



Міністерство освіти і науки України

Національний технічний університет України  
“Київський політехнічний інститут імені Ігоря Сікорського”

## **ЛАБОРАТОРНІ РОБОТИ З АНАЛІЗУ ДАНИХ**

**Виконала:**

студентка групи ФІ-73

Звичайна Анастасія Олександрівна

залікова книжка №04

## ЛАБОРАТОРНА РОБОТА №1

### Завдання до лабораторної роботи №1:

- Розширити список rss-каналів каналами комп'ютерної і телекомунікаційної спрямованості (але не торговельними майданчиками).
- Створити процедуру (скрипт) для періодичного скачування інформації із створеного переліку RSS-фідів.
- Реалізувати процедуру створення файлу, в якому об'єднуються всі скачані RSS-фіди, і підключити її до скрипту скачування.

### Виконання роботи:

Робота виконувалась у Google Colab, а результуючі файли зберігались на Google Drive. Всі коментарі і виводи на екран можна побачити у наступних рисунках та програмі:

```
[ ] pip install feedparser

Collecting feedparser
  Downloading https://files.pythonhosted.org/packages/1c/21/faf1bac028662cc8adb2b5ef7a6f3999a765baa2835331df365289b0ca56/feedparser-6.0.2-py3-none-any.whl (81kB) 3.5MB/s
Collecting sgmlib3k
  Downloading https://files.pythonhosted.org/packages/9e/bd/3704a8c3e0942d711c1299ebf7b9091930adae6675d7c8f476a7ce48653c/sgmlib3k-1.0.0.tar.gz
Building wheels for collected packages: sgmlib3k
  Building wheel for sgmlib3k (setup.py) ... done
  Created wheel for sgmlib3k: filename=sgmlib3k-1.0.0-cp37-none-any.whl size=6067 sha256=250145b42313b9039843d3e225bcff52783ad970a767600e31b9e43db48f61
  Stored in directory: /root/.cache/pip/wheels/f1/80/5a/444ba08a550cdd241bd9baf8bae44be750efe370adb944506a
Successfully built sgmlib3k
Installing collected packages: sgmlib3k, feedparser
Successfully installed feedparser-6.0.2 sgmlib3k-1.0.0
```

```
import feedparser
```

```
# 'https://www.jpl.nasa.gov/multimedia/rss/news.xml', 'https://blog.wolfram.com/category/mathematics/feed/', 'https://blog.wolfram.com/category/mathematics/feed/',
# 'https://www.newscientist.com/feed/home/?cmpid=RSS%7CNSNS-Home', 'https://pcnews.ru/feeds/latest/news/',
# 'https://jeremykun.com/feed/', 'https://codingnconcepts.com/index.xml', 'https://nakedsecurity.sophos.com/feed/',
# 'https://www.helpnetsecurity.com/feed', 'https://pcnews.ru/feeds/latest/articles', 'https://mobile-review.com/news/feed'
```

```
rss_list = [
```

```
    'https://www.overclockers.ua/rss.xml',
    'https://habr.com/ru/rss/all/all/?fl=ru',
    'https://mobile-review.com/news/feed',
    'https://techtoday.in.ua/feed'
```

```
]
```

```
# Приклад відсутності published_parsed
url1="https://pcnews.ru/feeds/latest/news/"
d=feedparser.parse(url1)
print(d)

{'bozo': False, 'entries': [{'title': 'Экспедиция, организованная XPeng, призвана показать возможности системы самоуправляемого вождения NGP', 'title_detail': {'type': 'text/plain',
```

```
[ ] # Створюємо список rss-фідів та виводимо titles, обмеження на час публікації 3 дні:
import time
for index, rss_type in enumerate(rss_list):
    print(f'{index}. {rss_type}')
    d = feedparser.parse(rss_type)
    for index2, entry in enumerate(d.entries):
        if time.time() - time.mktime(entry.published_parsed) < (86400*3):
            print(f'{index}. {index2}. {entry.title}')
            print('\n')
```

```
0. https://www.overclockers.ua/rss.xml
0.0. В этом году GlobalFoundries инвестирует $1,4 млрд в расширение производства

0.1. EKWB предлагает водоблок линейки EK-Classic для старших карт Radeon RX 6000

0.2. SK Hynix наладила серийный выпуск 18-гигабайтных чипов LPDDR5

0.3. MSI обеспечит поддержку Resizable BAR на всех платах LGA1151-v2

0.4. Скорость передачи информации по Thunderbolt 5 достигнет 80 Гбит/с
```

```
[ ] feedparser.parse

<function feedparser.api.parse>
```

```
# Перетворюємо список rss-фідів у DataFrame
```

```
import pandas as pd
```

```

data={}
data.setdefault('rss_link', [])
data.setdefault('title', [])
data.setdefault('site_link', [])

for index, rss_type in enumerate(rss_list):
    print(f'{index}. {rss_type}')
    d = feedparser.parse(rss_type)
    for index2, entry in enumerate(d.entries):
        if (time.time() - time.mktime(entry.published_parsed) < (86400*3)): # Встановлюємо обмеження на час публікації
            data['rss_link'].append(rss_list[index])
            data['title'].append(entry.title)
            data['site_link'].append(entry.link)

data=pd.DataFrame(data)
data.head()

```

```

0. https://www.overclockers.ua/rss.xml
1. https://habr.com/ru/rss/all/all/?fl=ru
2. https://mobile-review.com/news/feed/
3. https://techtoday.in.ua/feed

```

	rss_link	title	site_link
0	<a href="https://www.overclockers.ua/rss.xml">https://www.overclockers.ua/rss.xml</a>	В этом году GlobalFoundries инвестирует \$1,4 м...	<a href="https://www.overclockers.ua/news/hardware/2021...">https://www.overclockers.ua/news/hardware/2021...</a>
1	<a href="https://www.overclockers.ua/rss.xml">https://www.overclockers.ua/rss.xml</a>	EKWB предлагает водоблок линейки EK-Classic дл...	<a href="https://www.overclockers.ua/news/hardware/2021...">https://www.overclockers.ua/news/hardware/2021...</a>
2	<a href="https://www.overclockers.ua/rss.xml">https://www.overclockers.ua/rss.xml</a>	SK Hynix наладила серийный выпуск 18-гигабайтн...	<a href="https://www.overclockers.ua/news/hardware/2021...">https://www.overclockers.ua/news/hardware/2021...</a>
3	<a href="https://www.overclockers.ua/rss.xml">https://www.overclockers.ua/rss.xml</a>	MSI обеспечит поддержку Resizable BAR на всех ...	<a href="https://www.overclockers.ua/news/hardware/2021...">https://www.overclockers.ua/news/hardware/2021...</a>
4	<a href="https://www.overclockers.ua/rss.xml">https://www.overclockers.ua/rss.xml</a>	Скорость передачи информации по Thunderbolt 5 ...	<a href="https://www.overclockers.ua/news/hardware/2021...">https://www.overclockers.ua/news/hardware/2021...</a>

```

# Вихідний DataFrame зберігаємо на гугл-диск
from google.colab import drive
drive.mount('/drive', force_remount=True)
data.to_csv('/drive/My Drive/Colab Notebooks/Lab1.csv')

```

Mounted at /drive

## ЛАБОРАТОРНА РОБОТА №2

### Завдання до лабораторної роботи №2:

- Формування JSON-файлу, придатного для завантаження в систему ElasticSearch. Реалізувати процедуру конвертування RSS-фідів в формат JSON мовами Python або R.

### Виконання роботи:

Дану лабораторну роботу я виконала на Google Colab. Спочатку встановила потрібні бібліотеки, а потім зробила придатний для завантажування у систему ElasticSearch файл.

```
[ ] pip install xmldict
```

```

Collecting xmldict
  Downloading https://files.pythonhosted.org/packages/28/fd/30d5c1d3ac29ce229f6bdc40bbc20b28f716e8b363140c26eff19122d8a5/xmldict-0.12.0-py2.py3-none-any.whl
Installing collected packages: xmldict
Successfully installed xmldict-0.12.0

```

```
[ ] pip install requests
```

```

Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (2.23.0)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests) (2020.12.5)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests) (3.0.4)

```

```
[ ] pip install python-dateutil
```

```

Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-packages (2.8.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil) (1.15.0)

```

```

import xmltodict
import json
from datetime import datetime
import dateutil.parser as parser
import requests

class RssEntry:
    def __init__(self, source, title, body, pubDate, url):
        self.source = source
        self.title = title
        self.body = body
        self.pubDate = pubDate
        self.url = url

with open("./lab2.json", "w") as f:
    f.write("\n")
    for i, rss_url in enumerate(rss_list):
        data = requests.get(rss_url)
        json_doc = xmltodict.parse(data.text)

        source = json_doc["rss"]["channel"]["title"]
        entries = json_doc["rss"]["channel"]["item"]
        for j, e in enumerate(entries):
            date_time_str = parser.parse(e["pubDate"]).isoformat()

            rss_entry = RssEntry(
                source,
                e["title"],
                e["description"],
                date_time_str,
                e["link"]
            )
            json_str = json.dumps(rss_entry.__dict__, indent=2, ensure_ascii=False)
            f.write(json_str)
            if j != len(entries)-1 or i != len(rss_list)-1:
                f.write(",")
                f.write("\n")

    f.write("]")

```

### ЛАБОРАТОРНА РОБОТА №3

#### Завдання до лабораторної роботи №3:

- Встановлення системи Elasticsearch. Завантаження інформації Elasticsearch в базу даних із JSON-файлу.

#### Виконання роботи:

Дану лабораторну роботу спочатку я виконала на Google Colab, а потім локально на комп'ютері.

##### 1) Виконання на Google Colab:

Систему Elasticsearch я встановила у Google Colab наступним чином:

```

# install es server
!apt install default-jdk > /dev/null
!wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.5.4.tar.gz -q --show-progress
!tar -xzf elasticsearch-6.5.4.tar.gz

```

```
!chown -R daemon:daemon elasticsearch-6.5.4
# start server
import os
from subprocess import Popen, PIPE, STDOUT
es_server = Popen(['elasticsearch-6.5.4/bin/elasticsearch'],
                  stdout=PIPE, stderr=STDOUT,
                  preexec_fn=lambda: os.setuid(1) # as daemon
                  )
```

# client-side

```
!pip install elasticsearch -q
from elasticsearch import Elasticsearch
es = Elasticsearch()
es.ping() # got True
```

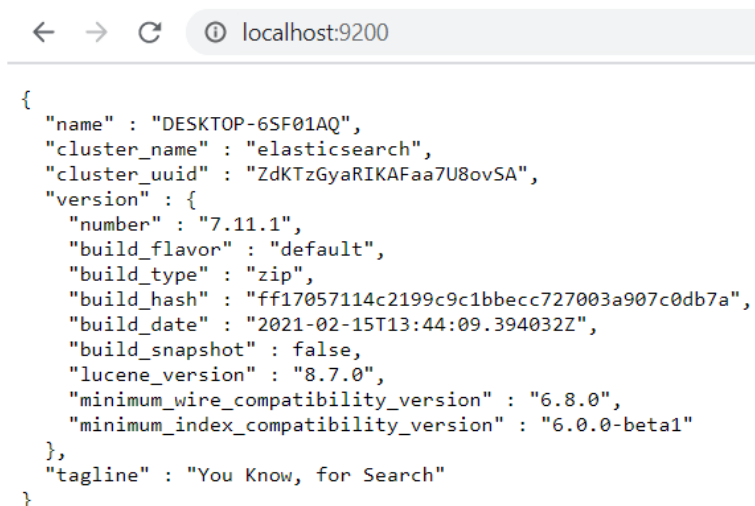
Для завантаження у систему ElasticSearch я зробила наступне:

# Index data from a JSON file in ElasticSearch

```
with open('./lab2.json') as json_file:
    data = json.load(json_file)
    for i, e in enumerate(data):
        es.index(index='myindex', ignore=400, doc_type='docket', id=i, body=e)
```

2) Виконання локально на комп'ютері:

Систему ElasticSearch я встановила за інструкцією за наступним посиланням [www.elastic.co/downloads/elasticsearch](http://www.elastic.co/downloads/elasticsearch). Підтвердження роботи ElasticSearch можна спостерігати на наступному рисунку:



```
{
  "name" : "DESKTOP-6SF01AQ",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "ZdKTzGyaRIKAFaa7U8ovSA",
  "version" : {
    "number" : "7.11.1",
    "build_flavor" : "default",
    "build_type" : "zip",
    "build_hash" : "ff17057114c2199c9c1bbecc727003a907c0db7a",
    "build_date" : "2021-02-15T13:44:09.394032Z",
    "build_snapshot" : false,
    "lucene_version" : "8.7.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

Завантаження у систему ElasticSearch на локально я зробила аналогічно як і на Google Colab.

## ЛАБОРАТОРНА РОБОТА №4

### Завдання до лабораторної роботи №4:

- Забезпечити функціонування на комп'ютері (сервері) засобів інтерактивної взаємодії з користувачем (наприклад, CGI).
- Ознайомитися із основними можливостями пошуку і фільтрації в Elasticsearch.
- Ознайомитись з основами HTML-розмітки.

### Виконання роботи:

Дану лабораторну роботу я виконала локально на комп'ютері.

