena-vasani,Sheena Vasani,Where to preorder the yellow iPhone 14 and iPhone 14 Plus,https://www.theverge.com/23628089/iphone-14-plus-yellow-preorder-apple-verizon-att-t-mobile-deal-sale  
https//:www.theverge.com/authors/monica-chin,Monica Chin,What’s the best student laptop? We asked students,https://www.theverge.com/21369381/best-student-laptops-school-high-college  
https//:www.theverge.com/authors/dan-seifert,Dan Seifert,"Yes, paper-feel screen protectors for the iPad are good",https://www.theverge.com/23634258/ipad-paper-feel-matte-screen-protector-guide  
https//:www.theverge.com/authors/sheena-vasani,Sheena Vasani,Ring’s battery-powered Video Doorbell 4 has dropped to its lowest price to date,https://www.theverge.com/2023/3/10/23632187/ring-video-doorbell-4-apple-macbook-air-m1-mario-kart-deal-sale  
https//:www.theverge.com/authors/antonio-g-di-benedetto,Antonio G. Di Benedetto,The best Mario Day deals on Nintendo Switch games and accessories,https://www.theverge.com/23627426/mario-day-deals-mar10-nintendo-switch-games-microsd-cards-cases-accessories-sale  
https//:www.theverge.com/authors/sheena-vasani,Sheena Vasani,Nintendo’s Mario-themed Switch is now available (and comes with a free game),https://www.theverge.com/2023/3/10/23633051/nintendo-switch-mario-choose-one-bundle-sale  
https//:www.theverge.com/authors/allison-johnson,Allison Johnson,How to edit ProRAW photos on your iPhone,https://www.theverge.com/23634077/iphone-edit-export-proraw-how-to  
https//:www.theverge.com/authors/jennifer-tuohy,Jennifer Pattison Tuohy,This smart oven solved my work-from-home lunchtime conundrum,https://www.theverge.com/23633000/tovala-smart-oven-air-fryer-review

**Task 2: seleneium**Your program should crawl over 6 random twitter pages using hashtag links. In this simple version, you do not have to keep track of already visited pages. Your program should only follow hyperlinks of hashtag links Your program should follow these steps:  1. Load a twitter page from a given url using Selenium 2. Wait for some tweeter feeds to load. You may use WebDriverWait with EC. 3. Select a hashtag randomly from the twitter page. 4. Visit the randomly selected hashtag's hyperlink from step 3 and go to step 1.     Keep counting the number of visited pages and stop crawling if you have visited 6 pages

from selenium import webdriver  
from selenium.webdriver.common.by import By  
from selenium.webdriver.support.ui import WebDriverWait  
from selenium.webdriver.support import expected\_conditions as EC  
import random  
import time  
visited\_pages = 0  
  
  
driver = webdriver.Chrome()  
  
# start url  
driver.get("https://twitter.com/search?q=%23SpaceX")  
  
time.sleep(5)  
  
# venter til siden har lastet  
WebDriverWait(driver, 20).until(EC.presence\_of\_element\_located((By.XPATH, '//div[@id="react-root"]')))  
  
#Kjører en while loop som finner linker via # og åpner den nye siden.  
while visited\_pages < 6:  
 wait = WebDriverWait(driver, 20)  
 # finner alle "#"linker  
 elements = driver.find\_elements(By.XPATH, "//a[@dir = 'ltr'][contains(@href, 'hashtag')]")  
 hrefs = []  
 for elem in elements:  
 hrefs.append(elem.get\_attribute("href"))  
 # velger en tilfeldig link  
 random\_hashtag = random.choice(hrefs)  
  
 # åpner linken  
 driver.get(random\_hashtag)  
 time.sleep(5)  
 # venter på at siden laster  
 WebDriverWait(driver, 20).until(EC.presence\_of\_element\_located((By.XPATH, '//div[@id="react-root"]')))  
  
 # teller besøkte sider.  
 visited\_pages += 1  
  
# Close the browser session  
driver.quit()

**Oblig 4**

**Task:** The zip file contains a csv file with multiple lines where each line contains a url to news articles. The task of this assignment is to extract information following links to the news articles. You may investigate a new python library called Newspaper3k for extracting & curating the news articles. In this assignment you have to write a python program that takes a csv file (with links to news articles) as input and produces another csv file as output. Your program should make a summary of the news article and then extract named-entities from the article summary. The output file should consist of the following information (seperated by comma):  • Link to the source news article  • Named entities   The source csv file may contain redundant news-article links, therefore, your program should keep tract of already visited news article pages.  For named entity recognition you may use spacy library or DBPedia Spotlight

import csv  
import requests  
from bs4 import BeautifulSoup  
import spacy  
  
# Loading English model  
nlp = spacy.load('en\_core\_web\_sm')  
new\_link = ["information from next link:" + "\n"]  
# Opening downloaded csv file and creating new csv file for output  
with open('20230324090000.export.CSV', newline='') as input\_file, \  
 open ('oblig4\_output.csv', 'w', newline='') as output\_file:  
  
 # reader and writer for input and output file. since columns in the input file are divided by tab we use delimiter  
 csv\_reader = csv.reader(input\_file, delimiter= '\t')  
 csv\_writer = csv.writer(output\_file, delimiter= ',')  
 # writes the column names in the new file  
 csv\_writer.writerow(['URL', 'Named entities'])  
  
 #Cunter, because it took a lot of time running the code.  
 url\_counter = 0  
 # to check for already visited urls  
 visited\_url = []  
 for row in csv\_reader:  
 if url\_counter > 10:  
 print(f"more than 10 urls have been visited ")  
 break  
 url = row[-1]  
 if url in visited\_url:  
 (print("already visited"))  
 continue  
 # gives each url 5 sec to open the link. runs an exception if there is something wrong  
 try:  
 response = requests.get(url, timeout=5)  
 except requests.exceptions.RequestException as e:  
 print(f"Error while requesting {url}: {e}")  
 continue  
  
 soup = BeautifulSoup(response.content, 'html.parser')  
 article\_text = soup.get\_text()  
  
