Molekulare Daten von Bakterien in einen Knowledge Graph überführen
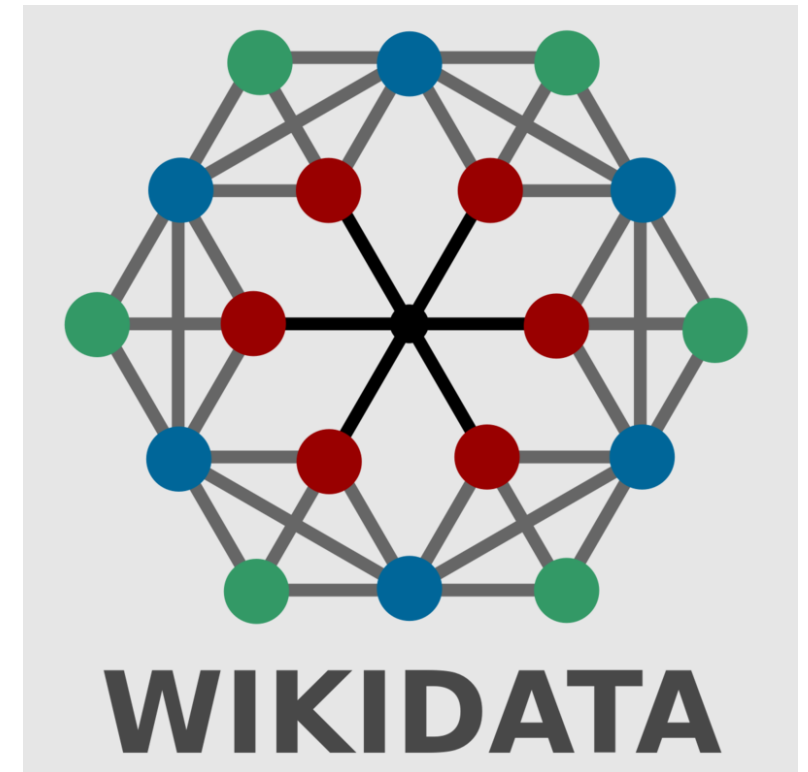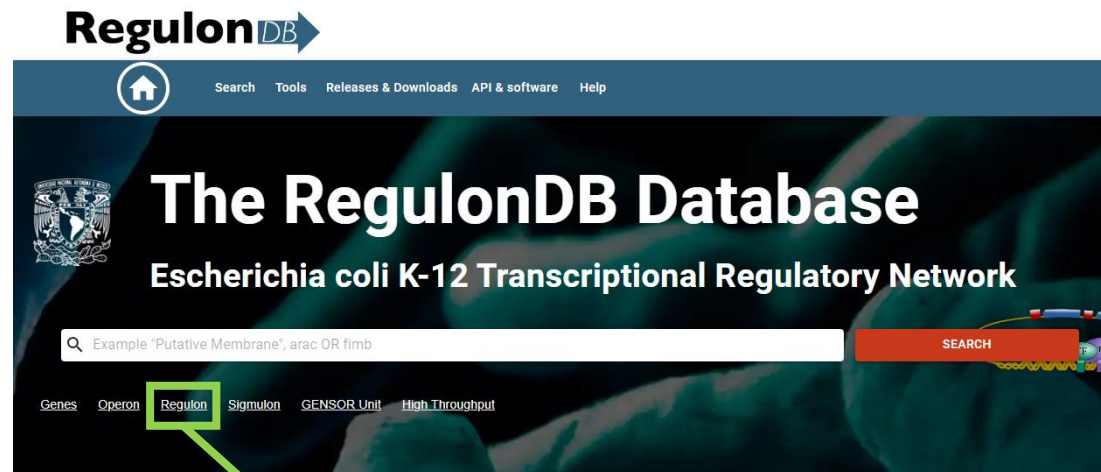
Gruppe 4

Kerem Can Balci, Tim Hendrik Thösen

31.07.2024

Ziel

Regulon

Regularprotein

Gene

DicF	repressor	ftsZ
DicF	repressor	xylR
DicF	repressor	pykA
DicF	repressor	manX

WIKIDATA

Schritt 1 – Daten sammeln

1. Regulon Kennzeichnung

DicF, DsrA, GcvB, McaS, …

+

2. Link zum DB Eintrag

Schritt 1 – Daten sammeln

```python
chrome_options = webdriver.ChromeOptions()
#chrome_options.add_argument("--headless")
chrome_options.add_argument("--no-sandbox")
chrome_options.add_argument('--window-size=1920,1080')
chrome_options.add_experimental_option("prefs", {
    "download.prompt_for_download": False,
    "download.directory_upgrade": True,
    "safebrowsing.enabled": True
})
driver = webdriver.Chrome(options=chrome_options)
actions = ActionChains(driver)

def click_button_wait(XPATH):
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, XPATH))).click()

def find_next_page_button():
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);") # scroll down
    button_xpath = "/html/body/div[1]/div/div[2]/div/div[2]/div/div[3]/div/table/tbody/tr[19]/td/div/div/div/button[3]"
    button = WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, button_xpath)))
    button.click()

# seite aufrufen
driver.get("https://regulondb.ccg.unam.mx/regulon")

# Cookies Rejecten
click_button_wait('/html/body/div[1]/div/div[4]/div/div[3]/button[2]')


is_last_page = False
regulon_links = {}
try:
    while True:
        # Get Table Body
        table_body = WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.XPATH,'/html/body/div[1]/div/di
        # jedes tr (zeile der tabelle) sammeln
        elements = table_body.find_elements(By.XPATH,".//*[contains(@id, 'tr')]")

        for i in range(0, len(elements)):
            try:
                link = elements[i].find_elements(By.XPATH,".//*")[0].find_element(By.XPATH,"div/div/a").get_attribute("hr
                name = elements[i].find_elements(By.XPATH,".//*")[0].find_element(By.XPATH,"div/div/a").get_attribute("te
                regulon_links[name] = link
            except:
                pass

        find_next_page_button()
except:
    print("finished")
```

```
{'DicF': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC00328',
 'DsrA': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC00358',
 'GcvB': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC03275',
 'McaS': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC03411',
 'GadY': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC03413',
 'MicF': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC03508',
 'OxyS': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC03909',
 'ChiX': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04122',
 'SgrS': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04132',
 'IstR-1': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04134'
 'RseX': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04142',
 'RydC': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04143',
 'OhsC': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04144',
 'SymR': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04148',
 'MgrR': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04149',
 'FnrS': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04150',
 'ArrS': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04151',
 'SdsN': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04154',
 'AgrB': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04160',
```

Schritt 2 – Daten abfragen

Tripletts

Schritt 2 – Daten abfragen

Schritt 2 – Daten abfragen

```
url = "https://regulondb.ccg.unam.mx/graphql"
```
→ **Graphql Endpoint**

```
request1["variables"]["advancedSearch"] = "RDBECOLIPDC00328[_id]"
response = json.loads(requests.post(url = url, json = request1, verify = False).content)
```

C:\Anaconda\lib\site-packages\urllib3\connectionpool.py:1061: InsecureRequestWarning: Unverified HTTPS request is being made to host 'regulondb.ccg.unam.mx'. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.html#ssl-warnings
  warnings.warn(

```
response
```

```
{'data': {'getRegulonBy': {'data': [{'_id': 'RDBECOLIPDC00328',
    'aligmentMatrix': {'aligment': None,
      'consensus': None,
      'matrix': None,
      'urlMatrixQualityResult': None,
      'urlPWMLogo': None,
      '__typename': 'AligmentMatrix'},
    'allCitations': [{'evidence': {'_id': 'RDBECOLIEVC00062',
        'additiveEvidenceCodeRule': None,
        'code': 'EXP-IGI',
        'name': 'Inferred from genetic interaction',
        'type': 'W',
        '__typename': 'Evidence'},
      'publication': {'_id': 'RDBECOLIPRC06277',
        'authors': ['Tetart F',
        'Albigot R',
        'Conter A',
        'Mulder E',
        'Bouche JP'],
        'citation': 'Tetart F, Albigot R, Conter A, Mulder E, Bouche JP, 1992, Involvement of FtsZ in coupling of nucleoid separation with septation.',
        'pmid': '1552861',
        'title': 'Involvement of FtsZ in coupling of nucleoid separation with septation.',
```

```
{'DicF': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC00328',
 'DsrA': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC00358',
 'GcvB': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC03275',
 'McaS': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC03411',
 'GadY': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC03413',
 'MicF': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC03508',
 'OxyS': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC03909',
 'ChiX': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04122',
 'SgrS': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04132',
 'IstR-1': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04134',
 'RseX': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04142',
 'RydC': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04143',
 'OhsC': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04144',
 'SymR': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04148',
 'MgrR': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04149',
 'FnrS': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04150',
 'ArrS': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04151',
 'SdsN': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04154',
 'AgrB': 'https://regulondb.ccg.unam.mx/regulon/RDBECOLIPDC04160',
```

Schritt 3 – Daten Preparen

Schritt 4 – Daten hochladen (PyWikiBot)

Genes erzeugen

```
1   i = 0
2   for key, value in list(gene_descriptions.items())[6:]:
3       label = key
4       description = value
5
6       # Create Item
7       data = {
8           'labels': {'en': {'language': 'en', 'value': label}},
9           'descriptions': {'en': {'language': 'en', 'value': str(description)}}
10      }
11      new_item = pywikibot.ItemPage(repo)
12      new_item.editEntity(data, summary=f'Creating new item: {label}')
13
14      # Create Statement
15      claim = pywikibot.Claim(repo, "P31")   # instance of
16      target = pywikibot.ItemPage(repo, "Q7187") # gene
17      claim.setTarget(target)
18      new_item.addClaim(claim, summary='Adding education information')
19
20      gene_ids[key] = new_item.getID()
21      print(key, i, "/", len(gene_descriptions))
22      i += 1
```

```
Sleeping for 9.4 seconds, 2024-07-24 15:43:27

citR 2686 / 2697

Sleeping for 9.3 seconds, 2024-07-24 15:43:38
Sleeping for 9.6 seconds, 2024-07-24 15:43:47

glsB 2687 / 2697

Sleeping for 9.4 seconds, 2024-07-24 15:43:57
Sleeping for 9.3 seconds, 2024-07-24 15:44:08

yneG 2688 / 2697

Sleeping for 9.2 seconds, 2024-07-24 15:44:18
Sleeping for 9.4 seconds, 2024-07-24 15:44:27

sfmC 2689 / 2697

Sleeping for 9.4 seconds, 2024-07-24 15:44:37
Sleeping for 9.5 seconds, 2024-07-24 15:44:47
```

Schritt 4 – Daten hochladen (PyWikiBot)

```python
def generate_claims(regulon, info):
    label = regulon
    if info["beschreibung"] != None:
        description = info["beschreibung"]
    else:
        description = "description: " + regulon

    if len(str(description)) >= 250:
        description = str(description[:249])
    # Create Item
    data = {
        'labels': {'en': {'language': 'en', 'value': label}},
        'descriptions': {'en': {'language': 'en', 'value': str(description)}}
    }
    new_item = pywikibot.ItemPage(repo)
    new_item.editEntity(data, summary=f'Creating new item: {label}')
```

Item wird erzeugt

```python
    claims = []
    # claims für instance of
    instance_of_rna = {'mainsnak': {'snaktype': 'value', 'property': "P31", 'datavalue': {'value': {'entity-type': 'item', 'numeric-id': "11053"}, 'type': 'wikibase-entityid'}}, 'type': 'statement', 'rank': 'normal'}
    claims.append(instance_of_rna)
    instance_of_non_coding_rna = {'mainsnak': {'snaktype': 'value', 'property': "P31", 'datavalue': {'value': {'entity-type': 'item', 'numeric-id': "427087"}, 'type': 'wikibase-entityid'}}, 'type': 'statement', 'rank': 'normal'}
    claims.append(instance_of_non_coding_rna)

    # claims für funktionen der gene (repressor, activator, dual)
    for i in range(0, len(triplets[regulon])):
        value = triplets[regulon][i]
        if value["predikat"] == "repressor":
            repressor = {'mainsnak': {'snaktype': 'value', 'property': "P3776", 'datavalue': {'value': {'entity-type': 'item', 'numeric-id': str(gene_ids[value["objekt"]][1:])}, 'type': 'wikibase-entityid'}}, 'type': 'statement', 'rank': 'normal'}
            claims.append(repressor.copy())
        elif value["predikat"] == "activator":
            activator_of = {'mainsnak': {'snaktype': 'value', 'property': "P3771", 'datavalue': {'value': {'entity-type': 'item', 'numeric-id': str(gene_ids[value["objekt"]][1:])}, 'type': 'wikibase-entityid'}}, 'type': 'statement', 'rank': 'normal'}
            claims.append(activator_of.copy())
        elif value["predikat"] == "dual":
            dual = {'mainsnak': {'snaktype': 'value', 'property': "P1322", 'datavalue': {'value': {'entity-type': 'item', 'numeric-id': str(gene_ids[value["objekt"]][1:])}, 'type': 'wikibase-entityid'}}, 'type': 'statement', 'rank': 'normal'}
            claims.append(dual.copy())

    # claim für found in taxon
    found_in_taxon = {'mainsnak': {'snaktype': 'value', 'property': "P703", 'datavalue': {'value': {'entity-type': 'item', 'numeric-id': 21102938}, 'type': 'wikibase-entityid'}}, 'type': 'statement', 'rank': 'normal'}
    claims.append(found_in_taxon)

    # claims für strand orientation
    if info["left_position"] != None:
        genomic_start = {'mainsnak': {'snaktype': 'value', 'property': "P644", 'datavalue': {'value': str(info["left_position"]), 'type': 'string'}, 'type': 'wikibase-entityid'}, 'type': 'statement', 'rank': 'normal'}
        claims.append(genomic_start)
    if info["right_position"] != None:
        genomic_end = {'mainsnak': {'snaktype': 'value', 'property': "P645", 'datavalue': {'value': str(info["right_position"]), 'type': 'string'}, 'type': 'wikibase-entityid'}, 'type': 'statement', 'rank': 'normal'}
        claims.append(genomic_end)
    if info["strand_orientation"] != None and info["left_position"] != None and info["right_position"] != None:
        switch = {"22809711" : "22809680", "22809680" : "22809711"}
        so = switch[str(info["strand_orientation"][1:])]
        strand_orientation = {'mainsnak': {'snaktype': 'value', 'property': "P2548", 'datavalue': {'value': {'entity-type': 'item', 'numeric-id': so}, 'type': 'wikibase-entityid'}}, 'type': 'statement', 'rank': 'normal'}
        claims.append(strand_orientation)

    # upload
    data = {'claims': claims}
    new_item.editEntity(data, summary='Adding multiple claims in one request')
    return new_item.getID()
```

Ergebnis