```
# Perform search operation
from pprint import pprint
res = es.search(index="myindex", body={"query": {"match": {"body": "крипто"}}})
pprint(res)
```

```
res = es.search(index="myindex", body={"query": {"match": {"body": "GeForce"}}})
pprint(res)
```

```
{'_shards': {'failed': 0, 'skipped': 0, 'successful': 5, 'total': 5},
 'hits': {'hits': [{'_id': '123',
                    '_index': 'myindex',
                    '_score': 3.0150714,
                    '_source': {'body': 'Представитель Nvidia рассказал '
                                       'веб-изданию Gучи3D, как именно '
                                       'реализован '
                                       '&laquo;крипто-ограничитель&raquo; для '
                                       'GeForce RTX 3060. По его словам, '
                                       '&laquo;между драйвером, чипом RTX '
                                       '3060 и BIOS (прошивкой) существует '
                                       'безопасное соединение, которое '
                                       'препятствует снятию ограничителя '
                                       'хешрейта&raquo;. Иными...',
                                       'pubDate': '2021-02-19T08:53:00+02:00',
                                       'source': 'Overclockers.ua / Новости и обзоры',
                                       'title': 'Nvidia уверена, что '
                                               '«крипто-ограничитель» GeForce RTX '
                                               '3060 нельзя взломать (обновлено)',
                                       'url': 'https://www.overclockers.ua/news/hardware/2021-02-19/128630/'},
                    '_type': 'docket'}]},
 'max_score': 3.0150714,
 'total': 1},
 'timed_out': False,
 'took': 8}
```

Для інтерактивної взаємодії з користувачем я використовувала веб-фреймворк Flask, мову Python та мову розмітки html. Для цього я створила файли application.py та index.html. Також використала командний рядок для того, щоб встановити flask та виконати програму наступним чином:

```
set FLASK_APP=application.py
set FLASK_ENV=development
flask run
```

#### **application.py**

```
from flask import Flask, render_template, request
import json
```

```
app = Flask(__name__)
```

```
from elasticsearch import Elasticsearch
es = Elasticsearch()
```

```
# Index data from a JSON file in ElasticSearch
with open('./lab2.json') as json_file:
    data = json.load(json_file)
    for i, e in enumerate(data):
        es.index(index='myindex', ignore=400, doc_type='docket', id=i, body=e)
```

```
def empty(s):
    if s is None:
        return True
    if len(s) == 0:
        return True
    else:
        return False
```

```
@app.route('/')
def index():
```

```
    query = request.args.get('q')
    source = request.args.get('src')
```

```
    total_rss_entries = None
    rss_entries = []
    if not empty(query) and not empty(source):
        res = es.search(index="myindex", body={
            "query": {
                "bool": {
                    "must": [
```

```

        {
            "multi_match": {
                "query": query,
                "fields": ["title", "body"]
            }
        }
    ],
    "filter": [
        {
            "match": {
                "source": {
                    "query": source,
                    "fuzziness": "auto"
                }
            }
        }
    ]
}
    }
    }
    }
    total_rss_entries = res["hits"]["total"]
    results = res["hits"]["hits"]

    for res in results:
        rss_entries.append({
            'title': res['_source']['title'],
            'text': res['_source']['body'],
            'url': res['_source']['url']
        })

    return render_template('index.html',
        query=query,
        source=source,
        total_rss_entries=total_rss_entries,
        rss_entries=rss_entries)

```

## index.html

```

<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Lab4 | Search RSS feeds</title>
</head>

<body>

    <form action="/" method="GET">
        Запит : <input type="text" name="q" value="{{ query | default("", true) }}" /><br/>
        Джерело: <input type="text" name="src" value="{{ source | default("", true) }}" /><br/>
        <input type="submit" name="submit" value="Старт"/>
    </form>

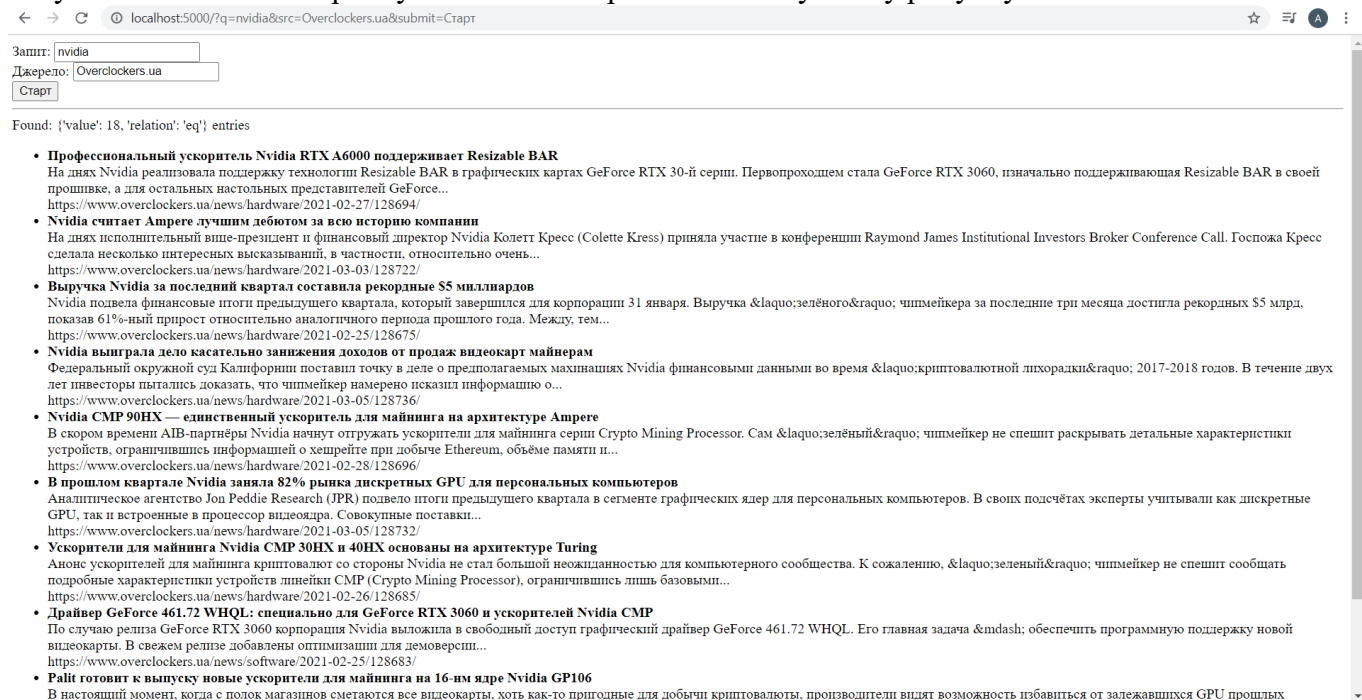
    {% if total_rss_entries %}
    <hr/>
    Found: {{ total_rss_entries }} entries
    <ul>
        {% for entry in rss_entries %}
        <li>
            <b>{{ entry.title }}</b><br />
            {{ entry.text }}<br />
            {{ entry.url }}<br />
        </li>
        {% endfor %}
    </ul>
    {% endif %}

</body>
</html>

```



## Результат виконання скрипту можна спостерігати на наступному рисунку:



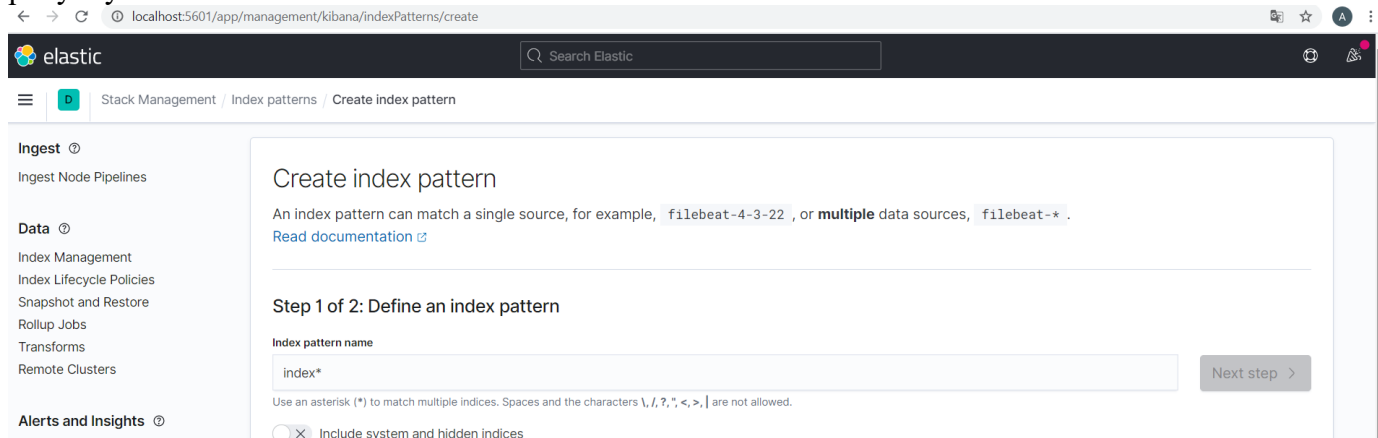
## ЛАБОРАТОРНА РОБОТА №5

### Завдання до лабораторної роботи №5:

- Забезпечити функціонування на комп'ютері системи Kibana.
- Детально ознайомитись із можливостями візуалізації в системі Kibana.
- Побудувати часові діаграми кількості публікацій із різних RSS-джерел

### Виконання роботи:

Систему Kibana я встановила на свій локальний комп'ютер за інструкцією за наступним посиланням [www.elastic.co/downloads/kibana](http://www.elastic.co/downloads/kibana). Підтвердження роботи Kibana можна спостерігати на наступному рисунку:



Ознайомлюватись з можливостями візуалізації у системі Kibana я почала з того, що передала дані у Elasticsearch наступним чином:

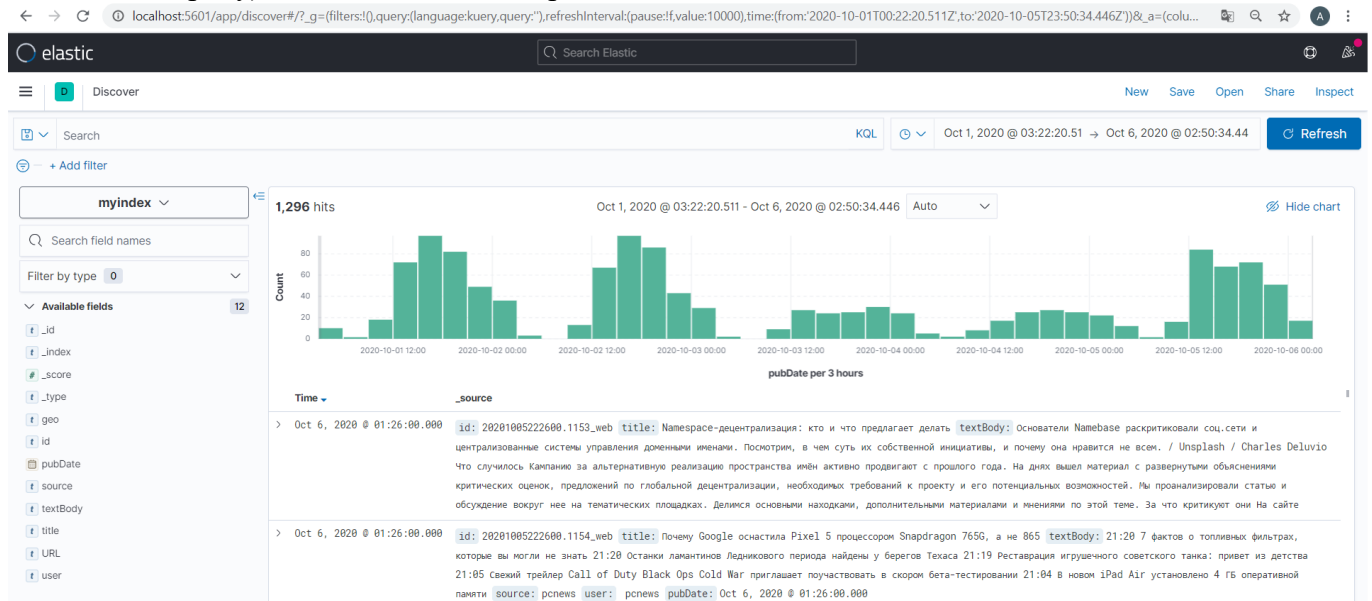
```
import json