 # Spcay to collect the named entities  
 doc = nlp(article\_text)  
 named\_entities = []  
 for entity in doc.ents:  
 if entity.label\_ == 'PERSON' or entity.label\_ == 'ORG':  
 named\_entities.append(entity.text)  
  
 # writes the collected information into the output file  
 csv\_writer.writerow([url] + list(named\_entities))  
 csv\_writer.writerow(new\_link)  
 url\_counter += 1  
 visited\_url.append(url)  
  
print("finished with urls")

example link from input csv: <https://www.msn.com/en-us/news/politics/women-have-more-power-than-ever-in-pennsylvanias-legislature-but-lawmakers-say-big-hurdles-remain/ar-AA18YK7O>

Links are always last element of the row.

**Oblig 5:**

Task: You have to load a graph representing Zachary's Karate club using the following code:  G = nx.karate\_club\_graph()

This graph contains nodes representing member of the club and includes attributes named 'club' which represents which group the members belong after the seperation of the club. You will find either Mr. Hi, or Officer... it indicates two different groups.

# Task 1.1: Print out the list of nodes and edges with attributes.

# Task 1.2: You have to provide a visualization of the Karate club where the color of nodes will represent the two groups after the seperation.

# Task 1.3:  Using nx.dijkstra\_path() find out the shortest path from node 24 to 16.  Print the list of nodes.

# Task 1.4: You have to provide a visualization of the Karate club and modify the color of the nodes by highlighting the selected nodes from the above list. Use red color to indicate the selection of the nodes.

import networkx as nx  
import matplotlib.pyplot as plt  
  
#Task 1.1 and 1.2  
  
graph = nx.karate\_club\_graph()  
  
for n in graph.nodes():  
 if graph.nodes[n]["club"] == "Mr. Hi":  
 graph.add\_node(n, color="green")  
 elif graph.nodes[n]["club"] == "Officer":  
 graph.add\_node(n, color="blue")  
  
node\_colors = {"green":"g", "blue":"b"}  
colors = []  
for n in graph.nodes():  
 color = graph.nodes[n]["color"]  
 colors.append(node\_colors[color])  
print(graph.nodes(data=True))  
nx.draw(graph, with\_labels=True, node\_color=colors)  
plt.show()  
  
#task 1.3  
dijkstra = nx.dijkstra\_path(graph, 24, 16)  
print(dijkstra)  
  
#Task 1.4  
highlight\_color = "red"  
node\_colors\_highlighted = []  
  
for n in graph.nodes():  
 if n in dijkstra:  
 node\_colors\_highlighted.append(highlight\_color)  
 else:  
 node\_colors\_highlighted.append(colors[n])  
  
nx.draw(graph, with\_labels=True, node\_color=node\_colors\_highlighted)  
plt.show()

Task 2:

Use selenium to follow hashtag links.  Your task is to perform breadth first search in two steps.

• In step1, you will perform crawling starting from a hashtag for example, #Oslo.

• In step 2, you will perform crawling starting from a hashtag for example, #Bergen.

While crawling hashtags, you will keep track of already visited hashtags. Your program should not visit a hashtag twice.  Follow minimum 20 hashtags in both steps.  While visiting hashtag pages, your program will add new connections in an undirected graph.   Your program should include error handling.

from selenium import webdriver  
from selenium.webdriver.common.keys import Keys  
from selenium.webdriver.common.by import By  
from selenium.webdriver.support.ui import WebDriverWait  
from selenium.webdriver.support import expected\_conditions as EC  
from urllib.parse import urlparse, parse\_qs  
import random  
import time  
import networkx as nx  
import matplotlib.pyplot as plt  
  
start = "https://twitter.com/i/flow/login"  
crawl1 = "https://twitter.com/search?q=%23Bergen"  
crawl2 = "https://twitter.com/search?q=%23Oslo"  
driver = webdriver.Chrome()  
visited\_links = []  
  
  
  
#remember to input username and password  
def logging\_in(driver, username, password):  
  
 # get starting hashtag  
 driver.get("https://twitter.com/i/flow/login")  
  
 # automatic login.  
 WebDriverWait(driver, 20).until(EC.presence\_of\_element\_located((By.XPATH, "//input[@name='text' and @type='text']")))  
  
 login\_username = driver.find\_element(By.XPATH, "//input[@name='text' and @type='text']")  
 login\_username.send\_keys(username)  
 time.sleep(1)  
 login\_username.send\_keys(Keys.RETURN)  
 time.sleep(1)  
  
 WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.XPATH, "//input[@name='password' and @type='password']")))  
  
 login\_password = driver.find\_element(By.XPATH, "//input[@name='password' and @type='password']")  
 time.sleep(3)  
 login\_password.send\_keys(password)  
 time.sleep(1)  
 login\_password.send\_keys(Keys.RETURN)  
  
  
def visit\_hashtags(driver, num\_pages, crawl, graph\_name):  
 hashtags\_found = []  
 hrefs = []  
 # initialize graph and visited links  
 g = nx.Graph()  
 global visited\_links  
  
 #logging in:  
 logging\_in(driver, "nemis12345", "nemispenis")  
 time.sleep(5)  
  