from elasticsearch import Elasticsearch
es = Elasticsearch()

# Index data from a JSON file in Elasticsearch
with open('./lab5.json', encoding="utf-8") as json_file:
    data = json.load(json_file)
    for i, e in enumerate(data):
        es.index(index='myindex', ignore=400, doc_type='docket', id=i, body=e)
    print('percentage: ', '{:.5f}'.format(i*100/len(data)), '%')
```

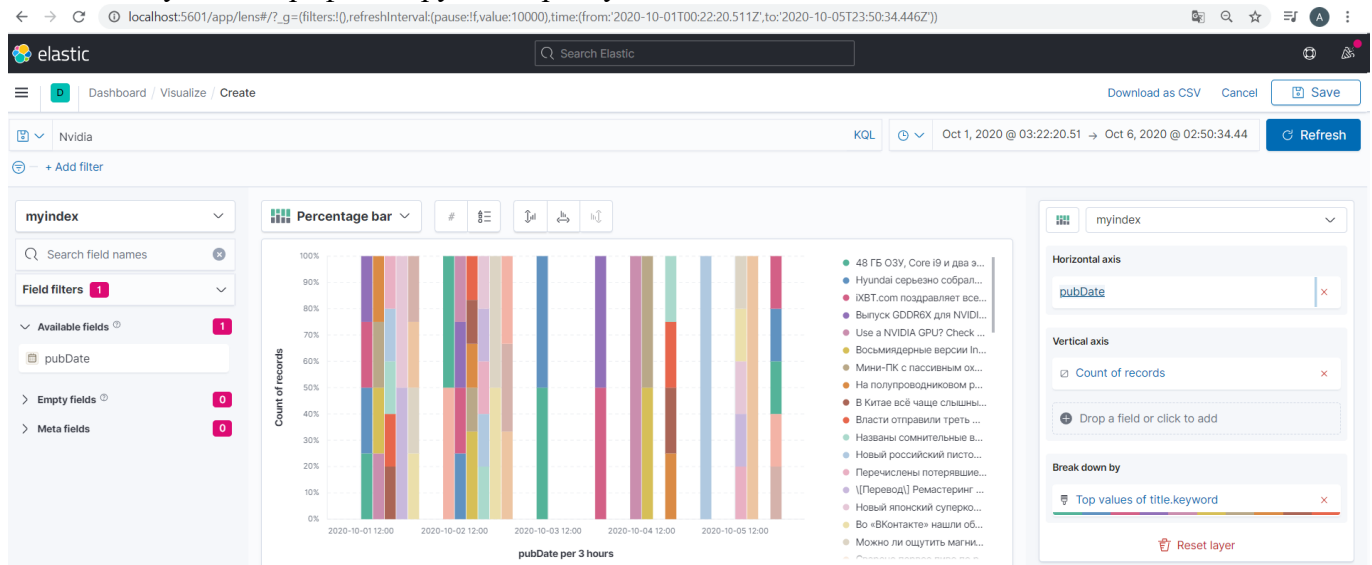


Після цього створила індекс myindex та переглянула дані у Discover (див. рисунок нижче). Через те, що розмір даних є дуже великим, вдалось завантажити лише частину даних (з 1 жовтня 2020 року до 6 жовтня 2020 року). З цими даними я і працювала далі.

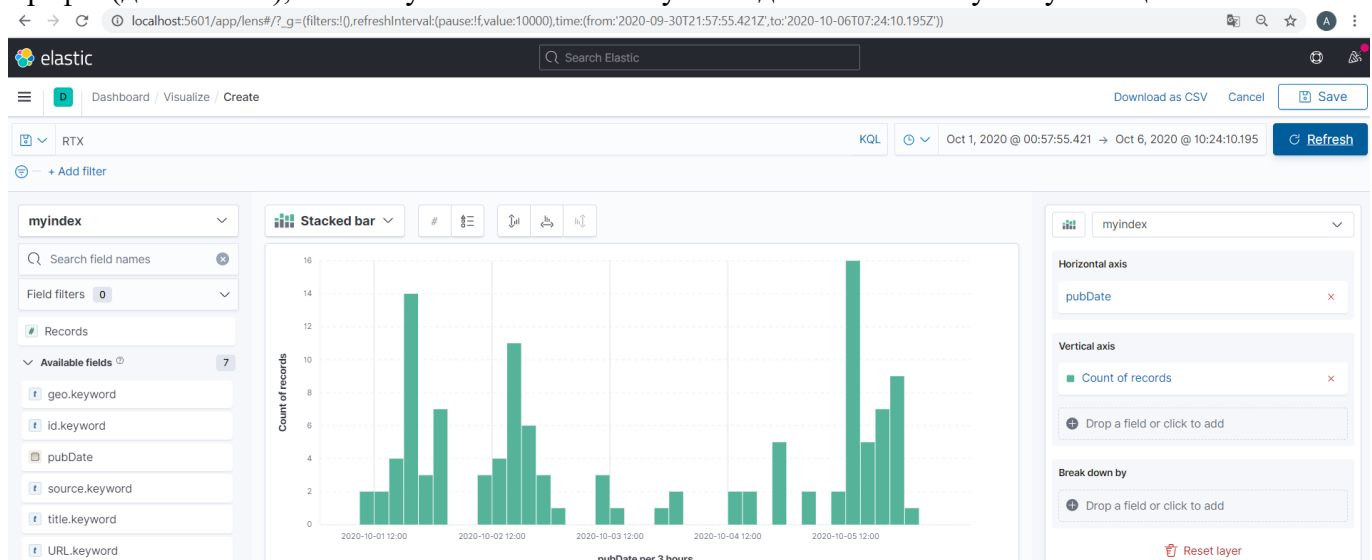


Під час роботи з Kibana я побудувала різні графіки.

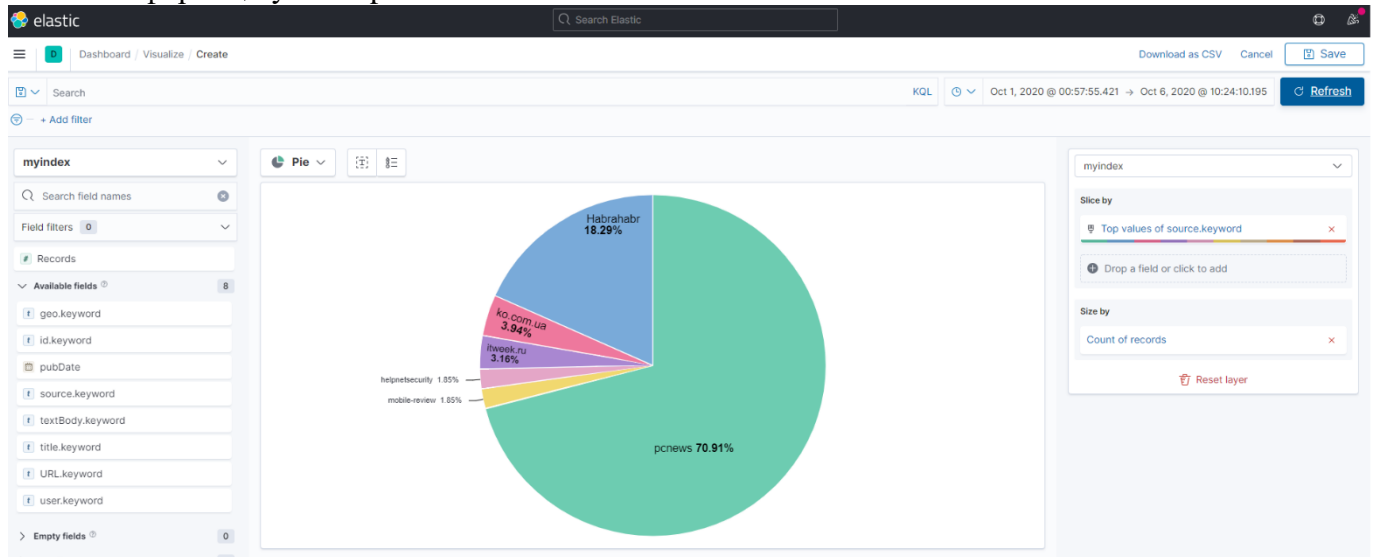
Графік (див. нижче), на якому зображені самі публікації (їх назви) за днями тижня. Публікацій не так багато, тому таким графіком зручно користуватись.



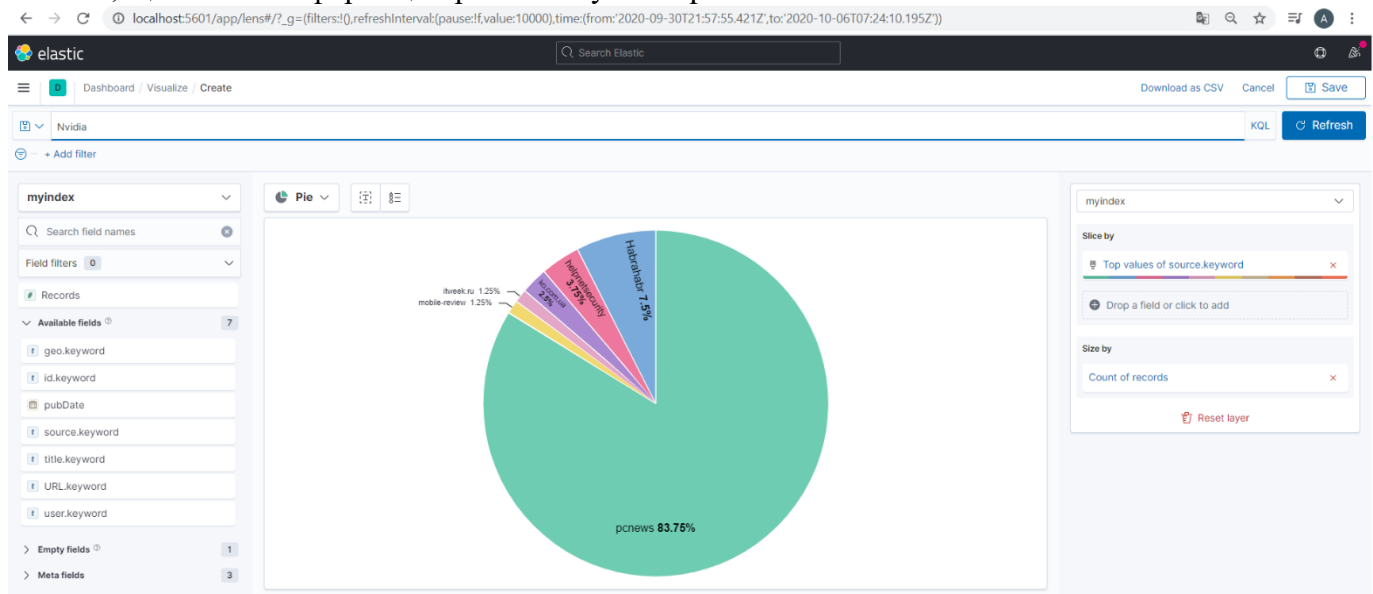
Графік (див. нижче), на якому можна побачити у який день і скільки було публікацій.



Графік (див. нижче), на якому можна побачити з яких джерел інформації у нас найбільше. Бачимо, що 70.91% інформації у нас з pcnews.



Графік (див. нижче), на якому можна побачити з яких джерел інформації про Nvidia у нас найбільше. Бачимо, що 83.75% інформації про Nvidia у нас з pcnews.



## ЛАБОРАТОРНА РОБОТА №6

### Завдання до лабораторної роботи №6:

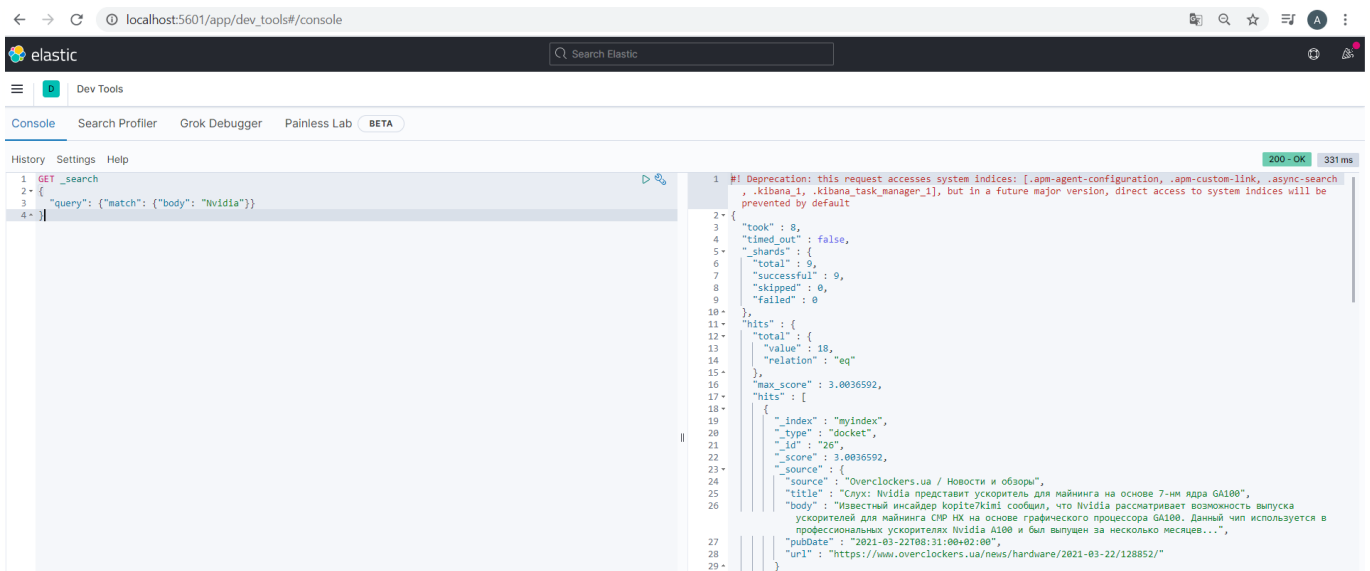
- Завантажити в базу даних системи Elasticsearch тестову базу даних із накопиченими даними з RSS-фідів з мережі Інтернет.
- Самостійно здійснити агрегацію цих даних за полем дата-час.
- Завантажити отримані результати у доступну систему типу DSP (Digital Signal Processing).

### Виконання роботи:

Спочатку я зробила запит у систему на пошук у ній джерел, що стосуються Nvidia, використовуючи дані, що вже були завантажені у систему. Я це зробила для того, щоб переконатись, що на початковому етапі все йде добре. На рисунку нижче наведений результат такого запиту.

GET \_search

```
{
  "query": {
    "match": { "body": "Nvidia" }
  }
}
```



Розуміючи те, що треба буде будувати тренд часового ряду, я намагалась завантажити максимальну кількість даних, які дозволяв мій комп'ютер. Для того, щоб відслідковувати кількість інформації, яку ще треба завантажити, я написала програму, яка виводить на екран процент завантаженої від загальної кількості інформації, яку нам запропонували використовувати на 250МБ, у відсотках.

```
import json
```

```
from elasticsearch import Elasticsearch
es = Elasticsearch()
```

```
# Index data from a JSON file in ElasticSearch
with open('./lab5.json', encoding = "utf-8") as json_file:
    data = json.load(json_file)
    for i, e in enumerate(data):
        es.index(index='myindex', ignore=400, doc_type='docket', id=i, body=e)
        print('percentage: ', '{:.5f}'.format(i*100/len(data)), '%')
```

Залишалось лише чекати на те, коли вся інформація завантажиться чи мій комп'ютер не почне протестувати від перенапруги, адже сама Kibana дуже навантажує процесор, відеокарту і файл на 250МБ є відносно великим.



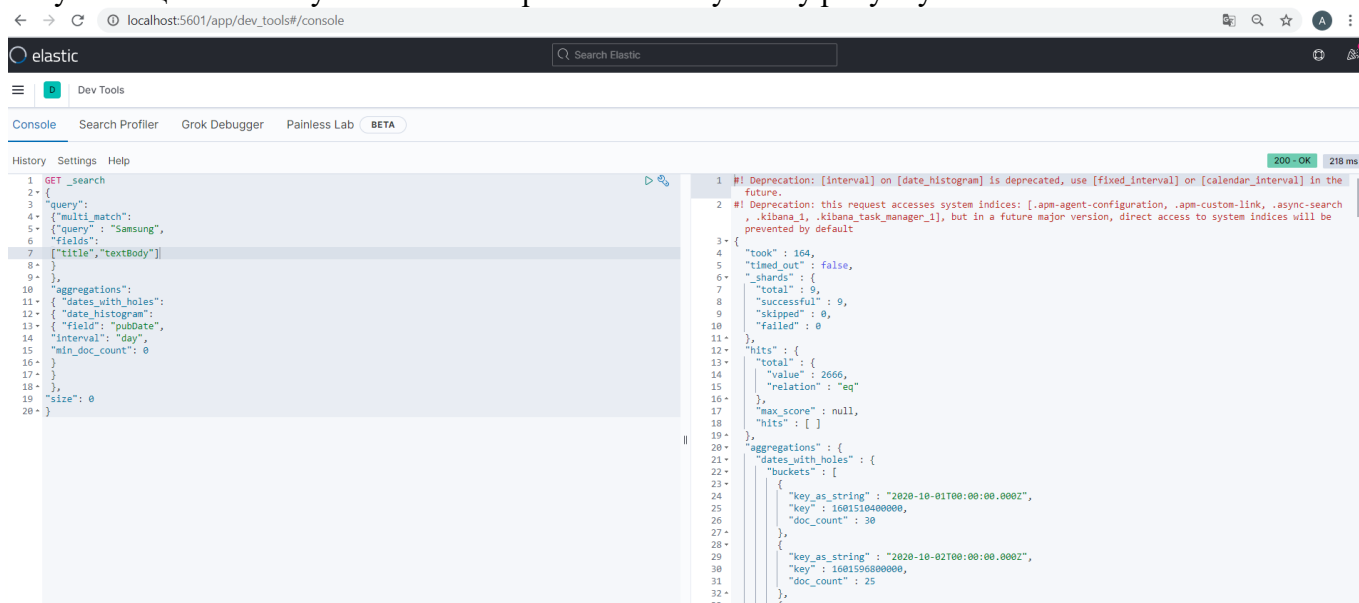
Мій комп'ютер став протестувати на 63%. Завантаження великої кількості даних не є швидким процесом, тому я працювала вже з тими даними, що вже завантажились. Тобто з даними розміром 157.5МБ. Це були дані за час з 1 жовтня 2020 року до 1 січня 2021 року.

Я зробила наступний запит у самому інтерфейсі Kibana:

GET \_search

```
{
"query":
{"multi_match":
{"query" : "Samsung",
"fields":
["title","textBody"]
}
},
"aggregations":
{ "dates_with_holes":
{ "date_histogram":
{ "field": "pubDate",
"interval": "day",
"min_doc_count": 0
}
}
},
"size": 0
}
```

Результат цього запиту можна спостерігати на наступному рисунку:



Далі я скопіювала у окремий файл інформацію, що йшла під buckets. Вона мала наступний вигляд:

```
[
{
  "key_as_string": "2020-10-01T00:00:00.000Z",
  "key": "1601510400000",
  "doc_count": 30
},
{
  "key_as_string": "2020-10-02T00:00:00.000Z",
  "key": "1601596800000",
  "doc_count": 25
},
{
  "key_as_string": "2020-10-03T00:00:00.000Z",
  "key": "1601683200000",

```

```

        "doc_count" : 18
    },
    {
        "key_as_string" : "2020-10-04T00:00:00.000Z",
        "key" : 1601769600000,
        "doc_count" : 6
    },
    ...
]

```

Відповідний їй файл я назвала data\_for\_lab6.json. Далі стояла задача переведення цього файлу у формат csv таким чином, щоб залишились лише поля "key\_as\_string" та "doc\_count". Також поле "key\_as\_string" за завданням потрібно було модифікувати: замість написання дати і часу, треба було лишити тільки дату. Наприклад, замість "2020-10-04T00:00:00.000Z" треба "2020-10-04". Для цього я написала наступну програму:

```

import json
import csv

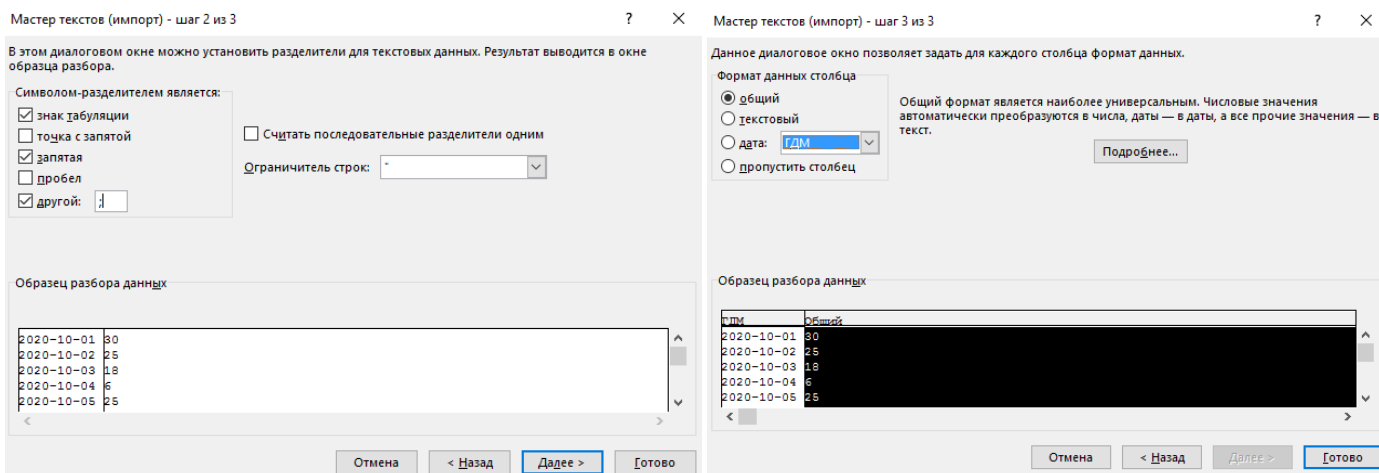
def list_to_string(s):
    result = ""
    for elem in s:
        result += elem
    return result

data = None
with open("./data_for_lab6.json") as json_file:
    data = json.load(json_file);

if data is not None:
    with open("./result_of_lab6.csv", "w") as csv_file:
        w = csv.writer(csv_file, delimiter=';')
        for i, e in enumerate(data):
            w.writerow([list_to_string(list(e["key_as_string"])[-14]), e["doc_count"]])

```

У результаті я отримувала файл result\_of\_lab6.csv, який завантажувала у Excel.



Використовуючи точкову діаграму та лінію тренду я побудувала графік, на якому за горизонтальною віссю відкладені дати, а за вертикальною – кількість повідомлень. Найкращого результату можна було досягти при апроксимації поліномом 6-ї степені. Однак, можна побачити, що достовірність апроксимації  $R^2=0.0279$ , що говорить про те, що прогноз не є точним і потрібно більше інформації використовувати. 157.5МБ мало для того, щоб зробити гарну апроксимацію.



## ЛАБОРАТОРНА РОБОТА №7

### Завдання до лабораторної роботи №7:

- Встановити на комп'ютері систему Python з підтримкою графічної бібліотеки matplotlib.
- Детально ознайомитись із можливостями бібліотек matplotlib і ruwt.
- Побудувати вейвлет-скейлограму з іншою базовою вейвлет-функцією. Дослідити різницю вейвлет-скейлограм при різних базових вейвлет-функціях

### Виконання роботи:

У даній роботі я скористалась Google Colab, адже це є дуже зручний для мене сервіс. В його основі лежить Jupyter Notebook.

Спочатку я для своїх даних зробила віконне згладжування за аналогією з методички:

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
from matplotlib.colors import LogNorm
```

```
D=[30,25,18,6,25,21,49,38,44,13,17,27,25,22,31,47,21,16,31,22,31,15,30,14,25,46,37,33,28,40,15,14,40,34,52,37,18,27,12,33,31,30,15,31,6,21,34,35,32,31,35,16,7,39,36,38,32,32,13,20,32,47,14,29,34,22,12,21,30,40,35,46,35,9,26,39,31,44,57,14,30,51,56,15,41,35,21,5,49,51,37,15]
```

```
M=47
```

```
N=len(D)
```

```
Z=np.random.rand(M,N)
```

```
C=D
```

```
for i in range(N):
```

```
    for j in range(M):
```

```
        Z[j][i]=D[i]
```

```
for j in range(M):
```

```
    for k in range(j,N-j):
```

```
        Z[j][k]=0;
```

```
        for l in range(1,j):
```

```
            Z[j][k]=Z[j][k]+D[k-l]
```

```
            Z[j][k]=Z[j][k]+D[k+l]
```

```
        Z[j][k]=Z[j][k]-D[k]
```

```
        Z[j][k]=Z[j][k]/(2*j+1)
```

```
fig,(ax0,ax1,ax2)=plt.subplots(3,1)
```

```
ax0.plot(D)
```

```
ax0.set_title('Рівні згладжування часового ряду')
```

```
fig.tight_layout()
```

```
for i in range(N):
```

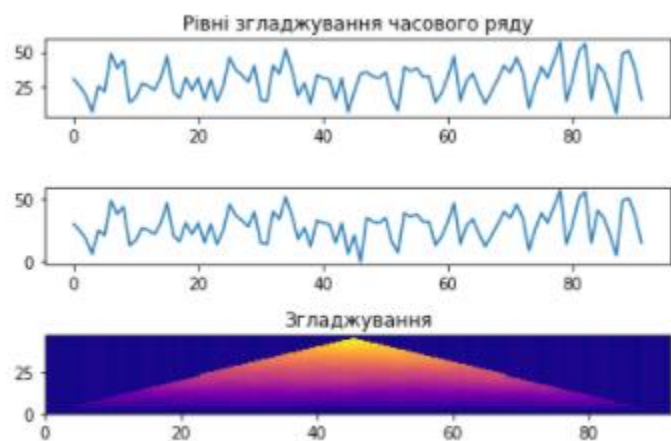
```
    C[i]=Z[46][i]
```

```
ax1.plot(C)
```

```
fig.tight_layout()
```

```
ax2.pcolor(Z,cmap='plasma')
```

У результаті було отримано наступні графіки:



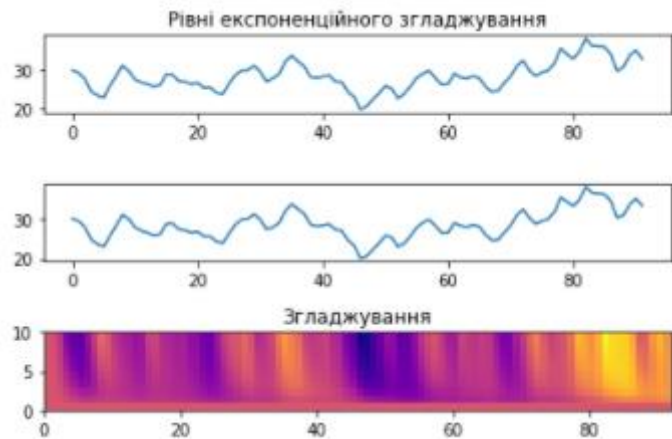
```

ax2.set_title('Згладжування')
fig.tight_layout()
plt.show()
Зрозуміло, що через те, що спостережень мало, і вони розкидані на відносно невеликому проміжку,
то згладжування нам не дає особливого ефекту, але все ж таки можемо так зробити. Аналогічна
ситуація спостерігалась і у випадку експоненційного згладжування.
M=10
N=len(D)
Z=np.random.rand(M,N)
C=D
for i in range(N):
    for j in range(M):
        Z[j][i]=D[i]

for j in range(M):
    alf=j/M
    for k in range(1,N):
        Z[j][k]=D[k]*alf+Z[j][k-1]*(1-alf);
fig,(ax0,ax1,ax2)=plt.subplots(3,1)
ax0.plot(D)
ax0.set_title('Рівні експоненційного згладжування')
fig.tight_layout()
for i in range(N):
    C[i]=Z[9][i]
ax1.plot(C)
fig.tight_layout()
ax2.pcolor(Z,cmap='plasma')
ax2.set_title('Згладжування')
fig.tight_layout()
plt.show()

```

У результаті було отримано наступні графіки:



Після цього я почала будувати вейвлет-скейлограму. Для цього я використала вейвлет-функції:

- мексиканський капелюх (mexh)

$$\psi(t) = \frac{2}{\sqrt{3}\sqrt{\pi}} \exp^{-\frac{t^2}{2}} (1 - t^2)$$

- Морле (morl)

$$\psi(t) = \exp^{-\frac{t^2}{2}} \cos(5t)$$

- Гауса (gaus8)

$$\psi(t) = C \exp^{-t^2}$$

```

import matplotlib.pyplot as plt
import pywt
f_s = 100 # Sampling rate
x=[30,25,18,6,25,21,49,38,44,13,17,27,25,22,31,47,21,16,31,22,31,15,30,14,25,46,37,33,28,40,15,14,40,34,52,37,18,27,12,33,31,
30,15,31,6,21,34,35,32,31,35,16,7,39,36,38,32,32,13,20,32,47,14,29,34,22,12,21,30,40,35,46,35,9,26,39,31,44,57,14,30,51,56,15,
41,35,21,5,49,51,37,15]
N=len(x)
t= range(N)
# Visualization
fig, (ax1, ax2, ax3, ax4) = plt.subplots(4,1, sharex = True, figsize = (10,8))
fig.subplots_adjust(bottom = -0.4)
# Signal

```

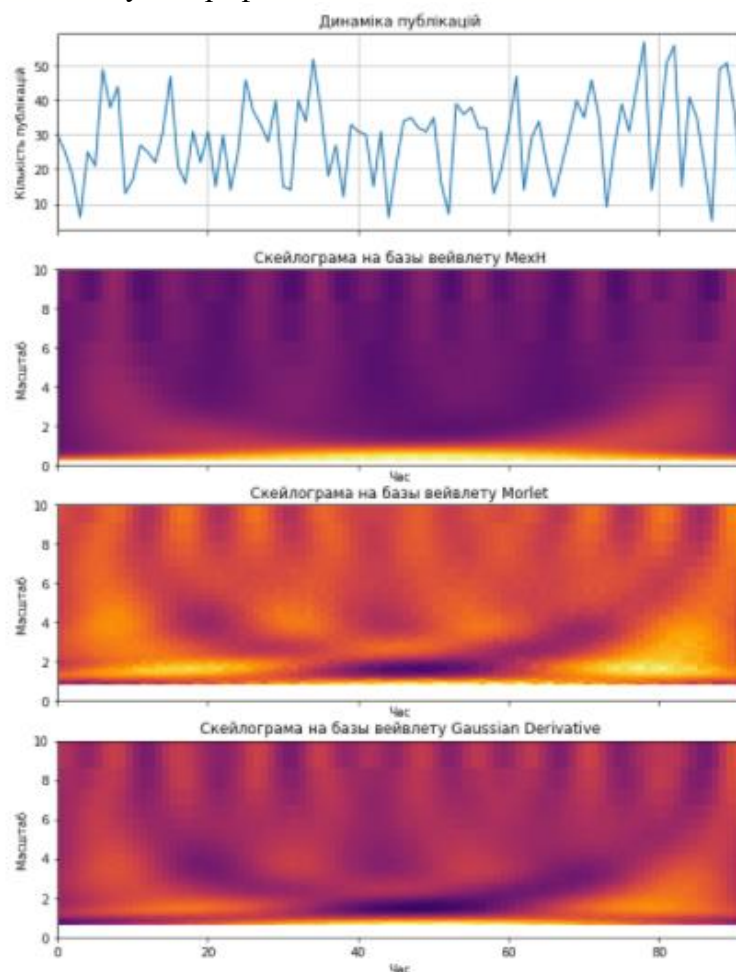


```

ax1.plot(t, x)
ax1.grid(True)
ax1.set_ylabel("Кількість публікацій")
ax1.set_title("Динаміка публікацій")
# Wavelet transform, i.e. scaleogram
cwtmatr, freqs = pywt.cwt(x, range(1, N), "mexh", sampling_period = 1 / f_s)
cwtmatr2, freqs2 = pywt.cwt(x, range(1, N), "morl", sampling_period = 1 / f_s)
cwtmatr3, freqs3 = pywt.cwt(x, range(1, N), "gaus8", sampling_period = 1 / f_s)
ax2.pcolormesh(t, freqs, cwtmatr, vmin=-100, cmap = "inferno" )
ax2.set_ylim(0,10)
ax2.set_ylabel("Масштаб")
ax2.set_xlabel("Час")
ax2.set_title("Скейлограма на бази вейвлету MexH")
#
ax3.pcolormesh(t, freqs2, cwtmatr2, vmin=-100, cmap = "inferno" )
ax3.set_ylim(0,10)
ax3.set_ylabel("Масштаб")
ax3.set_xlabel("Час")
ax3.set_title("Скейлограма на бази вейвлету Morlet")
#
ax4.pcolormesh(t, freqs3, cwtmatr3, vmin=-100, cmap = "inferno" )
ax4.set_ylim(0,10)
ax4.set_ylabel("Масштаб")
ax4.set_xlabel("Час")
ax4.set_title("Скейлограма на бази вейвлету Gaussian Derivative")
plt.show()