 #etter innlogging, så starter vi fra #oslo eller #bergen  
 driver.get(crawl)  
 visited\_links.append(crawl)  
  
 parsed\_url = urlparse(driver.current\_url)  
 hashtag\_node\_name = parsed\_url.path.split('/')[-1]  
 previous\_hashtag\_name = hashtag\_node\_name  
  
 visited\_pages = 1  
  
 while visited\_pages <= num\_pages:  
 time.sleep(4)  
 # find all hashtag links on the page  
 elements = driver.find\_elements(By.XPATH, "//a[@dir='ltr'][contains(@href, 'hashtag')]")  
  
 # add current hashtag node to the graph and create an edge from the previous hashtag node.  
 parsed\_url = urlparse(driver.current\_url)  
 hashtag\_node\_name = parsed\_url.path.split('/')[-1]  
 g.add\_node(hashtag\_node\_name)  
 g.add\_edge(previous\_hashtag\_name, hashtag\_node\_name)  
 previous\_hashtag\_name = hashtag\_node\_name  
  
 #find all hashtags and add them to a list for visiting.  
 if elements:  
 for elem in elements:  
 href = elem.get\_attribute("href")  
 if href in visited\_links:  
 continue  
 else:  
 hrefs.append(href)  
 parsed\_hashtag = urlparse(href)  
 hashtag = parsed\_hashtag.path.split('/')[-1]  
  
 if hashtag not in hashtags\_found:  
 hashtags\_found.append(hashtag)  
  
 for h in hashtags\_found:  
 g.add\_node(h)  
 g.add\_edge(previous\_hashtag\_name, h)  
 # visit the hashtag link  
 try:  
 # choose a random hashtag link  
 random\_hashtag = random.choice(hrefs)  
 driver.get(random\_hashtag)  
 WebDriverWait(driver, 20).until(EC.presence\_of\_element\_located((By.XPATH, '//div[@id="react-root"]')))  
  
 # add visited link to the list of visited links  
 visited\_pages += 1  
 visited\_links.append(random\_hashtag)  
  
 #skriver grafen til et eget dokument  
 nx.write\_graphml(g, graph\_name)  
 except:  
 print(f"No more pages to visit{hrefs}")  
 break  
 else:  
 print(f"No hashtags found")  
 break  
 nx.draw(g, with\_labels=True)  
 plt.show()  
 print(len(hrefs))  
 print(len(visited\_links))  
 print(len(hashtags\_found))  
 return g.nodes  
  
print(visit\_hashtags(driver, 25, crawl1, "BergenHashtags.graphml"))  
print(visit\_hashtags(driver, 25, crawl2, "OsloHashtags.graphml"))

**Oblig 6:**

Task 1: : In this assignment you will work with network centrality and modularity using NetworkX library. The details are given below:

• Load a graph using networkx library from the following csv file:         'nutrients.csv':  <https://mitt.uib.no/courses/26797/files/folder/code/L10?preview=3240723>

• Measure centrality based on degree/eigenvector centrality using networkx library.

• Detect modularity of the graph using label propagation or louvain method (use networkx libraries).

• Modify the size and color based on the above measures.

• Use spring layout or fruchterman\_reingold layout for visualizing the graph.

Task 2: Write SPARQL code for these sentences.

* 1. Find all American Politicians whose fathers were also politician.

- SELECT ?politicianLabel ?fatherLabel WHERE {  
 #instance of human  
 ?politician wdt:P31 wd:Q5.  
 #American citizenship  
 ?politician wdt:P27 wd:Q30.  
 #politician   
 ?politician wdt:P106 wd:Q82955.  
 #father  
 ?politician wdt:P22 ?father.  
 #father also politician  
 ?father wdt:P106 wd:Q82955  
   
 SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO\_LANGUAGE],en". } # retrieve labels  
}

* 1. Find all Norwegian Poets whose date of birth is after 1950. (TIPS: USE FILTER)

- SELECT ?poetLabel ?dob WHERE {  
 #instance of human  
 ?poet wdt:P31 wd:Q5.  
 #citizenship  
 ?poet wdt:P27 wd:Q20.  
 #occupation   
 ?poet wdt:P106 wd:Q49757.  
 #born after 1950  
 ?poet wdt:P569 ?dob.  
 FILTER(YEAR(?dob) > 1950)  
   
 SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO\_LANGUAGE], en". }   
}

* 1. Count number of universities in wikidata.

- SELECT (COUNT(?universities) AS ?count) WHERE{  
 #instance of university  
 ?universities wdt:P31 wd:Q3918  
   
 SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO\_LANGUAGE], en". }   
}

* 1. Find all Norwegian Poets who are also politicians. Show their birthplace in a map.

- SELECT ?poetLabel ?birthplaceLabel ?coords WHERE {  
 ?poet wdt:P31 wd:Q5;  
 wdt:P27 wd:Q20;  
 wdt:P106 wd:Q49757, wd:Q82955;  
 wdt:P19 ?birthplace.  
 ?birthplace wdt:P625 ?coords.  
 SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO\_LANGUAGE], en". }  
}

* 1. Show the birthplace of all people in a map who received Nobel prizes.

1. SELECT ?nobelLabel ?winnerLabel ?birthplaceLabel ?coords WHERE {  
    #instance of human and award reciecved  
    ?winner wdt:P31 wd:Q5 ;  
    wdt:P166 ?nobel .  
   # which nobel prize  
    ?nobel wdt:P279 wd:Q7191 .  
    #birthplace   
    ?winner wdt:P19 ?birthplace.  
    #birthplace coords  
    ?birthplace wdt:P625 ?coords  
    SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }  
   }



**Teori:**

**Innhold:**

1. **internetbasics**
2. **Packet switching, TCP/IP, DNS**
3. **OSI Model**
4. **Client-server architecture**
5. **Web based technologies**
6. **Client side web technologies**
7. **HTML basics**
8. **XPath**
9. **What is Web Service? What are the benefits of using WebService?**  
   A web service is a standardized way for systems to communicate with each other over the internet. It uses standard web protocols and technologies such as HTTP, XML, SOAP, and REST to exchange data. The data exchanged can be in different formats like XML, JSON, etc.

**Benefits of using Web Services:**

* 1. **Interoperability:** Since web services are based on open standards, they can be used across different platforms and programming languages.
  2. **Reusability:** Web services can encapsulate important services within your organization that can be reused across different applications.
  3. **Scalability:** As your system grows, web services can be added or upgraded independently of each other.
  4. **Modularity:** Web services are standalone entities. This means that they can be updated or modified without affecting other services.

1. **What do we mean by service oriented architecture (SOA)?**

Service Oriented Architecture is an architectural design pattern based on structuring an application as a collection of services that are network accessible through standard interfaces. These services are self-contained and do not depend on the context or state of other services, promoting reusability, modularity, and maintainability.

1. **What is API**

An API (Application Programming Interface) is a set of routines, protocols and tools for building software applications. APIs are often considered as machine readable interfaces.

APIs can be classified into different types:

1. **Web APIs:** Also known as HTTP API or REST API. These APIs work over the internet, allowing different software applications to communicate with each other over the internet or a network. They typically exchange data in a format such as JSON or XML.
2. **Library or Framework APIs:** These are APIs that provide pre-built functions and methods to perform certain operations, making it easier for developers to use certain features or functionality.
3. **Operating System APIs:** These APIs provide methods and functions for interacting with the underlying operating system. For example, an OS API might provide functions for file handling, network communication, device interaction, etc.
4. **Database APIs:** These APIs are used to interact with a database, providing methods for operations like create, read, update, and delete (often referred to as CRUD operations).

APIs play a crucial role in modern software development, allowing for modular, maintainable, and interoperable software systems.

1. **What is RestAPI ?**

REST (Representational State Transfer) is an architectural style for designing networked applications. A REST API (Application Programming Interface) is a set of conventions for how to send requests and receive responses between client and server. It uses HTTP methods (GET, POST, PUT, DELETE, etc.) and status codes, and it often exchanges data in JSON format.

* 1. HTTP GET: GET request transfers the resource identified by the URI from the server to the client
  2. HTTP POST: Creates a resource identified by the URI
  3. HTTP PUT: Updates a resource identified by the URI
  4. HTTP DELETE: Removes the resource identified by the URI

1. **How does MVC framework work ?**

MVC stands for Model-View-Controller. It's a design pattern that separates an application into three interconnected components:

* 1. Model: The central component, or the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.
  2. View: Any representation of information, a visualization of the model’s data.
  3. Controller: Accepts input and converts it to commands for the model or view. The controller receives user input, manipulates the model, and causes the view to update appropriately.

1. **How does Django MVT work ?**

Django uses a slightly different pattern called Model-View-Template (MVT). It’s similar to MVC, but Django handles the controller part itself and leaves us with the Model, View, and Template.

* 1. Model: Defines the data structure. This is a representation of your database structure, with fields defined for each piece of data you want to store.
  2. View: The view controls what a user sees. It retrieves data from the appropriate model and executes any calculation made to the data, then passes it to the template.
  3. Template: The template is what the user sees, and it’s constructed by the view. It specifies how the data returned by the view should be rendered into HTML.

Django’s MVT and MVC are quite similar, and the difference is mostly about Django handling the controller part and leaving the developer with the model (the data), the view (the pages), and the templates (the design, or how things look).

1. **What is Web Scraping?**

Web scraping is a method used to extract large amounts of data from websites where the data is extracted and saved to a local file in your computer or to a database in a table (spreadsheet) format. This is done by using software known as web scrapers that fetch the web pages and parse the HTML code to extract the data you want.

1. **What are the benefits of Web Scraping?**

**Benefits:**

* 1. Data extraction: It allows you to extract data from multiple websites quickly, which could be time-consuming and difficult to do manually.
  2. Automation: The process can be fully automated, saving significant time and resources.
  3. Competitive advantage: Many businesses use web scraping to gather relevant industry data from various sources.

**Negatives**:

1. Legal issues: Some websites explicitly prohibit web scraping in their terms of service. Violating these terms could lead to legal repercussions.
2. Privacy concerns: Web scraping could be seen as an invasion of privacy if it's used to gather private, personal data.
3. High maintenance: Web scrapers need to be updated often due to the changing nature of the web. If the layout of a website changes, the scraper might need to be adjusted.
4. **Tell me about Basics of parsing. Search patterns and regular expressions.**

Parsing in programming is the process of analyzing a string of symbols (text) according to certain rules (typically a formal grammar). For example, web scraping often involves parsing HTML, which means examining the HTML code and extracting the parts that contain the needed data.

Search patterns and regular expressions (often called "regex") are tools used to identify specific strings of text within larger bodies of text. Regular expressions are sequences of characters that define a search pattern, mainly for use in pattern matching with strings, or string matching.

A simple example of a regular expression could be the pattern "a.b". In this pattern, the "." stands for any character, so this regular expression matches any three-character string where the first character is "a" and the third character is "b". Thus, it would match "aab", "acb", "adb", etc.

Regular expressions can become quite complex and can be a powerful tool for searching and manipulating text. They are used extensively in web scraping to identify the parts of the HTML code that contain the desired data.

1. **What is Web Crawler? How do they work?Positives and negatives?**

A web crawler, also known as a spider or spiderbot, is a type of bot that visits websites and reads their pages and other information to create entries for a search engine index.

Here's a basic description of how web crawlers work:

1. The crawler visits a list of web pages (URLs).
2. It identifies all the hyperlinks in these pages and adds them to a list of places to visit next.
3. It then visits the links in its list, recursively performing the same actions.

**Positives of Web Crawlers:**

* Data collection: Web crawlers are a key tool for gathering and organizing information on the vast and growing internet.
* Search engines: They're used to index the web, making effective search engines like Google possible.
* Data updating: They help in keeping the data up-to-date as they constantly crawl and update the information.

**Negatives of Web Crawlers:**

* Privacy concerns: Web crawlers may access and index sensitive data if not correctly blocked using rules defined in the website's robots.txt file.
* Server load: Frequent and multiple requests by crawlers can overload a website's server.
* Content duplication: If not handled properly, crawlers might lead to content duplication.

1. **What are the critical factors that need to be considered while using a web crawler?**

Error handling

Keeping track of already visited pages

Stopping condition

Continuous update

Legal aspects of scraping:

* 1. Respect for robots.txt: Websites use a file called robots.txt to signal how they want to be crawled. Respecting these rules is important.
  2. Crawl rate limiting: Sending too many requests in a short time can overload the server and might get your IP address blocked.
  3. Copyright and privacy laws: You should be aware of and respect the copyright and privacy laws applicable in your region and the region of the website you're crawling.
  4. Data storage: The data collected by the crawler can be vast, so appropriate data storage and management methods should be in place.

1. **What separates web scraping and web crawling?**

Web crawling is used primarily for indexing the content of websites and building a map of the web. It involves going through every web page of a website, or even every website on the web, and tracking all of them for indexing.

On the other hand, web scraping focuses on a specific set of data on a website. These could be product details, stock prices, sports scores, and so on. So, web scraping is used when you want to extract specific information from web pages.

In short, web crawling is used for data collection, while web scraping is used for data extraction. It's common to use both web scraping and web crawling as part of the same data gathering process.

1. **How to develop a robust and scalable web scraper?**
   1. Define Your Goal: Clearly identify the data you wish to scrape.
   2. Choose the Right Tools: Python and JavaScript are popular choices with various scraping libraries.
   3. Understand the Website Structure: Inspect the HTML to find the elements you want to scrape.
   4. Write Your Scraper: Send an HTTP request and parse the HTML response to extract data.
   5. Respectful and Legal Scraping: Comply with robots.txt, don't overload servers, and respect laws.
   6. Error Handling: Make your scraper resilient against network errors, data extraction issues, and website changes.
   7. Scale Your Scraper: Manage numerous requests, rotate IP addresses, and parallelize for speed.
   8. Store and Use Your Data: Clean, structure, and store your data for convenient usage.
   9. Maintain Your Scraper: Keep your scraper updated to handle changes in the website structure.
   10. Test Your Scraper: Regular testing is essential to ensure proper functioning.
2. **What is jacaScript**

• Javascript is a dynamic computer programming language.

• It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages.

• It is an interpreted programming language with object-oriented capabilities.   
JavaScript can be implemented using JavaScript statements that are placed within the HTML tags in a web page. You can place the script tags containing your jacascript anywhere within your webpage.

1. **What are the challenges of extracting webpage that includes javaScript?**
   1. Dynamic Content: Many websites use JavaScript to load or modify content dynamically. This content may not be present when the page initially loads, which can cause traditional web scrapers, which just fetch the HTML of a page, to miss this content.
   2. User Interactions: Some content may only appear following specific user interactions, like clicking a button or scrolling down, which are often controlled by JavaScript.
   3. Loading Times: JavaScript can cause content to load at different times, which might require your scraper to wait for the content to load before scraping.
   4. Complexity: JavaScript adds an additional layer of complexity to a website's structure, which can make it more difficult to determine how and where the data you want is being loaded from.
2. **How javaScripts work in the client side?**

JavaScript is a client-side scripting language, meaning it runs in the user's browser, not on the server. When a user navigates to a web page, the server sends HTML and JavaScript code to the user's browser. The browser then interprets and executes the JavaScript code, which can manipulate the HTML elements on the page, change styles, or fetch and send data to the server. This allows for interactive web pages that can update and change in response to user actions or other events.

1. **Ways to execute javaScript in a browser and access data.**

If you need to scrape a website that uses JavaScript to load or display its data, you'll need to use a technique that can execute that JavaScript code:

* 1. Headless Browsers: Tools like Puppeteer (Node.js library), Selenium, or Playwright can control a browser (often in a "headless" mode without a visible UI) and execute the JavaScript on a page. These tools can mimic user actions, wait for content to load, and then access the fully-rendered HTML that includes the JavaScript-generated content.
  2. Browser Automation Libraries: Some web scraping libraries, like Selenium, can automate a web browser, allowing you to interact with JavaScript elements and wait for JavaScript events to complete before scraping the data.
  3. JavaScript Injection: You can inject your own JavaScript code into the page to extract the data directly. This can be done using the evaluate function in Puppeteer, for example.

1. **What is a webdriver object?**

The WebDriver object is a bit like a browser in that it can load websites, but it can also be used to find page elements, interact with elements on the page (send text, click, etc.), and do other actions to drive the web scrapers.

1. **What are NLP related concepts and terminologies**

NLP, or Natural Language Processing, involves several key concepts and terminologies. Here are a few:

* 1. Tokenization: The process of breaking down text into smaller units (tokens), usually words or phrases.
  2. Stemming and Lemmatization: Techniques to reduce words to their base or root form. Stemming is a rudimentary rule-based process of stripping suffixes (“ing”, “ed”, “s”), while lemmatization is more advanced and considers the context and the part of speech of a word.
  3. Stop Words: Common words (like "the", "is", "and") that are often filtered out because they don't provide much information for understanding the meaning of a text.
  4. Part-of-Speech (POS) Tagging: The process of marking up the words in a text as corresponding to a particular part of speech (noun, verb, adjective, etc.), based on both its definition and its context.
  5. Named Entity Recognition (NER): The process of identifying named entities (people, locations, organizations, etc.) in text.
  6. Dependency Parsing: Analysis that annotates sentences with structural information, depicting how words relate to each other.
  7. Word Embedding: Techniques used to represent words in vector space that can capture semantic relationships between words.
  8. Sentiment Analysis: Determining the emotional tone behind words to gain understanding of the attitudes, opinions and emotions expressed within a document.

1. **What are Linguistic features of NLP**

The linguistic features involved in NLP can include phonetic:

* 1. Phonetic Features: These involve the sounds in language and can be important in speech recognition and speech synthesis.
  2. Syntactic Features: These involve the structure of sentences. **POS tagging**, parsing, and chunking are techniques that deal with syntactic features.
  3. Semantic Features: These involve the meaning of words and sentences. Techniques dealing with semantic features include word sense disambiguation, semantic role labeling, and semantic similarity comparison.
  4. Pragmatic Features: These involve the use of language in different contexts. This includes dialogue act recognition, anaphora resolution (linking pronouns to the right nouns), and sarcasm detection.

1. **Navigating parse tree**

A parse tree is a tree structure that represents the syntactic structure of a sentence according to a particular grammar.

Each node in the tree represents a construct in the sentence (a word or a group of words). The root of the tree usually represents the entire sentence, with lower levels of the tree representing different parts of the sentence.

Navigation through the parse tree involves traversing from the root to the leaves, examining each node and its children. This can be done in different ways, including depth-first (examining a node's children before its siblings) or breadth-first (examining all nodes at one level before moving to the next level) methods.

Python's NLTK (Natural Language Toolkit) provides a simple way to generate and navigate through parse trees. It includes methods for traversing the tree, finding and viewing subtrees, and other tasks.

1. **Identify Emerging Trends:**Based on the timestamp of documents, new and emerging topics or entities can be identified. For retail business it would enable retailers to figure out which products are becoming popular and which are losing their grip on the market
2. **What is Network**  
   Network: A pattern of interconnections among a set of things.
   * 1. Social network: A network where the things are people and the interconnections are social
     2. interactions.
     3. Social network analysis: The application of graph and network theory to investigate socialstructures.
3. **Tell me about the models and metrics of networks.**
4. **Fundamentals of Graph theory** 
   1. **Types of graph** 
      * Undirected graphs - A graph in which edges have no direction. The edges indicate a two-way relationship, in that each edge can be traversed in both directions.
      * Directed graphs - A graph in which edges have directions.
      * Weighted graphs - A graph in which each edge is assigned a weight or cost.
      * Unweighted graphs - A graph in which the edges do not have a weight or cost associated with them.
      * Cyclic Graph: A graph that has at least one cycle (a closed path where the first node is the same as the last node).
      * Acyclic Graph: A graph with no cycles.
   2. **Definition of path** 
      * A path in a graph is a sequence of edges that connects a sequence of nodes.
   3. **Dijkstra's shortest path algorithm**
      * Dijkstra's algorithm is a method for finding the shortest paths between nodes in a graph. It works by visiting vertices in the graph starting from the source vertex, at each step picking the vertex with the smallest path length from the source, and relaxing all of its edges (updating the path length if a shorter path is found).
   4. **How to calculate path length and diameter in graphs**
      * The distance (d) between two nodes in a graph is the length of the shortest path linking the two nodes.
      * The diameter of the graph is the maximum distance between any pair of its nodes.
5. **Node attributes and how to add nodes and edges in python code**

In Python, with the library NetworkX, you can create nodes and edges, and add attributes to them as follows:

Add a single node**: G.add\_node(“node\_name”)**

Adding nodes from a list**: G.add\_nodes\_from([Per, Pål, Pung])** adds the list as nodes.

Adding edge: G.add\_edge(“from\_node, to\_node”)

Adding edge from list: G.add\_edges\_from([“("folates", "liver"), ("folates", "asparagus")])

NetworkX offers three options for setting node and edge attributes:

1.Define attributes at the time of adding nodes or edges:

* 1. G.add\_node("Honey", edible=True)
  2. G.add\_nodes\_from([("Steel", {"edible" : False}), ])
  3. G.add\_edge("Honey", "Steel", weight=0.0)
  4. G.add\_edges\_from([("Honey", "Zn"),], related=False)

2. Define or change an attribute of existing nodes and edges by calling nx.set\_node\_attributes() or

b. nx.set\_edge\_attributes():

c. nx.set\_node\_attributes(G, node\_dict, att\_name)

d. nx.set\_edge\_attributes(G, edge\_dict, att\_name)

3. Define or change an attribute of individual existing nodes and edges directly through the dictionary interfaces G.nodes (indexed by node labels) and G.edges (double indexed by start and end node labels):

a. G.nodes["Zn"]["nutrient"] = True # Zinc is a nutrient

b. G.edges[("Honey", "Zn")]["weight"] = 0.95 # Zinc and beef are connected

The dictionary interface allows you to remove unwanted attributes:

c.del G.nodes["Zn"]["nutrient"]

d. del G.edges[("Honey", "Zn")]["weight"]

1. **Partitions/communities in graphs, why?**

Partitioning or finding communities in graphs is a common task in network analysis, useful in many applications. The goal is to divide the nodes of the graph into groups, such that there are more edges within each group than between groups. This can help reveal the underlying structure of the network, identify important nodes, and predict missing edges among other things. Examples of algorithms for community detection include Louvain Modularity and Girvan-Newman algorithm.

1. **Degree**

In graph theory, the degree of a vertex (or node) in a graph is the number of edges that are incident on it. For an undirected graph, it simply counts the number of edges connected to a particular vertex. If the graph is directed then we distinguish between the in-degree and the out-degree of a vertex.

The in-degree of a vertex is the number of incoming edges to the vertex.

The out-degree of a vertex is the number of outgoing edges from the vertex.

The concept of degree in graphs is fundamental in network analysis, as it helps identify important vertices in the graph. For instance, in social network analysis, a node with a high degree could be seen as an influential person.

1. **Graph layouts**

A picture containing text, screenshot, font, number

Description automatically generated

1. **Understand the difference between basic centrality measures Calculation of various centrality for networks**
2. **Adjacency matrix**

The adjacency matrix, sometimes also called the connection matrix, of a simple labelled graph is a matrix with rows and columns labelled by graph vertices, with a 1 or 0 in position according to whether nodes are adjacent or not.

Properties: For a simple graph with no self-loops, the adjacency matrix must have 0's on the diagonal. For an undirected graph, the adjacency matrix is symmetric.

1. **More about social networks**

Individuals, groups, and organizations also form networks. Such networks are called social networks. Social network nodes are explicitly related through friendship, kinship, and membership.  
Two different forms

* 1. **Egocentric networks**
     + An egocentric network (or ego network, for short) is the social network of a particular individual. An ego network includes all the individual’s contacts and all the relationships among them. Egocentric networks are used to understand the structure, function, and composition of connections around a single person. The central node of an ego network is referred to as ego (as in egoism and alter ego); all the other nodes are called alter (as in alternative and alter ego, again). To construct a social ego network, start with an ego—say, yourself. Obtain the list of the ego’s contacts—the alters.
  2. **Sociocentric networks** 
     + A sociocentric network, or just a social network, is any social network that is not egocentric. Ideally, a sociocentric network is a combination of the ego networks of all egos and includes all relevant (whatever it means to you as a researcher) alters.  
       A sociocentric network is the prime focus of attention of social network analysts. It reveals all significant relationships of each actor in the network, exposes hierarchical groups of actors, and provides a framework for explaining the structure and evolution of individual edges and node groups.

1. **What is Graph density?**Graph density measures the fraction of existing edges out of all potentially possible edges. Density is a number between 0 and 1, inclusive. A network with density 0 has no edges whatsoever.  
   A network with density 1 is a complete graph. *For a directed network with n nodes and m edges, density is calculated as m/(n(n-1)); for undirected networks, it is calculated as 2m/(n(n-1)),* because, compared to directed networks, they have only half of potentially possible edges. You can measure density by calling a namesake function. nx.density(G)
2. **What Is centrality**

The centrality of a network is a quantitative measure for revelling the importance of a node. A node with more centrality is expected to be more important in a network. Several centrality indices exists such as degree centrality, closeness centrality, betweenness centrality, etc. Different centrality indices capture different properties of a network.

1. **What is Degree centrality**

Degree centrality is the simplest centrality measure. For a node, it is the number of connections it has to other nodes. In a directed graph, we can consider in-degree (incoming connections) and out-degree (outgoing connections) centralities.  
  
deg\_centrality = nx.degree\_centrality(G)

1. **What is Closeness centrality**Closeness centrality of a node is the average length of the shortest path between the node and all other nodes in the graph. Thus, nodes with lower average shortest paths have higher closeness.V is a set of all vertices in a graph G. We use dG (s,t) to denote the distance between vertices s and t, i.e. the minimum length of any path connecting s and t in G.

close\_centrality = nx.closeness\_centrality(G)

1. **What is Betweenness centrality**

Betweenness centrality measures the extent to which a certain vertex lies on the shortest paths between other vertices. In other words, it helps identify individuals who play a “bridge spanning” role in a network. Vertices with high betweenness may have considerable influence within a network by virtue of their control over information passing between others. They are also the ones whose removal from the network will most disrupt communications between other vertices because they lie on the largest number of paths taken by messages.  
  
bet\_centrality = nx.betweenness\_centrality(G, normalized = True, endpoints = False) # parameters normalized # and endpoints ensure whether we normalize the value # and consider the endpoints respectively.

1. **What is Eigenvector centrality**

Eigenvector centrality influence scores for strategically connected people. Eigenvector centrality is a more sophisticated view of centrality: a person with few connections could have a very high eigenvector centrality if those few connections were to very well-connected others. Eigenvector centrality allows for connections to have a variable value, so that connecting to some vertices has more benefit than connecting to others. Eigenvalue centrality is possibly the best measure of social media influence. A high EigenCentrality score indicates a strong influence over other nodes in the network. It is useful because it indicates not just direct influence, but also implies influence over nodes more than one ‘hop’ away.  
  
The following function can be used to measure eigenvector centrality of a graph. eigenvector\_centrality(G, max\_iter=100, tol=1e-06, nstart=None, weight=None)   
Parameters:

* + G (graph) – A networkx graph
  + max\_iter (integer, optional (default=100)) – Maximum number of iterations in power method.
  + tol (float, optional (default=1.0e-6)) – Error tolerance used to check convergence in power method iteration.
  + nstart (dictionary, optional (default=None)) – Starting value of eigenvector iteration for each node.
  + weight (None or string, optional (default=None)) – If None, all edge weights are considered equal. Otherwise holds the name of the edge attribute used as weight.
  + Returns nodes – Dictionary of nodes with eigenvector centrality as the value.

1. **What is Pagerank centrality**

PageRank is a variant of eigenvector centrality, which was originally used by Google to rank websites in their search engine results. It assigns a relative importance to each node based on the sum of the PageRank scores of all nodes that link to it, adjusted by the number of links from those nodes*.  
G = nx.DiGraph(nx.path\_graph(4)) pr = nx.pagerank(G, alpha=0.9) # alpha is the dumping parameter for pagerank*.

1. **What is a community?**

Nodes form tightly knit groups called communities.

1. **How to identify communities?**

**•**Number of edges that connect vertices within a community are high.

•Number of edges across communities are low.

1. **What is Modularity?**

* In the context of network science, modularity is a metric that measures the strength of division of a network into modules (also called groups, clusters, or communities). Networks with high modularity have dense connections between the nodes within modules but sparse connections between nodes in different modules.
* More formally, modularity is defined as the fraction of the edges that fall within the given modules minus the expected such fraction if edges were distributed at random. The value of the modularity lies in the range [-1/2,1]. It is positive if the number of edges within groups exceeds the number expected on the basis of chance. For a given division of the network, modularity reaches its maximum value if the number of intra-module edges is as high as possible, and the number of inter-module edges is as low as possible.  
  Modularity is often used in optimization methods for detecting community structure in networks, where the aim is to maximize the modularity. One popular example of such a method is the Louvain method.
* *modularity(G, communities, weight='weight')   
  Returns the modularity of the given partition of the graph. Parameters*
  + *G (NetworkX Graph)*
  + *communities (list or iterable of set of nodes) – These node sets must represent a partition of G’s nodes.*
  + *weight (string or None, optional (default=”weight”)) – The edge attribute that holds the numerical value used as a weight. If None or an edge does not have that attribute, then that edge has weight 1.*

1. **what is Community detection algorithm?**

The algorithms for community detection are categorized into approaches based on:

* + **graph partitioning**
    - Graph partitioning is a technique where a graph is divided into smaller components, called partitions or subgraphs, such that the number of edges that cut across the subgraphs is minimized and the number of edges within each subgraph is maximized. The aim is to optimize computational efficiency for operations on large graphs and to detect community structure, among other things.
  + **Clustering**
    - In the context of graphs, clustering refers to the task of identifying densely connected subgraphs, i.e., groups of nodes that have a higher number of edges within the group than with the rest of the graph. These densely connected groups are often called clusters or communities. Graph clustering is used in network analysis to study the structure and properties of the network. One such clustering is K-means clustering.
  + **Louvain method**
    - The Louvain method is a popular community detection method for large networks. It is a heuristic method based on modularity optimization. The algorithm is composed of two phases that are repeated iteratively: (1) in the first phase, it looks for "small" communities by optimizing modularity locally, (2) in the second phase, it aggregates nodes belonging to the same community and builds a new network whose nodes are the communities. These steps are repeated iteratively until a maximum of modularity is attained.  
        
        
      Applying Louvain method using network:

A screenshot of a computer program

Description automatically generated with low confidence

* + **label propagation**
    - Label propagation is another method used for detecting communities in networks. It is a fast, nearly-linear time algorithm that works by propagating labels (representing community identifiers) throughout the network and forming communities based on the process of label propagation. Each node is initialized with a unique label and at every step, each node adopts the label that most of its neighbors currently have. This process is iteratively repeated until convergence.
    - How to find clusters with networkx's label propagation algorithm ?  • import networkx.algorithms.community as nx\_comm G = nx.barbell\_graph(3, 0) nx\_comm.modularity(G, nx\_comm.label\_propagation\_communities(G))

1. **Semantic web technologies**
   * **What is the semantic web?**
     + The Semantic Web is an extension of the current web in which information is given well-defined meaning, enabling computers and people to work in better cooperation. It's a vision of information that's understandable by computers, enabling them to perform more of the tedious work involved in finding, combining, and acting upon information on the web.
2. **RDF**

Resource description framework(RDF) is a standard model for data interchange on the web recommended by the W3C (World Wide Web Consortium). It provides a way to describe relationships between data and objects, making it easier to merge data from different sources. RDF uses triples, which consist of a subject, a predicate, and an object, to make statements about resources in a way that can be interpreted by machines.  
  
Here are examples of RDF triples (informally expressed in

pseudocode):

• EXAMPLE 1: Sample triples (informal)

• <Bob> <is a> <person>.

• <Bob> <is a friend of> <Alice>.

• <Bob> <is born on> <the 4th of July 1990>.

• <Bob> <is interested in> <the Mona Lisa>.

1. **SPARQL**

SPARQL is a RDF query language, also standardized by W3C, that's used to retrieve and manipulate data stored in RDF format. It allows users to write queries against what can be seen as a relational database using RDF triples. SPARQL is capable of querying complex patterns of RDF triples, which include conjunctions, disjunctions, and optional patterns.

**Functions of SPARQL:**

COUNT • YEAR • UNION • MINUS • LIMIT • OPTIONAL • ORDER BY  ASC() / DESC()  • VALUES • FILTER

1. **Knowledge Graph**

A knowledge graph is a powerful data structure that allows the representation of real-world entities and their interrelationships. It's essentially a graph-based structure where nodes represent entities and edges represent relationships between entities.

* Features of knowledge graph:
  + Flexible relations among knowledge in topical domains: A knowledge graph
    - (i) defines abstract classes and relations of entities in a schema,
    - ii) mainly describes real world entities and their interrelations, organized in a graph
    - (iii) allows for potentially interrelating arbitrary entities with each other, and
    - (iv) covers various topical domains.
  + General structure: A network of entities, their semantic types, properties, and relationships.
  + Supporting reasoning over inferred ontologies: A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge.

Showing hospitals in western Europe. (Might not be 100% correct)

#map of hospitals

#deafult map view

SELECT distinct ?hospital ?geo WHERE {

#instance of hospital

?hospital wdt:P31/wdt:P279 wd:Q16917.

#lokasjon

?hospotal wdt:P17 wd:Q27496.

?hospital wdt:P625 ?geo.

}