```

У результаті було отримано наступні графіки:



## ЛАБОРАТОРНА РОБОТА №8

## Завдання до лабораторної роботи №8:

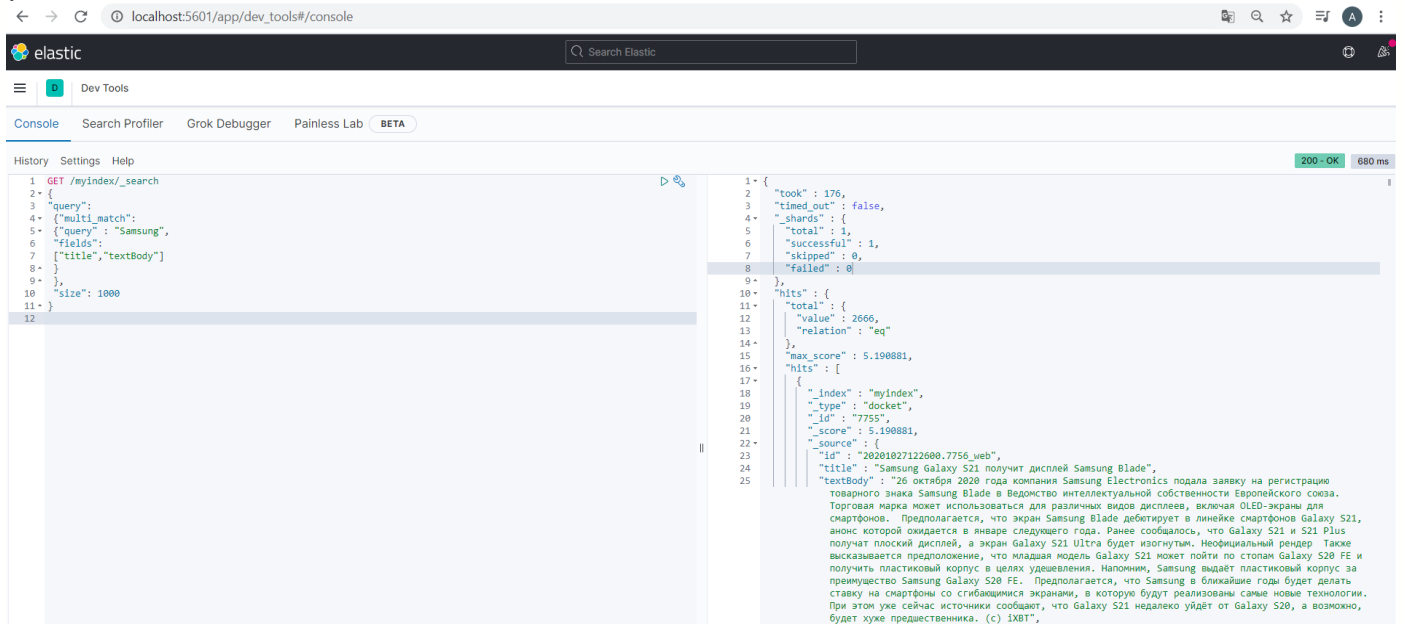
- Детально ознайомитись із засобами обробки тестових рядків в мові Python.
- Мовою Python самостійно розробити програмний модуль підрахунку ваги слів.
- Ознайомитись з іншими методами розрахунку ваги слів, зокрема методом TF-IDF, дисперсійним методом і методом горизонтальної видимості (HVG).

### Виконання роботи:

Спочатку треба було отримати дані, з якими працювати. Для цього я зробила запит до системи Elasticsearch у Dev Tools Kibana, результат роботи якого можна бачити на рисунку нижче у правій частині інтерфейсу.

## GET /myindex/\_search

```
{
  "query": {
    "multi_match": {
      "query": "Samsung",
      "fields": ["title", "textBody"]
    }
  },
  "size": 1000
}
```



Далі я скопіювала результат запити у файл, який назвала `samsung.txt`. Щоб зробити словник слів, які зустрічаються від `title` до `textBody`, підрахувати кількість зустрічань кожного слова, зробити висновок про топ 20 найвідоміших слів за запитом «Samsung», що не є стоп-словами, я дослідила програму, яка була запропонована у методичці. Для цього я запустила її на своїх даних. Як виявилось, у них були ще символи '%', '@' та '!', тому у регулярному виразі я їх дописала. Також варто зазначити, що `stop_words.txt`, у якому зберігались українські, російські та англійські стоп-слова, потрібно було зберігати у кодуванні ANSI.

```
import re
import string
f = open("./samsung.txt", "r", encoding="utf-8")
t = f.read()
f.close()
#print(t)
xml = t.split('\n')
t = ""
for i in range(len(xml)):
```

```
t=t+" "+xml[i]
#print(t)
title = re.findall("title" : "(.+)"source", t)
t=""
for i in range(len(title)):
    t=t+" "+title[i]
t=re.sub("textBody" :',',t)
t=re.sub('[—|\\|\\/?0-9",.()%!@$>«—:;_...]',',',t)
t=t.upper()
```

```

t=re.sub("\s|w\s|' ','t)
t=re.sub("\s|w\s|w\s|' ','t)
t=re.sub("\s|s+',' ','t)
word =t.split(' ')
word.sort()
# Dictionary building
d={}
old=""
n=0;
for i in range(len(word)):
    if (word[i] == old):
        n=n+1;
    else:
        #print(old,n)
        d[old]=n
        old=word[i]
        n=1
#print(old,n)
d[old]=n
#print(d)
sorted_dict = { }
sorted_keys = sorted(d, key=d.get, reverse=True) # [1, 3, 2]
for w in sorted_keys:

```

```

sorted_dict[w] = d[w]
print(sorted_dict)

f = open("./stop_words.txt","r")
t = f.read()
f.close()
t=t.upper()
stop =t.split("\n")
M=50
j=1
sorted_dict = { }
sorted_keys = sorted(d, key=d.get, reverse=True) # [1, 3, 2]
for w in sorted_keys:
    sorted_dict[w] = d[w]
    pr=0
    for i in range(len(stop)):
        if (stop[i] == w):
            pr=1
    if (pr == 0):
        print(j,w,sorted_dict[w])
        j=j+1
    if (j > M):
        break

```

Результат роботи програми можна бачити на рисунку нижче. Зрозуміло, що даних багато, тому наведена лише частина виводу на екран.

Найбільш ваговими словами за запитом «Samsung» виявились (число справа є кількістю зустрічань цього слова):

- 1 SAMSUNG 3691
- 2 GALAXY 2701
- 3 КОМПАНИЯ 689
- 4 СМАРТФОНОВ 633
- 5 СМАРТФОН 475
- 6 APPLE 445
- 7 IPHONE 427
- 8 NOTE 402

- 9 ULTRA 401
- 10 IXBT 399
- 11 КОМПАНИИ 379
- 12 HUAWEI 364
- 13 СМАРТФОНА 357
- 14 PRO 347
- 15 XIAOMI 320
- 16 СМАРТФОНЫ 282
- 17 ANDROID 261
- 18 SNAPDRAGON 253
- 19 EXYNOS 251
- 20 ПАМЯТИ 216

Далі я ознайомилась з методом TF і змінила програмну частину, де відбувається формування словника та підрахунок ваги слів. На сайті [nlpx.net/archives/57](http://nlpx.net/archives/57) я дізналась про те, що це робиться дуже просто: викликом функції Counter з бібліотеки collections. Порівнюючи результати отриманої програми з минулими можна було побачити, що вони співпадають (тому я їх не наводжу). Так виходить через те, що функція Counter є TF без нормалізації, тобто ми не робимо ділення на загальну кількість слів у тексті, а виконуємо такі ж дії як у методичці.

```
from collections import Counter
# Dictionary building
d = Counter(word)
```



```
# Dictionary building
d={}
old=""
n=0;
for i in range(len(word)):
    if (word[i] == old):
        n=n+1;
    else:
        #print(old,n)
        d[old]=n
        old=word[i]
        n=1
    #print(old,n)
    d[old]=n
#print(d)
```

Виконавши циклом ділення на кількість слів у тексті, отримуємо наступне (зроблене округлення до 5 знаків за допомогою методу format):

```
d = Counter(word)
for i in d:
    d[i] = d[i]/float(len(word))
```

- 1 SAMSUNG 0.03064
- 2 GALAXY 0.02242
- 3 КОМПАНИЯ 0.00572
- 4 СМАРТФОНОВ 0.00525
- 5 СМАРТФОН 0.00394
- 6 APPLE 0.00370
- 7 IPHONE 0.00354
- 8 NOTE 0.00334
- 9 ULTRA 0.00333
- 10 IXBT 0.00331
- 11 КОМПАНИИ 0.00315
- 12 HUAWEI 0.00302
- 13 СМАРТФОНА 0.00296
- 14 PRO 0.00288
- 15 XIAOMI 0.00267
- 16 СМАРТФОНЫ 0.00234
- 17 ANDROID 0.00217
- 18 SNAPDRAGON 0.00210
- 19 EXYNOS 0.00208
- 20 ПАМЯТИ 0.00179

## ЛАБОРАТОРНА РОБОТА №9

### Завдання до лабораторної роботи №9:

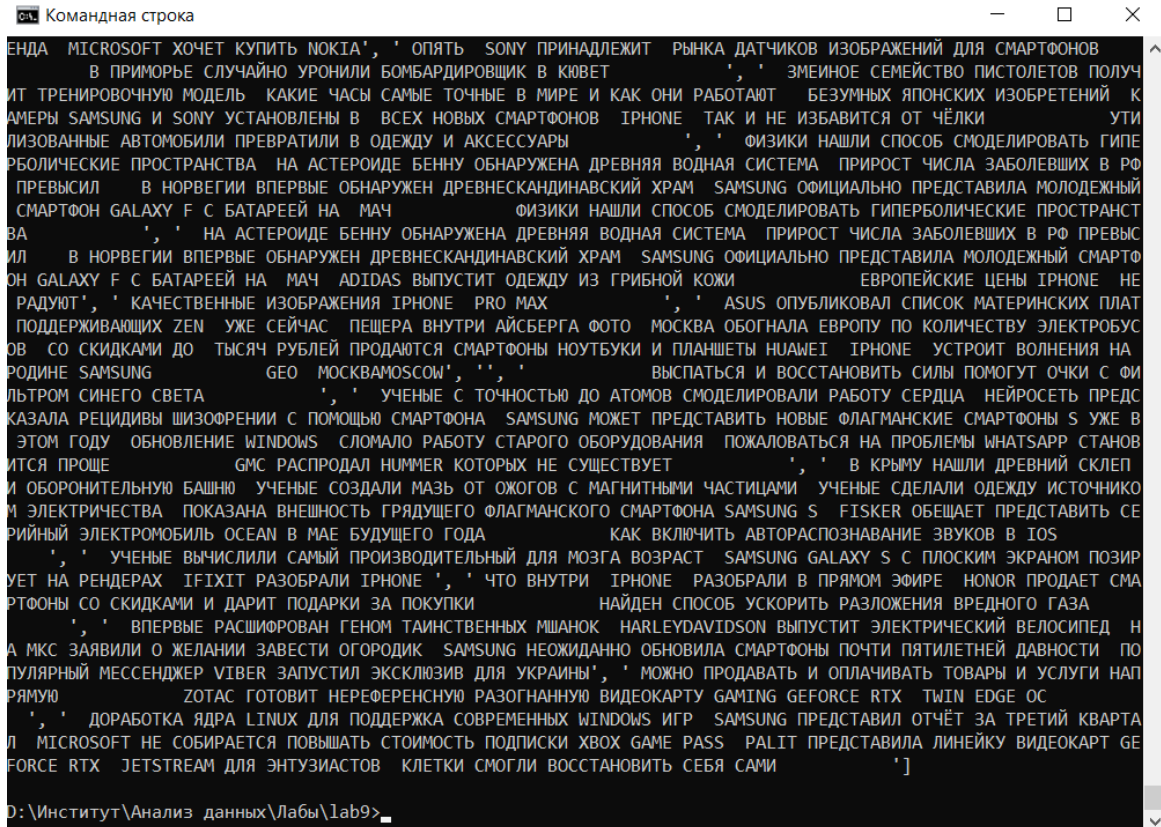
- Детально ознайомитись із засобами перетворення типів даних в мові Python.
- Самостійно встановити на комп'ютері систему Gephi.
- Ознайомитись з основними режимами і можливостями системи Gephi

### Виконання роботи:

Для виконання лабораторної роботи достатньо було проробити кроки, що були запропоновані у методичці, на своїх даних.

```
import re
import string
import numpy as np
f = open("./samsung.txt", "r", encoding="utf-8")
t = f.read()
f.close()
json = t.split('\n')
t=""
for i in range(len(json)):
    t=t+" "+json[i]
title = re.findall('title' : "(.+)"source"', t)
t=""
for i in range(len(title)):
    t=t+" "+title[i]
t=re.sub("textBody" :','.',t)
t=re.sub('[—%—||[\]@&?!\\0-9",()$+»«—:;_...]', "", t)
t=t.upper()
sent = t.split('.')
print(sent)
```

Результат роботи цієї частинки програми:



```
Командная строка
ЕНДА MICROSOFT ХОЧЕТ КУПИТЬ NOKIA', ' ОПЯТЬ SONY ПРИНАДЛЕЖИТ РЫНКА ДАТЧИКОВ ИЗОБРАЖЕНИЙ ДЛЯ СМАРТФОНОВ
В ПРИМОРЬЕ СЛУЧАЙНО УРОНИЛИ БОМБАРДИРОВЩИК В КЮВЕТ ', ' ЗМЕИНОЕ СЕМЕЙСТВО ПИСТОЛЕТОВ ПОЛУЧ
ИТ ТРЕНИРОВОЧНУЮ МОДЕЛЬ КАКИЕ ЧАСЫ САМЫЕ ТОЧНЫЕ В МИРЕ И КАК ОНИ РАБОТАЮТ БЕЗУМНЫХ ЯПОНСКИХ ИЗОБРЕТЕНИЙ К
АМЕРЫ SAMSUNG И SONY УСТАНОВЛЕНЫ В ВСЕХ НОВЫХ СМАРТФОНОВ IPHONE ТАК И НЕ ИЗБАВИТСЯ ОТ ЧЁЛКИ УТИ
ЛИЗОВАННЫЕ АВТОМОБИЛИ ПРЕВРАТИЛИ В ОДЕЖДУ И АКСЕССУАРЫ ', ' ФИЗИКИ НАШЛИ СПОСОБ СМОДЕЛИРОВАТЬ ГИПЕ
РБОЛИЧЕСКИЕ ПРОСТРАНСТВА НА АСТЕРОИДЕ БЕННУ ОБНАРУЖЕНА ДРЕВНЯЯ ВОДНАЯ СИСТЕМА ПРИРОСТ ЧИСЛА ЗАБОЛЕВШИХ В РФ
ПРЕВЫСИЛ В НОРВЕГИИ ВПЕРВЫЕ ОБНАРУЖЕН ДРЕВНЕСКАНДИНАВСКИЙ ХРАМ SAMSUNG ОФИЦИАЛЬНО ПРЕДСТАВИЛА МОЛОДЕЖНЫЙ
СМАРТФОН GALAXY F С БАТАРЕЕЙ НА МАЧ ADIDAS ВЫПУСТИТ ОДЕЖДУ ИЗ ГРИБНОЙ КОЖИ ФИЗИКИ НАШЛИ СПОСОБ СМОДЕЛИРОВАТЬ ГИПЕРБОЛИЧЕСКИЕ ПРОСТРАНСТ
ВА ', ' НА АСТЕРОИДЕ БЕННУ ОБНАРУЖЕНА ДРЕВНЯЯ ВОДНАЯ СИСТЕМА ПРИРОСТ ЧИСЛА ЗАБОЛЕВШИХ В РФ ПРЕВЫС
ИЛ В НОРВЕГИИ ВПЕРВЫЕ ОБНАРУЖЕН ДРЕВНЕСКАНДИНАВСКИЙ ХРАМ SAMSUNG ОФИЦИАЛЬНО ПРЕДСТАВИЛА МОЛОДЕЖНЫЙ СМАРТФ
ОН GALAXY F С БАТАРЕЕЙ НА МАЧ ADIDAS ВЫПУСТИТ ОДЕЖДУ ИЗ ГРИБНОЙ КОЖИ ЕВРОПЕЙСКИЕ ЦЕНЫ IPHONE НЕ
РАДУЮТ', ' КАЧЕСТВЕННЫЕ ИЗОБРАЖЕНИЯ IPHONE PRO MAX ', ' ASUS ОПУБЛИКОВАЛ СПИСОК МАТЕРИНСКИХ ПЛАТ
ПОДДЕРЖИВАЮЩИХ ZEN УЖЕ СЕЙЧАС ПЕЩЕРА ВНУТРИ АЙСБЕРГА ФОТО МОСКВА ОБОГНАЛА ЕВРОПУ ПО КОЛИЧЕСТВУ ЭЛЕКТРОБУС
ОВ СО СКИДКАМИ ДО ТЫСЯЧ РУБЛЕЙ ПРОДАЮТСЯ СМАРТФОНЫ НОУТБУКИ И ПЛАНШЕТЫ HUAWEI IPHONE УСТРОИТ ВОЛНЕНИЯ НА
РОДИНЕ SAMSUNG GEO МОСКВА MOSCOW', '', ' ВЫСПАТЬСЯ И ВОССТАНОВИТЬ СИЛЫ ПОМОГУТ ОЧКИ С ФИ
ЛЬТРОМ СИНЕГО СВЕТА ', ' УЧЕНЫЕ С ТОЧНОСТЬЮ ДО АТОМОВ СМОДЕЛИРОВАЛИ РАБОТУ СЕРДЦА НЕЙРОСЕТЬ ПРЕДС
КАЗАЛА РЕЦИДИВЫ ШИЗОФРЕНИИ С ПОМОЩЬЮ СМАРТФОНА SAMSUNG МОЖЕТ ПРЕДСТАВИТЬ НОВЫЕ ФЛАГМАНСКИЕ СМАРТФОНЫ S УЖЕ В
ЭТОМ ГОДУ ОБНОВЛЕНИЕ WINDOWS СЛОМАЛО РАБОТУ СТАРОГО ОБОРУДОВАНИЯ ПОЖАЛОВАТЬСЯ НА ПРОБЛЕМЫ WHATSAPP СТАНОВ
ИТСЯ ПРОЩЕ ГМС РАСПРОДАЛ HUMMER КОТОРЫХ НЕ СУЩЕСТВУЕТ ', ' В КРЫМУ НАШЛИ ДРЕВНИЙ СКЛЕП
И ОБОРОНИТЕЛЬНУЮ БАШНЮ УЧЕНЫЕ СОЗДАЛИ МАЗЬ ОТ ОЖОГОВ С МАГНИТНЫМИ ЧАСТИЦАМИ УЧЕНЫЕ СДЕЛАЛИ ОДЕЖДУ ИСТОЧНИКО
М ЭЛЕКТРИЧЕСТВА ПОКАЗАНА ВНЕШНОСТЬ ГРЯДУЩЕГО ФЛАГМАНСКОГО СМАРТФОНА SAMSUNG S FISKER ОБЕЩАЕТ ПРЕДСТАВИТЬ СЕ
РИЙНЫЙ ЭЛЕКТРОМОБИЛЬ OCEAN В МАЕ БУДУЩЕГО ГОДА КАК ВКЛЮЧИТЬ АВТОРАСПОЗНАВАНИЕ ЗВУКОВ В IOS
', ' УЧЕНЫЕ ВЫЧИСЛИЛИ САМЫЙ ПРОИЗВОДИТЕЛЬНЫЙ ДЛЯ МОЗГА ВОЗРАСТ SAMSUNG GALAXY S С ПЛОСКИМ ЭКРАНОМ ПОЗИР
УЕТ НА РЕНДЕРАХ IFIXIT РАЗОБРАЛИ IPHONE ', ' ЧТО ВНУТРИ IPHONE РАЗОБРАЛИ В ПРЯМОМ ЭФИРЕ HONOR ПРОДАЕТ СМА
РТФОНЫ СО СКИДКАМИ И ДАРИТ ПОДАРКИ ЗА ПОКУПКИ НАЙДЕН СПОСОБ УСКОРИТЬ РАЗЛОЖЕНИЯ ВРЕДНОГО ГАЗА
', ' ВПЕРВЫЕ РАШИФРОВАН ГЕНОМ ТАИНСТВЕННЫХ МШАНОК HARLEYDAVIDSON ВЫПУСТИТ ЭЛЕКТРИЧЕСКИЙ ВЕЛОСИПЕД Н
А МКС ЗАЯВИЛИ О ЖЕЛАНИИ ЗАВЕСТИ ОГОРОДИК SAMSUNG НЕОЖИДАННО ОБНОВИЛА СМАРТФОНЫ ПОЧТИ ПЯТИЛЕТНЕЙ ДАВНОСТИ ПО
ПУЛЯРНЫЙ МЕССЕНДЖЕР VIBER ЗАПУСТИЛ ЭКСКЛЮЗИВ ДЛЯ УКРАИНЫ', ' МОЖНО ПРОДАВАТЬ И ОПЛАЧИВАТЬ ТОВАРЫ И УСЛУГИ НАП
РЯМУЮ ZOTAC ГОТОВИТ НЕРЕФЕРЕНСНУЮ РАЗОГНАННУЮ ВИДЕОКАРТУ GAMING GEFORCE RTX TWIN EDGE OC
', ' ДОРАБОТКА ЯДРА LINUX ДЛЯ ПОДДЕРЖКИ СОВРЕМЕННЫХ WINDOWS ИГР SAMSUNG ПРЕДСТАВИЛ ОТЧЁТ ЗА ТРЕТИЙ КВАРТА
Л MICROSOFT НЕ СОБИРАЕТСЯ ПОВЫШАТЬ СТОИМОСТЬ ПОДПИСКИ XBOX GAME PASS PALIT ПРЕДСТАВИЛА ЛИНЕЙКУ ВИДЕОКАРТ GE
FORCE RTX JETSTREAM ДЛЯ ЭНТУЗИАСТОВ КЛЕТКИ СМОГЛИ ВОССТАНОВИТЬ СЕБЯ САМИ ']
```

```
f = open("./exit8.txt", "r", encoding="utf-8")
t = f.read()
f.close()
t=t.upper()
```



```
w=t.split('\n')
```

```
for i in range(len(w)):
    s=w[i].split(' ')
    w[i]=s[1]
mtr = np.eye(len(w))
for i in range(len(w)):
    mtr[i][i]=0
stroka=""
for i in range(len(w)):
    stroka=stroka+";"+w[i]
print(stroka)
```

Результат роботи цієї частинки програми:

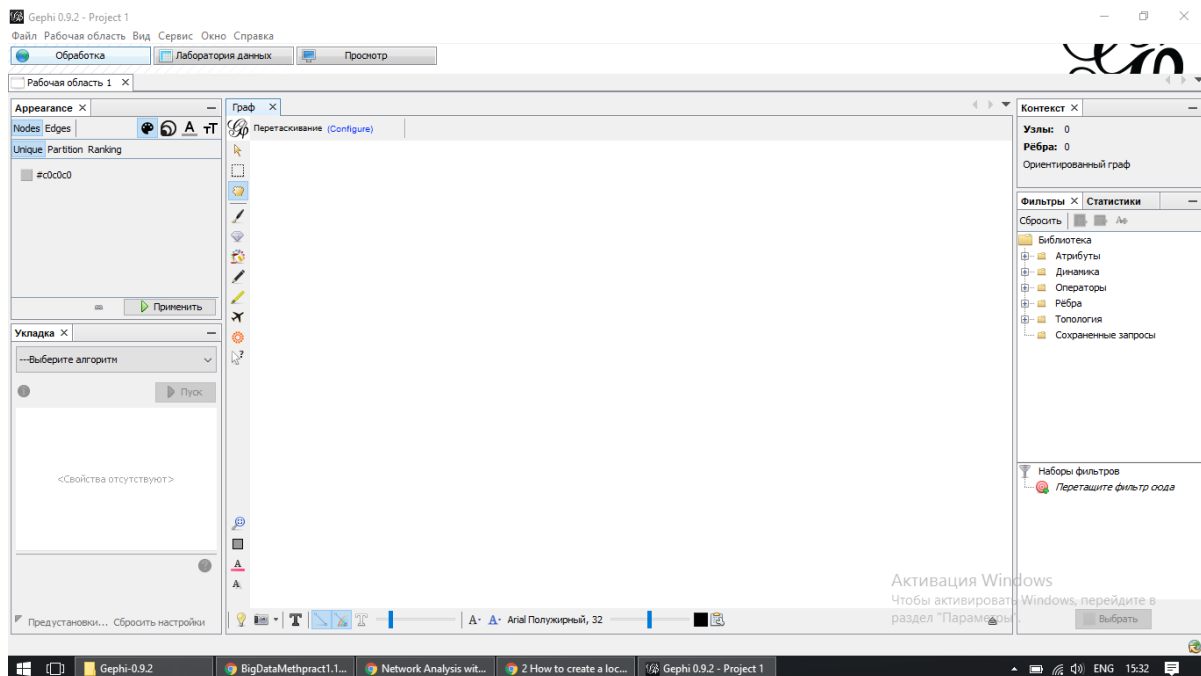
```
D:\Институт\Анализ данных\лабы\lab9>python lab9.py
;SAMSUNG;GALAXY;КОМПАНИЯ;СМАРТФОНОВ;СМАРТФОН;APPLE;IPHONE;NOTE;ULTRA;IXBT;КОМПАНИИ;HUAWEI;СМАРТФОНА;PRO;XIAOMI;СМАРТФОНЫ;ANDROID;SNAPDRAGON;EXYNOS;ПАМЯТИ
```

```
for i in range(len(sent)):
    for j in range(len(w)):
        for k in range(len(w)):
            if(j!=k):
                if (re.search(w[j],sent[i])):
                    if (re.search(w[k],sent[i])):
                        mtr[k][j]=mtr[k][j]+1
for i in range(len(w)):
    stroka=w[i]+";"
    for j in range(len(w)):
        a=int(mtr[i][j])
        b=a.__str__()
        if (j<len(w)-1):
            stroka=stroka+b+";"
        else:
            stroka=stroka+b
    print(stroka)
```

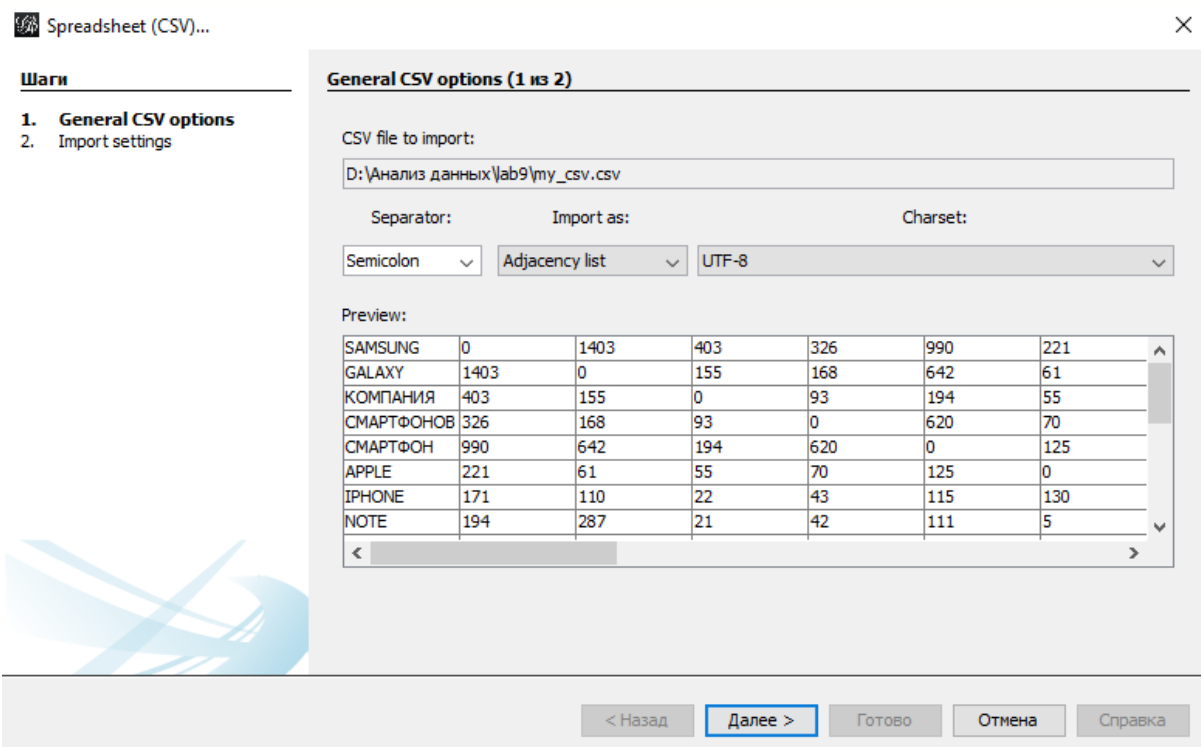
Результат роботи цієї частинки програми (матриця суміжності, матриця зв'язків):

```
SAMSUNG;0;1403;403;326;990;221;171;194;227;274;167;168;242;162;120;169;97;102;130;48
GALAXY;1403;0;155;168;642;61;110;287;336;145;56;60;182;137;45;104;82;66;64;41
КОМПАНИЯ;403;155;0;93;194;55;22;21;7;3;26;47;41;12;24;26;14;9;18;10
СМАРТФОНОВ;326;168;93;0;620;70;43;42;20;32;51;36;13;17;55;13;28;11;6;13
СМАРТФОН;990;642;194;620;0;125;115;111;82;107;90;87;481;78;123;278;94;68;59;43
APPLE;221;61;55;70;125;0;130;5;6;22;23;36;22;25;44;25;11;3;5;4
IPHONE;171;110;22;43;115;130;0;19;27;24;10;20;26;93;25;23;10;4;4;5
NOTE;194;287;21;42;111;5;19;0;94;22;5;15;21;52;19;14;17;17;8;3
ULTRA;227;336;7;20;82;6;27;94;0;29;9;22;22;37;15;9;12;18;18;8
IXBT;274;145;3;32;107;22;24;22;29;0;1;15;19;19;20;21;10;12;16;0
КОМПАНИИ;167;56;26;51;90;23;10;5;9;1;0;29;17;7;12;11;5;4;9;9
HUAWEI;168;60;47;36;87;36;20;15;22;15;29;0;19;54;45;18;6;5;2;9
СМАРТФОНА;242;182;41;13;481;22;26;21;22;19;17;19;0;25;20;4;12;13;12;6
PRO;162;137;12;17;78;25;93;52;37;19;7;54;25;0;32;15;17;12;5;3
XIAOMI;120;45;24;55;123;44;25;19;15;20;12;45;20;32;0;22;11;22;7;2
СМАРТФОНЫ;169;104;26;13;278;25;23;14;9;21;11;18;4;15;22;0;19;12;14;0
ANDROID;97;82;14;28;94;11;10;17;12;10;5;6;12;17;11;19;0;6;6;13
SNAPDRAGON;102;66;9;11;68;3;4;17;18;12;4;5;13;12;22;12;6;0;71;15
EXYNOS;130;64;18;6;59;5;4;8;18;16;9;2;12;5;7;14;6;71;0;12
ПАМЯТИ;48;41;10;13;43;4;5;3;8;0;9;9;6;3;2;0;13;15;12;0
```

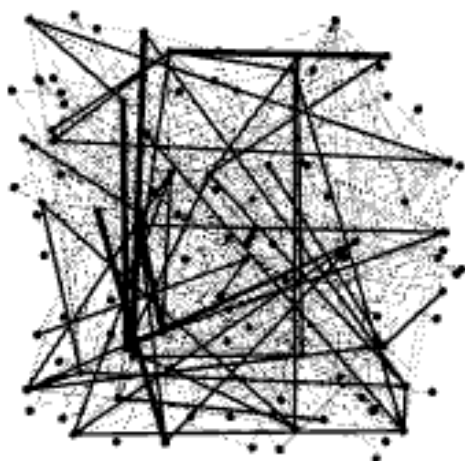
Далі я встановила Gerpi на свій локальний комп'ютер. Для роботи з даною системою треба було у конфігураційному файлі додатка змінити шлях до release файлу Java.



Потім я завантажила дані до системи.



Отримала наступний граф:





Потім я отримала наступне (тобто отримала 9 кластерів):

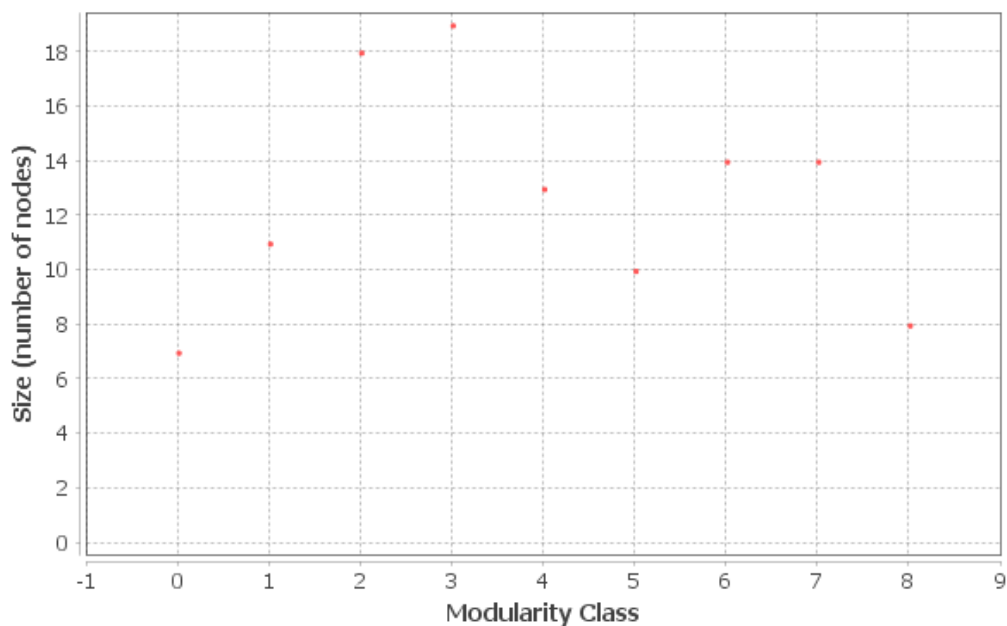
## Results:

Modularity: 0,285

Modularity with resolution: 0,285

Number of Communities: 9

### Size Distribution



Далі обробила мережу шляхом запуску алгоритму OpenOrd і отримала:

