https://developers.google.com/custom-search/ (дата звернення 01.05.2020).

ДОДАТОК А

Приклади вмісту використовуваних наборів даних

```
artist,song,link,text
ABBA,Ahe's My Kind Of Girl,/a/abba/ahes+my+kind+of+girl_20598417.html,"Look at her face, it's a wonderful face
And it means something special to me
Look at the way that she smiles when she sees me
How lucky can one fellow be?

She's just my kind of girl, she makes me feel fine
Who could ever believe that she could be mine?
She's just my kind of girl, without her I'm blue
And if she ever leaves me what could I do, what could I do?

And when we go for a walk in the park
And she holds me and squeezes my hand
We'll go on walking for hours and talking
About all the things that we plan

She's just my kind of girl, she makes me feel fine
Who could ever believe that she could be mine?
She's just my kind of girl, without her I'm blue
And if she ever leaves me what could I do, what could I do?

"
ABBA,"Andante, Andante",/a/abba/andante+andante_20002708.html,"Take it easy with me, please
Touch me gently like a summer evening breeze
Take your time, make it slow
Andante, Andante
Just let the feeling grow

Make your fingers soft and light
Let your body be the velvet of the night
Touch my soul, you know how
Andante, Andante
```

Рисунок А.1 - Приклад вміста набору даних «Songs»

0	1
0	2
0	3
0	4
0	5
0	6
0	7
0	8
0	9
0	10
0	11
0	12
0	13
0	14

Рисунок А.2 - Приклад основного вміста набору даних «Frienship»

node_id	label
0	male_noah
1	male_liam
2	male_william
3	male_benjamin
4	male_jacob
5	male_elijah
6	male_ethan
7	female_emma
8	female_olivia
9	female_ava
10	female_isabella
11	female_sophia
12	female_mia
13	female_amelia

Рисунок А.3 - Приклад довідкової таблиці набору даних «Friendship»

0	1
0	10
0	100
0	102
0	103
0	104
0	105
0	106
0	107
0	108
0	109
0	11
0	110
0	111
0	112
0	113
0	116

Рисунок А.4 - Приклад основного вміста набору даних «Enron»

node_id	address
0	phillip.allen@enron.com
1	tim.belden@enron.com
2	john.lavorato@enron.com
3	leah.arsdall@enron.com
4	randall.gay@enron.com
5	greg.piper@enron.com
6	david.l.johnson@enron.com
7	john.shafer@enron.com
8	joyce.teixeira@enron.com
9	mark.scott@enron.com
10	david.delainey@enron.com
11	paula.harris@enron.com
12	tim.heizenrader@enron.com
13	pallen@enron.com
14	jeffrey.hodge@enron.com

Рисунок А.5 - Прилад довідкової таблиці набору даних «Enron»

Рисунки демонструють записи, що знаходяться в використаних у імітаційному моделюванні методів ранжування наборів даних.

ДОДАТОК Б

Лістинг програмного коду практичних реалізацій методів ранжування

```
df = pd.read_csv('songdata.csv')
data = df["text"][:100]
stopWords = stopwords.words('english')
cv=CountVectorizer(stop_words = stopWords)
from time import time
start_time = time()
word_count_vector=cv.fit_transform(data)
print(word_count_vector.shape)
print(cv.transform(data))
tfidf_transformer=TfidfTransformer(smooth_idf=True,use_idf=True)
tfidf_transformer.fit(word_count_vector)
df_idf = pd.DataFrame(tfidf_transformer.idf_, index=cv.get_feature_names(),columns=["idf_weights"])
print(df_idf.sort_values(by=['idf_weights']))
count_vector=cv.transform(data)
print(count_vector)
tf_idf_vector=tfidf_transformer.transform(count_vector)
print(tf_idf_vector)
feature_names = cv.get_feature_names()
first_document_vector=tf_idf_vector[0]
df = pd.DataFrame(first_document_vector.T.todense(), index=feature_names, columns=["tfidf"])
print(df.sort_values(by=["tfidf"],ascending=False))
query = ["every dream is like"]
print(query)
query_vec = cv.transform(query).toarray()
print(query_vec)
results = cosine_similarity(tf_idf_vector,query_vec).reshape((-1,))
spr_run_time = time() - start_time
print("Results")
d1 = pd.read_csv('songdata.csv')
for i in results.argsort()[-10:][::-1]:
     print(d1.iloc[i,0], "--", d1.iloc[i,1])
print("Running time: {:.4f} seconds".format(spr_run_time))
```

Рисунок Б.1 - Програмний код реалізації текстового ранжування набору даних «Songs»

```
class SparsePageRank:
    def load_graph_dataset(self, data_home, is_undirected=False):
        edge_path = "{}/edges.tsv".format(data_home)
        edges = np.loadtxt(edge_path, dtype=int)
        n = int(np.amax(edges)) + 1
        rows = edges[:, 0]
        cols = edges[:, 1]
        weights = edges[:, 2]
        self.A = csr_matrix((weights, (rows, cols)), shape=(n, n))
        if is_undirected == True:
            self.A = self.A + self.A.T
        self.n = self.A.shape[0]  # number of nodes
        self.m = self.A.nnz  # number of edges
```

Рисунок Б.2 - Вміст функції load_graph_dataset() реалізації метода PageRank для перетворення набору даних на посилальний граф

```python
class SparsePageRank(SparsePageRank):
    def normalize(self):
        '''
        Perform the row-normalization of the given adjacency matrix
        '''
        d = self.A.sum(axis=1)  # since A is csr_matrix, the result of sum() is not a normal vector
        d = np.asarray(d).flatten()  # to make it vector

        d = np.maximum(d, np.ones(self.n))
        invd = 1.0 / d
        invD = spdiags(invd, 0, self.n, self.n)

        self.nA = invD.dot(self.A)
        self.nAT = self.nA.T

        self.out_degrees = d
```

Рисунок Б.3 - Вміст функції normalize() реалізації метода Pagerank для нормалізації матриці інцидентності графа

```python
class SparsePageRank(SparsePageRank):
    def iterate_PageRank(self, b=0.15, epsilon=1e-9, maxIters=100, handles_deadend=True):
        q = np.ones(self.n) / self.n
        old_p = q
        residuals = []

        for t in range(maxIters):
            if handles_deadend:
                p = (1 - b) * (self.nAT.dot(old_p))
                S = np.sum(p)
                p = p + (1 - S) * q
            else:
                p = (1 - b) * (self.nAT.dot(old_p)) + (b * q)

            residual = np.linalg.norm(p - old_p, 1)
            residuals.append(residual)
            old_p = p

            if residual < epsilon:
                break

        return p, residuals
```

Рисунок Б.4 - Вміст функції iterate_PageRank() реалізації метода PageRank для обчислення показників авторитетності

```python
class SparsePageRank(SparsePageRank):
    def rank_nodes(self, ranking_scores, topk=-1):
        sorted_nodes = np.flipud(np.argsort(ranking_scores))  # argsort in the descending order
        sorted_scores = ranking_scores[sorted_nodes]  # sort the ranking scores
        ranks = range(1, self.n + 1)  # 0~n-1

        result_labels = self.node_labels.iloc[sorted_nodes][0:topk]
        result_labels.insert(0, "rank", ranks[0:topk])
        result_labels["score"] = sorted_scores[0:topk]
        result_labels.reset_index(drop=True, inplace=True)
        return result_labels
```

Рисунок Б.5 - Вміст функції rank_nodes() реалізації метода PageRank для сортування вершин графа

```
data_home = './data/enron'
spr = SparsePageRank()
spr.load_graph_dataset(data_home, is_undirected=False)
spr.load_node_labels(data_home)
spr.normalize()
from time import time
start_time = time()
p, p_residuals = spr.iterate_PageRank(b=0.15, epsilon=1e-9, maxIters=100, handles_deadend=True)
plot_residuals(p_residuals, 'Change of PageRank Residuals from PageRank')
spr_run_time = time() - start_time
print("\nEnron dataset")
print("Running time: {:.4f} seconds".format(spr_run_time))
p_exact = dpr.compute_exact_PageRank(b=0.15)
p_iter, _ = dpr.iterate_PageRank(b=0.15, epsilon=1e-9, maxIters=100)
error = np.linalg.norm(p_exact - p_iter, 1)
print("Error between exact and iterative PageRank scores: {:e}".format(error))#print("Adjacency matrix of Enron:")
print(spr.nA)
row_sums = np.asarray(spr.nA.sum(axis=1)).flatten()
print("sum of each row in the row-normalized matrix of Enron")
for (i, degree, row_sum) in zip(range(spr.n), spr.out_degrees, row_sums):
    print("node: {:2d}, out-degree: {:2d},  row_sum: {:.2f}".format(i, int(degree), row_sum))
num_deadends = np.count_nonzero(spr.out_degrees == True)
print("The number n of nodes matrix of Enron: {}".format(spr.n))
print("The number m of edges matrix of Enron: {}".format(spr.m))
print("The number of deadend nodes matrix of Enron: {}".format(num_deadends))
print("The sum of the PageRank score vector for Enron: {:.2f}".format(np.sum(p)))
print("PageRank score for Enron")
for (i, score) in zip(range(spr.n), p):
    print("node: {:2d}, PageRank score: {:.4f}".format(i, score))
print("Top-10 rankings based on the authority score vector:")
print(spr.rank_nodes(p, topk=10))
```

Рисунок Б.6 - Програмний код реалізації метода PageRank з використанням набору даних «Enron»

```
import networkx as nx
import matplotlib.pyplot as plt
print("Networkx - 10 elements")
G=nx.fast_gnp_random_graph(15,0.5,directed=True)
print("Graph: number of nodes  = {:2d}, number of edges - {:2d}".format(G.number_of_nodes(), G.number_of_edges()))
print("Nodes")
print(G.nodes)
print("Out edges")
print(G.out_edges)
print("Edges")
print(G.edges)
nx.draw(G,with_labels=True)
import operator
from time import time
start_time = time()
pr=nx.pagerank(G,alpha=0.85, )
spr_run_time = time() - start_time
rank_vector=np.array([[*pr.values()]])
print("PageRank score")
print(pr)
print("Rank vector")
print(rank_vector)
print("Sorted rank vector")
print(sorted(pr.items(), key=operator.itemgetter(1), reverse=True))
best_node=np.argmax(rank_vector)
print("The most popular website is {}".format(best_node))
print("Running time: {:.4f} seconds".format(spr_run_time))
plt.show()
```

Рисунок Б.7 - Програмний код реалізації метода PageRank, з використанням бібліотеки NetworkX

```
class SparseHITS:
    def load_graph_dataset(self, data_home, is_undirected=False):
        edge_path = "{}/edges.tsv".format(data_home)
        edges = np.loadtxt(edge_path, dtype=int)
        n = int(np.amax(edges[:, 0:2])) + 1
        rows = edges[:, 0]
        cols = edges[:, 1]
        weights = edges[:, 2]
        self.A = csr_matrix((weights, (rows, cols)), shape=(n, n), dtype=float)
        if is_undirected == True:
            self.A = self.A + self.A.T
        self.AT = self.A.T
        self.n = self.A.shape[0]  # number of nodes
        self.m = self.A.nnz  # number of edges
```

Рисунок Б.8 - Вміст функції load_graph_dataset() реалізації метода HITS для перетворення набору даних на посилальний граф

```
class SparseHITS(SparseHITS):
    def iterate_HITS(self, epsilon=1e-9, maxIters=100):
        old_h = np.ones(self.n) / self.n
        old_a = np.ones(self.n) / self.n
        h_residuals = []
        a_residuals = []
        for t in range(maxIters):
            h = self.A.dot(old_a)
            a = self.AT.dot(h)
            h = h / np.linalg.norm(h, 2)
            a = a / np.linalg.norm(a, 2)
            h_residual = np.linalg.norm(h - old_h, 1)
            a_residual = np.linalg.norm(a - old_a, 1)
            h_residuals.append(h_residual)
            a_residuals.append(a_residual)
            old_h = h
            old_a = a
            if h_residual < epsilon and a_residual < epsilon:
                break
        return h, a, h_residuals, a_residuals
```

Рисунок Б.9 - Вміст функції iterate_HITS() реалізації метода HITS для обчислення показників авторитетності та посередницької оцінки

```
class SparseHITS(SparseHITS):
    def rank_nodes(self, ranking_scores, topk=-1):
        sorted_nodes = np.flipud(np.argsort(ranking_scores)) # argsort in the descending order
        sorted_scores = ranking_scores[sorted_nodes]         # sort the ranking scores
        ranks = range(1, self.n+1) # 0~n-1

        result_labels = self.node_labels.iloc[sorted_nodes][0:topk]
        result_labels.insert(0, "rank", ranks[0:topk])
        result_labels["score"] = sorted_scores[0:topk]
        result_labels.reset_index(drop = True, inplace = True)
        return result_labels
```

Рисунок Б.10 - Вміст функції rank_nodes() реалізації метода HITS для сортування вершин графа

```
from time import time
data_home = './data/enron'
hits = SparseHITS()
start_time = time()
hits.load_graph_dataset(data_home, is_undirected=False)
hits.load_node_labels(data_home)
h, a, h_residuals, a_residuals = hits.iterate_HITS(epsilon=1e-9, maxIters=100)
hits_run_time = time() - start_time
h, a, h_residuals, a_residuals = hits.iterate_HITS(epsilon=1e-9, maxIters=100)
plot_residuals(h_residuals, 'Change of Hub Residuals from HITS')
plot_residuals(a_residuals, 'Change of Authority Residuals from HITS')
print("The number n of nodes matrix of Enron: {}".format(hits.n))
print("The number m of edges matrix of Enron: {}".format(hits.m))
print("Hub and Auth score for Enron")
for (i, h, a) in zip(range(hits.n), h, a):
    print("node: {:2d}, Hub: {:.4f}, Auth: {:.4f}".format(i, h, a))
print("Running time: {:.4f} seconds".format(hits_run_time))
print("Top-10 rankings based on the hub score vector:")
print(hits.rank_nodes(h, topk=10))
print("Top-10 rankings based on the authority score vector:")
print(hits.rank_nodes(a, topk=10))
```

Рисунок Б.11 - Програмний код реалізації метода HITS з використанням набору даних «Enron»

```
print("Graph: number of nodes  = {:2d}, number of edges - {:2d}".format(G.number_of_nodes(), G.number_of_edges()))
print("Nodes")
print(G.nodes)
print("Out edges")
print(G.out_edges)
print("Edges")
print(G.edges)
import operator
from time import time
start_time = time()
hubs, authorities =nx.hits(G, normalized=True)
spr_run_time = time() - start_time
rank_vector_hub=np.array([[*hubs.values()]])
rank_vector_auth=np.array([[*authorities.values()]])
print("Hub score")
print(hubs)
print("Authorities score")
print(authorities)
print("Hub vector")
print(rank_vector_hub)
print("Authorities vector")
print(rank_vector_auth)
print("Sorted hub vector")
print(sorted(hubs.items(), key=operator.itemgetter(1), reverse=True))
print("Sorted authorities vector")
print(sorted(authorities.items(), key=operator.itemgetter(1), reverse=True))
best_node_hub=np.argmax(rank_vector_hub)
best_node_auth=np.argmax(rank_vector_auth)
print("The most popular website by hub is {}".format(best_node_hub))
print("The most popular website by auths is {}".format(best_node_auth))
print("Running time: {:.4f} seconds".format(spr_run_time))
plt.show()
```

Рисунок Б.12 - Програмний код реалізації метода HITS, з використанням бібліотеки NetworkX

```
4 references
class Links
{
    5 references
    public Result Item { get; set; }
    6 references
    public int Counter { get; set; }
}
```

Рисунок Б.13 - Клас для списка повторних результатів реалізації гібридного методу ранжування

```
0 references
public static IList<Result> Search(string query)
{

    CseResource.ListRequest listRequest = Service.Cse.List(query);
    listRequest.Cx = cx;

    Search search = listRequest.Execute();
    return search.Items;
}
```

Рисунок Б.14 - Вміст функції Search() реалізації гібридного методу ранжування для безпосередньої відправки запиту до пошукової системи, а також отримання результатів пошуку

```
1 reference
private static void SearchQuery(string query)
{
    while (query == "")
    {
        Console.ReadLine();
    }
    const string apiKey = "AIzaSyDK_rWlN29otcRupBZBFE60145EZPbxWf4";
    const string searchEngineId = "005356074109675249183:zryoa17pnmy";
    var customSearchService = new CustomsearchService(new BaseClientService.Initializer { ApiKey = apiKey });
    var listRequest = customSearchService.Cse.List(query);
    listRequest.Cx = searchEngineId;
    IList<Result> paging = new List<Result>();
    var count = 0;
    while (paging != null && count != 2)
    {
        Console.WriteLine($"Сторінка {count}");
        listRequest.Start = count * 10 + 1;
        paging = listRequest.Execute().Items;
        if (paging != null)
            foreach (var item in paging)
            {
                var listItem = results.FirstOrDefault(x => x.Item.Link == item.Link);

                if (listItem != null)
                {
                    listItem.Counter += 1;
                }
                else
                {
                    results.Insert(index: 0, new Links() { Item = item, Counter = 0 });
                }
                Console.WriteLine(item.Title + Environment.NewLine +  item.Link +
                                  Environment.NewLine + Environment.NewLine);
            }

        count++;
    }

    Console.WriteLine("Виконано.");
}
```

Рисунок Б.15 - Вміст функції SearchQuery() реалізації гібридного методу ранжування для прийняття запиту, його відправки, обробки отриманих сторінок та списку повторних результатів

```
static void Main(string[] args)
{
    Console.OutputEncoding = Encoding.UTF8;
    results.Insert(index:0, new Links() { Item = new Result() { Link = "" }, Counter = 0 });
    string x1 = "";
    while (true)
    {
        Console.InputEncoding = Encoding.UTF8;
        x1 = ReadLineUTF();
        SearchQuery(x1);
        var ordered = results.OrderByDescending(x => x.Counter).ToList();
        Console.WriteLine("Попередній запит: " + x1 + Environment.NewLine);
        Console.WriteLine("Релевантні результати");
        foreach (var result in ordered)
        {
            if (result.Counter != 0)
            {
                Console.WriteLine(result.Item.Title + Environment.NewLine + result.Item.Link +
                                  Environment.NewLine + "[Знайдено разів : " + result.Counter + "]" + Environment.NewLine + Environment.NewLine);
            }
        }
        Console.WriteLine("Введіть новий запит");
    }
```

Рисунок Б.16 - Вміст функціїї Main() реалізації гібридного методу ранжування для запуску програмного кода та виведення результатів на екран

Рисунки демонструють вміст функцій, які були написані та використані для практичної реалізації методів ранжування.

ДОДАТОК В

Результати імітаційного моделювання методів ранжування

```
TF-IDF with feature names
            tfidf
little   0.589463
tiny     0.589463
house    0.475575
mouse    0.280882
ate      0.000000
away     0.000000
cat      0.000000
end      0.000000
finally  0.000000
ran      0.000000
saw      0.000000
story    0.000000
```

Рисунок В.1 - Вектор с показниками TF-IDF для документа з контентом «the house had a tiny little mouse»

```
TF_IDF scores
  (0, 1960) 0.11212748841469226
  (0, 1953) 0.17060970254869146
  (0, 1911) 0.05783283379507727
  (0, 1892) 0.09728925424100486
  (0, 1889) 0.09728925424100486
  (0, 1737) 0.07708108140310208
  (0, 1708) 0.10621082855603481
  (0, 1611) 0.130506312607491
  (0, 1597) 0.09374866422189355
  (0, 1578) 0.07536984002909483
  (0, 1562) 0.11975538964735696
  (0, 1470) 0.11975538964735696
  (0, 1268) 0.11975538964735696
  (0, 1224) 0.10621082855603481
  (0, 1194) 0.053141509252811664
  (0, 1109) 0.21242165711206962
  (0, 1080) 0.11212748841469226
  (0, 1060) 0.16599548252351906
  (0, 1050) 0.11975538964735696
```

Рисунок В.2 - Частина TF-IDF показників для набору даних «Songs»

TF-IDF with feature names

	tfidf
could	0.476809
kind	0.323502
ever	0.295050
girl	0.267704
mine	0.212422
...	...
follows	0.000000
following	0.000000
follow	0.000000
folks	0.000000
zoo	0.000000

Рисунок В.3 - Вектор с показниками TF-IDF для першого документа з набору даних «Songs»

Def Leppard,Have You Ever Needed Someone So Bad,/d/def+leppard/have+you+ever+needed+someone+so+bad_20038876.html,"Here I am, I'm in the wrong bed again
It's a game I just can't win
There you are breathin' soft on my skin, yeah
Still you won't let me in

Why save your kisses for a rainy day
Baby let the moment take your heart away

[Chorus]
Have you ever needed someone so bad, yeah
Have you ever wanted someone
You just couldn't have
Did you ever try so hard
That your world just fell apart
Have you ever needed someone so bad

And to the girl I gotta have
I gotta have you baby

There you go, midnight promises again, yeah
But they're broken by the dawn
You want to go further, faster every day, baby
But in the morning you'll be gone
And I'm alone

Why save your kisses for a rainy day
Baby let the moment take your heart away

Рисунок В.4 - Частина тексту найрелевантнішого до запиту «every dream is like» документа набору даних «Songs»

```
Sum of each row in the row-normalized matrix of small dataset
node:  0, out-degree: 14,  row_sum: 1.00
node:  1, out-degree: 15,  row_sum: 1.00
node:  2, out-degree: 17,  row_sum: 1.00
node:  3, out-degree: 23,  row_sum: 1.00
node:  4, out-degree: 27,  row_sum: 1.00
node:  5, out-degree: 27,  row_sum: 1.00
node:  6, out-degree: 12,  row_sum: 1.00
node:  7, out-degree: 14,  row_sum: 1.00
node:  8, out-degree: 16,  row_sum: 1.00
node:  9, out-degree: 12,  row_sum: 1.00
node: 10, out-degree: 14,  row_sum: 1.00
node: 11, out-degree: 10,  row_sum: 1.00
node: 12, out-degree: 13,  row_sum: 1.00
node: 13, out-degree:  8,  row_sum: 1.00
node: 14, out-degree:  9,  row_sum: 1.00
node: 15, out-degree:  5,  row_sum: 1.00
node: 16, out-degree:  6,  row_sum: 1.00
node: 17, out-degree:  1,  row_sum: 1.00
node: 18, out-degree:  6,  row_sum: 1.00
```

Рисунок В.5 - Перевірка нормалізації матриці за рядком у методі PageRank, використовучи набір даних «Friendship»

```
Small dataset - Hub and Auth scores
node:  0, Hub: 0.1897, Auth: 0.1463
node:  1, Hub: 0.1758, Auth: 0.2055
node:  2, Hub: 0.2249, Auth: 0.1926
node:  3, Hub: 0.2934, Auth: 0.0941
node:  4, Hub: 0.3209, Auth: 0.1355
node:  5, Hub: 0.3175, Auth: 0.1873
node:  6, Hub: 0.1488, Auth: 0.1295
node:  7, Hub: 0.2084, Auth: 0.2692
node:  8, Hub: 0.2191, Auth: 0.2625
node:  9, Hub: 0.1572, Auth: 0.1972
node: 10, Hub: 0.1994, Auth: 0.2600
node: 11, Hub: 0.1467, Auth: 0.2429
node: 12, Hub: 0.1978, Auth: 0.2335
node: 13, Hub: 0.1290, Auth: 0.2493
node: 14, Hub: 0.1334, Auth: 0.2246
node: 15, Hub: 0.0367, Auth: 0.1010
node: 16, Hub: 0.0521, Auth: 0.1104
```

Рисунок В.6 - Частина показників авторитетності та посредницької оцінки кожної вершини у посилальному графу набору даних «Friendship»

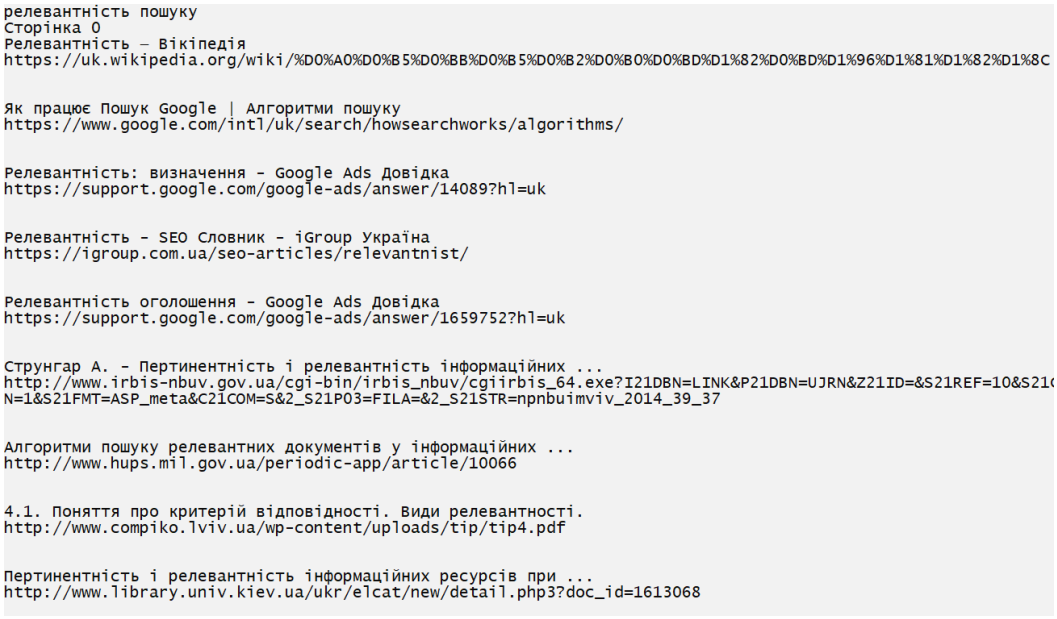


Рисунок В.7 - Список результатів реалізації гібридного метода ранжування після відправки запиту «Релевантність пошуку»

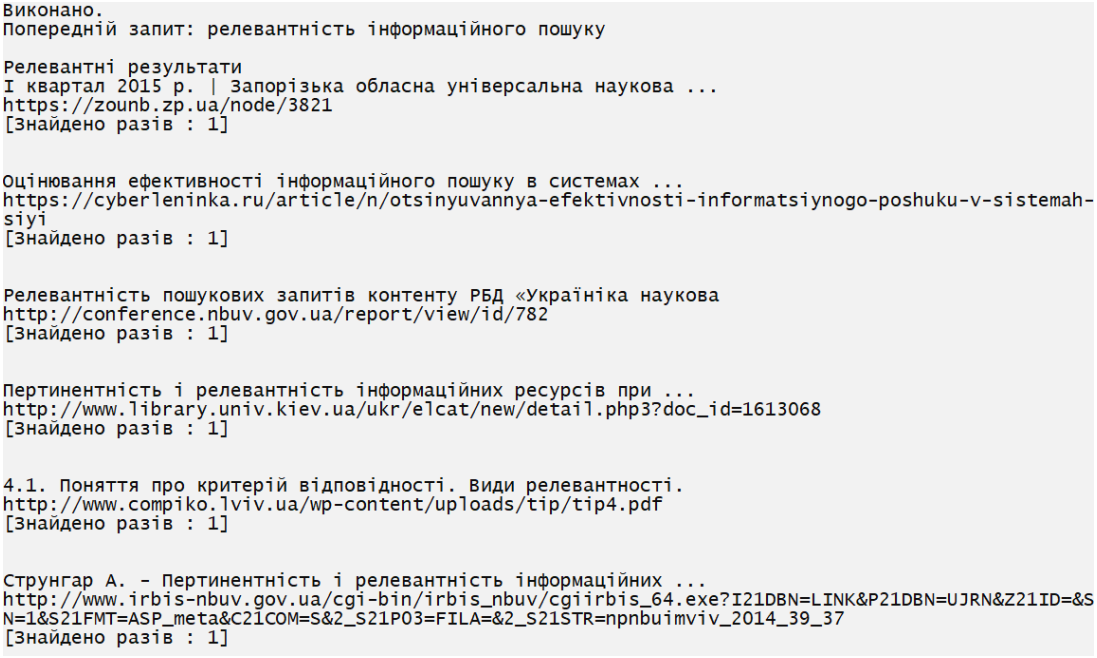


Рисунок В.8 - Список повторних результатів реалізації гібридного метода ранжування після відправки запиту «Релевантність інформаційного пошуку»

```
Виконано.
Попередній запит: підвищення релевантності інформаційного пошуку

Релевантні результати
4.1. Поняття про критерій відповідності. Види релевантності.
http://www.compiko.lviv.ua/wp-content/uploads/tip/tip4.pdf
[Знайдено разів : 2]


Релевантність - SEO Словник - iGroup Україна
https://igroup.com.ua/seo-articles/relevantnist/
[Знайдено разів : 2]


Релевантність — Вікіпедія
https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D0%B5%D0%B2%D0%B0%D0%BD%D1
[Знайдено разів : 2]


Использование онтологического описания предметной области ...
http://dspace.nbuv.gov.ua/handle/123456789/1336
[Знайдено разів : 1]


ПЕРСПЕКТИВИ ВДОСКОНАЛЕННЯ ІНФОРМАЦІЙНОГО ПОШУКУ
http://molodyvcheny.in.ua/files/journal/2017/4.4/25.pdf
[Знайдено разів : 1]


I квартал 2015 р. | Запорізька обласна універсальна наукова ...
https://zounb.zp.ua/node/3821
[Знайдено разів : 1]
```

Рисунок В.9 - Список повторних результатів реалізації гібридного метода ранжування після відправки запиту «Підвищення релевантності інформаційного пошуку»

```
Виконано.
Попередній запит: ефективність інформаційного пошуку

Релевантні результати
4.1. Поняття про критерій відповідності. Види релевантності.
http://www.compiko.lviv.ua/wp-content/uploads/tip/tip4.pdf
[Знайдено разів : 3]


Оцінювання ефективності інформаційного пошуку в системах ...
https://cyberleninka.ru/article/n/otsinyuvannya-efektivnosti-informatsiynogo-posl
siyi
[Знайдено разів : 2]


Релевантність - SEO Словник - iGroup Україна
https://igroup.com.ua/seo-articles/relevantnist/
[Знайдено разів : 2]


Релевантність — Вікіпедія
https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D0%B5%D0%B2%D0%B0%D0%BD%D1%82%D
[Знайдено разів : 2]


Показники ефективності інформаційного пошуку
http://elartu.tntu.edu.ua/handle/123456789/11069
[Знайдено разів : 1]


Использование онтологического описания предметной области ...
http://dspace.nbuv.gov.ua/handle/123456789/1336
[Знайдено разів : 1]
```

Рисунок В.10 - Список повторних результатів реалізації гібридного метода ранжування після відправки запиту «Ефективність інформаційного пошуку»

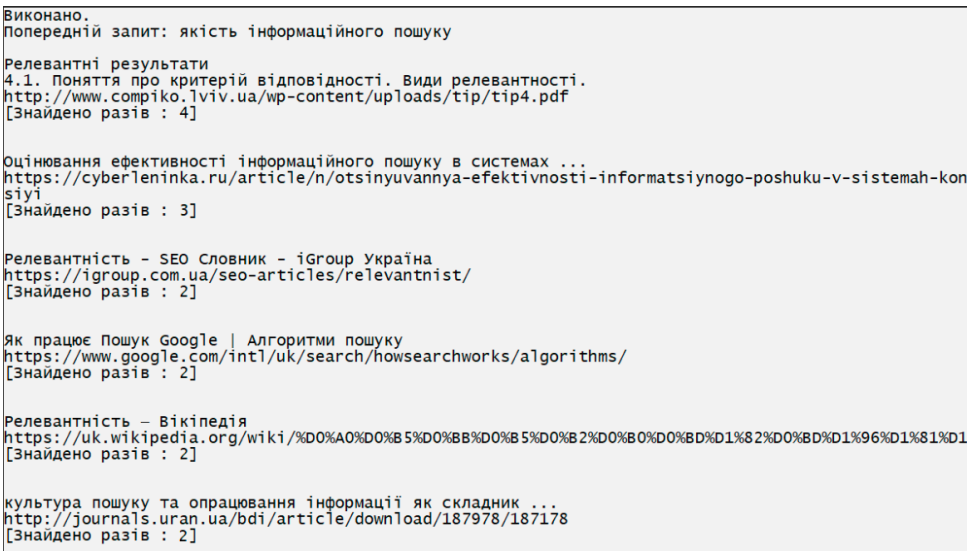


Рисунок В.11 - Список повторних результатів реалізаціїї гібридного метода ранжування після відправки запиту «Як збільшити ефективність інформаційного пошуку»

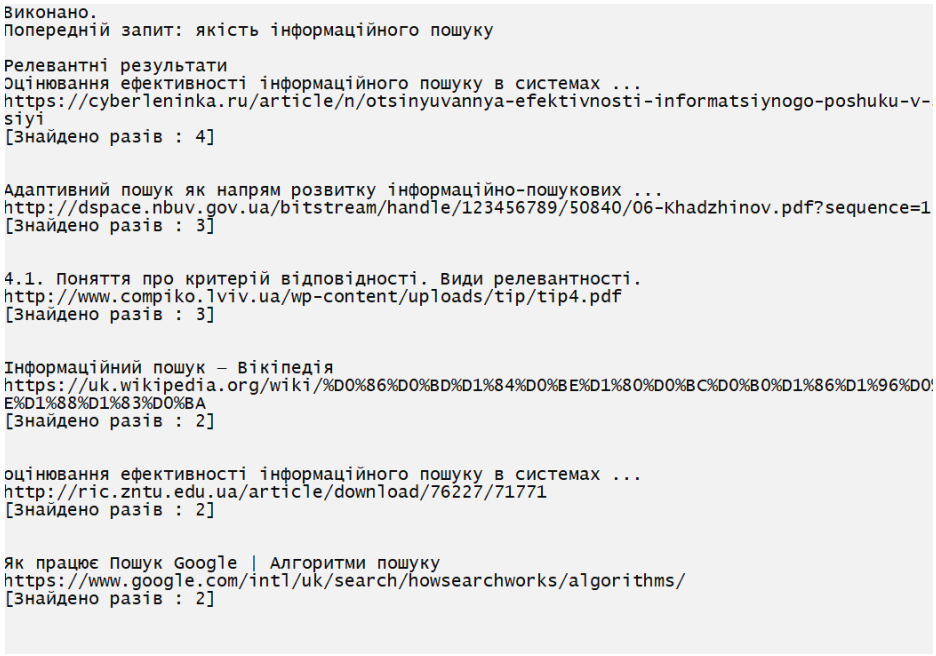


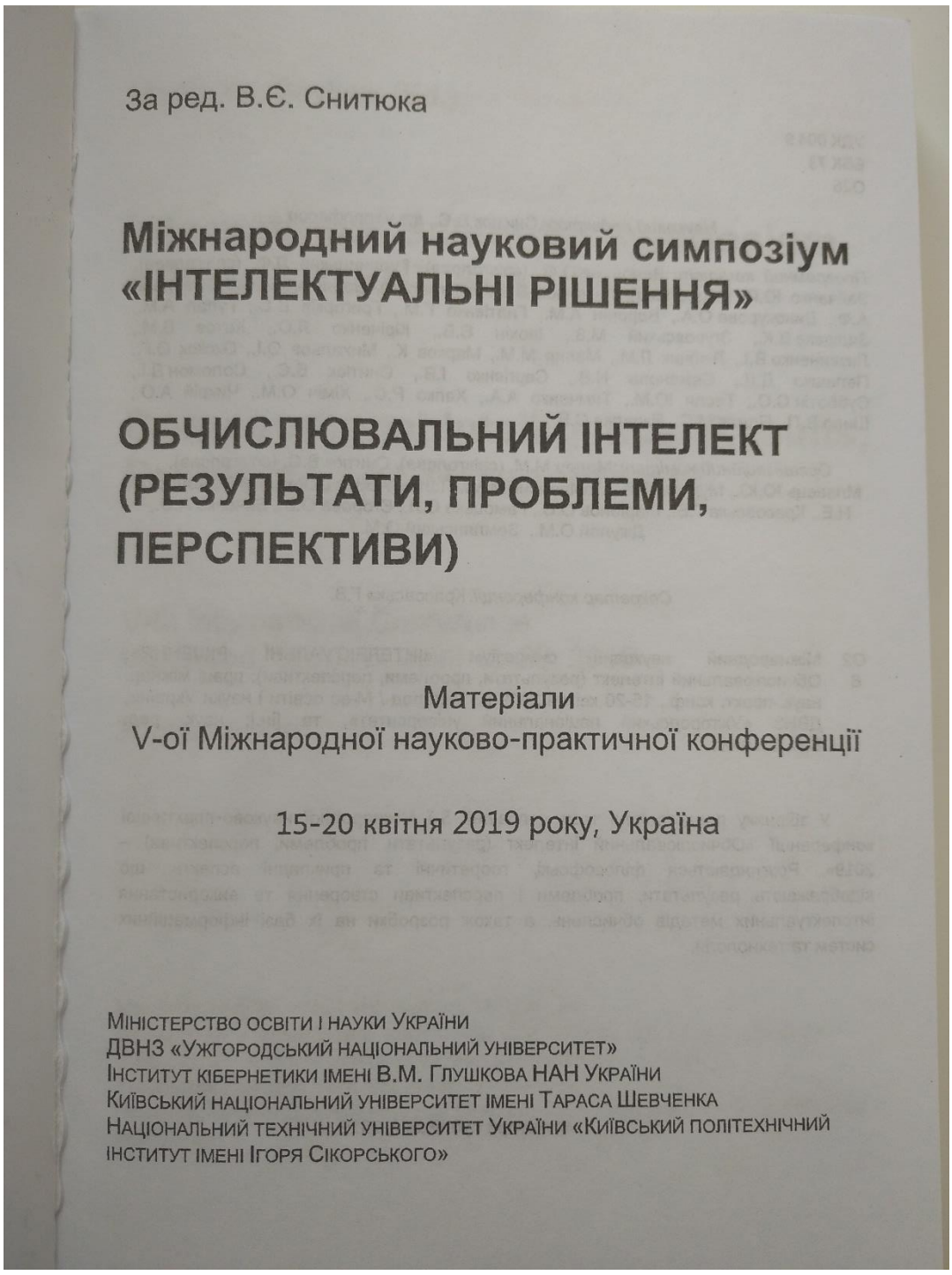
Рисунок В.12 - Список повторних результатів реалізаціїї гібридного метода ранжування після відправки запиту «Якість інформаційного пошуку»

Рисунки демонструють результати проведеного імітаційного моделювання вибраних та досліджених методів ранжування

ДОДАТОК Г

Тези доповіді «Гібридний метод ранжирування результатів запиту у пошукових системах», матеріали V-ої Міжнародної науково-практичної конференції «Обчислювальний інтелект (результати, проблеми, перспективи) - 2019»

Рисунок Г.1 – Обкладинка матеріалів конференції


За ред. В.Є. Снитюка

Міжнародний науковий симпозіум
«ІНТЕЛЕКТУАЛЬНІ РІШЕННЯ»

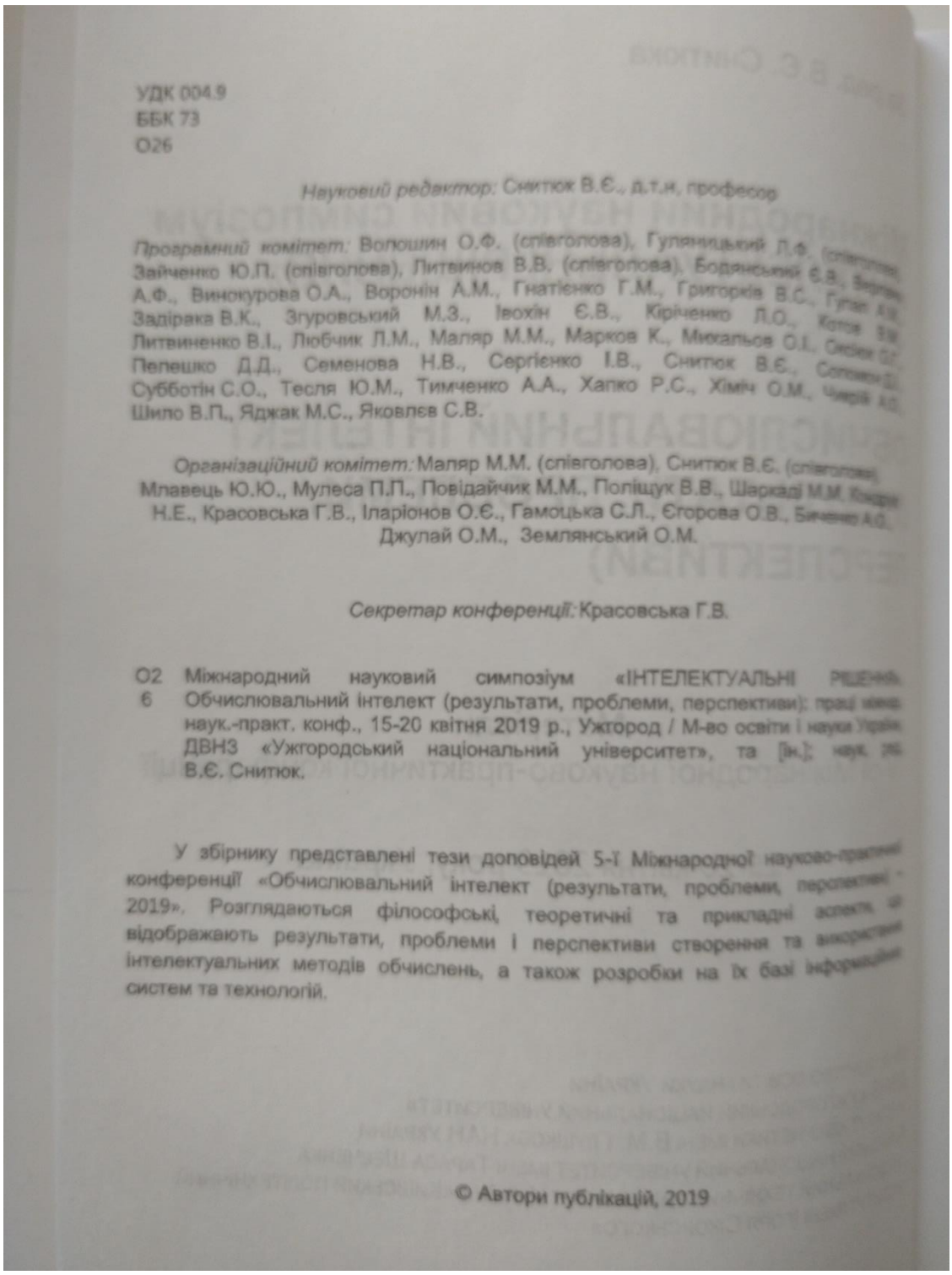
ОБЧИСЛЮВАЛЬНИЙ ІНТЕЛЕКТ
(РЕЗУЛЬТАТИ, ПРОБЛЕМИ,
ПЕРСПЕКТИВИ)

Матеріали
V-ої Міжнародної науково-практичної конференції

15-20 квітня 2019 року, Україна

Міністерство освіти і науки України
ДВНЗ «Ужгородський національний університет»
Інститут кібернетики імені В.М. Глушкова НАН України
Київський національний університет імені Тараса Шевченка
Національний технічний університет України «Київський політехнічний
інститут імені Ігоря Сікорського»

Рисунок Г.2 – Титульна сторінка матеріалів конференції



УДК 004.9
ББК 73
О26

Науковий редактор: Снитюк В.Є., д.т.н, професор

Програмний комітет: Волошин О.Ф. (співголова), Гуляницький Л.Ф. (співголова), Зайченко Ю.П. (співголова), Литвинов В.В. (співголова), Бодянський Є.В., Вергун А.Ф., Винокурова О.А., Воронін А.М., Гнатієнко Г.М., Григорків В.С., Гупал А.М., Задірака В.К., Згуровський М.З., Івохін Є.В., Кіріченко Л.О., Котов Ю.В., Литвиненко В.І., Любчик Л.М., Маляр М.М., Марков К., Михальов О.І., Онкіня О.І., Пелешко Д.Д., Семенова Н.В., Сергієнко І.В., Снитюк В.Є., Соловйов Д.І., Субботін С.О., Тесля Ю.М., Тимченко А.А., Хапко Р.С., Хімич О.М., Чумій А.О., Шило В.П., Яджак М.С., Яковлєв С.В.

Організаційний комітет: Маляр М.М. (співголова), Снитюк В.Є. (співголова), Млавець Ю.Ю., Мулеса П.П., Повідайчик М.М., Поліщук В.В., Шаркаді М.М., Кондрат Н.Е., Красовська Г.В., Іларіонов О.Є., Гамоцька С.Л., Єгорова О.В., Биченко А.О., Джулай О.М., Землянський О.М.

Секретар конференції: Красовська Г.В.

О2 Міжнародний науковий симпозіум «ІНТЕЛЕКТУАЛЬНІ РІШЕННЯ»
6 Обчислювальний інтелект (результати, проблеми, перспективи): праці між. наук.-практ. конф., 15-20 квітня 2019 р., Ужгород / М-во освіти і науки України, ДВНЗ «Ужгородський національний університет», та [ін.]; наук. ред. В.Є. Снитюк.

У збірнику представлені тези доповідей 5-ї Міжнародної науково-практичної конференції «Обчислювальний інтелект (результати, проблеми, перспективи) - 2019». Розглядаються філософські, теоретичні та прикладні аспекти, що відображають результати, проблеми і перспективи створення та використання інтелектуальних методів обчислень, а також розробки на їх базі інформаційних систем та технологій.

Рисунок Г.3 – Видаткова інформація матеріалів конференції

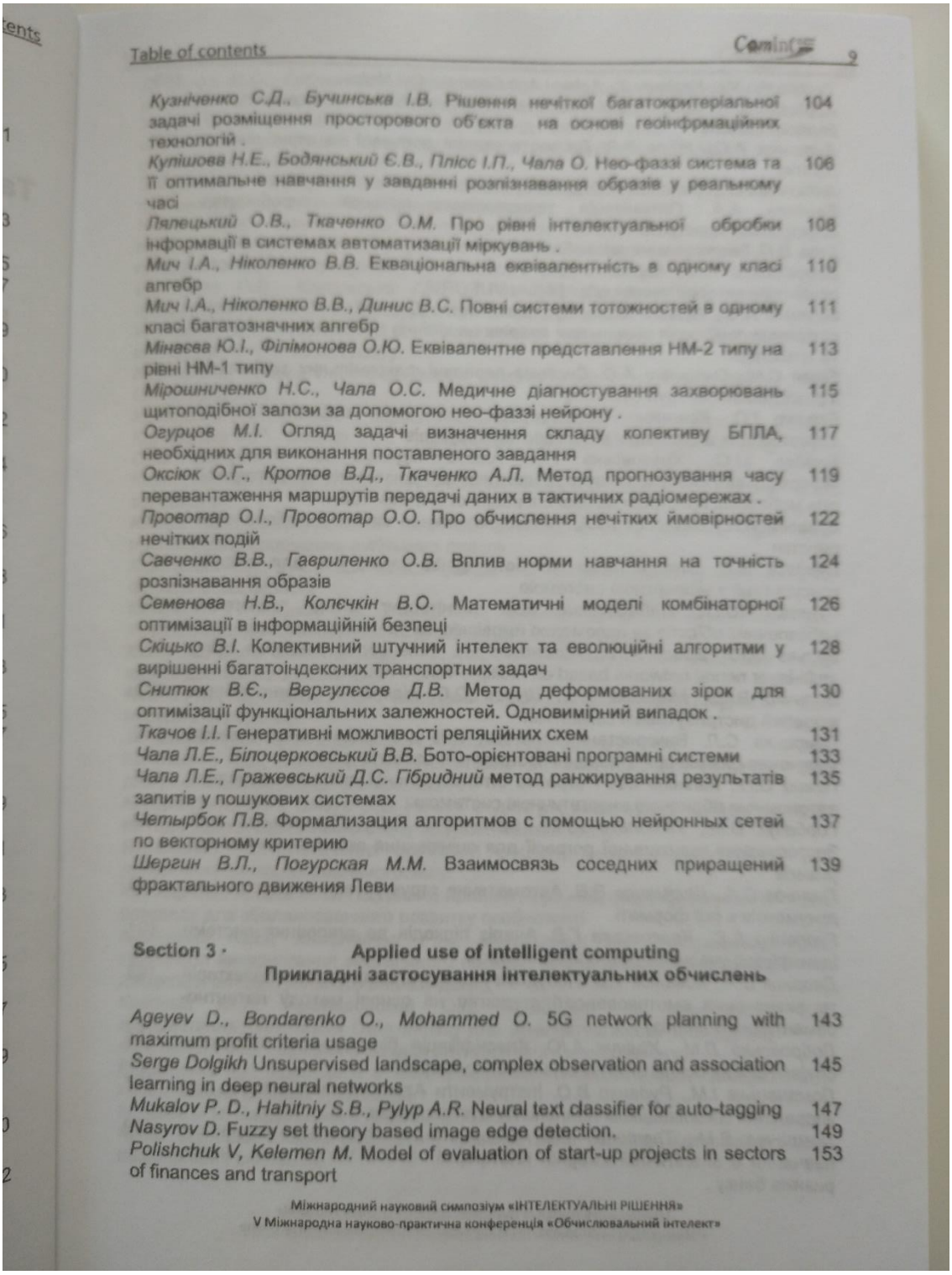
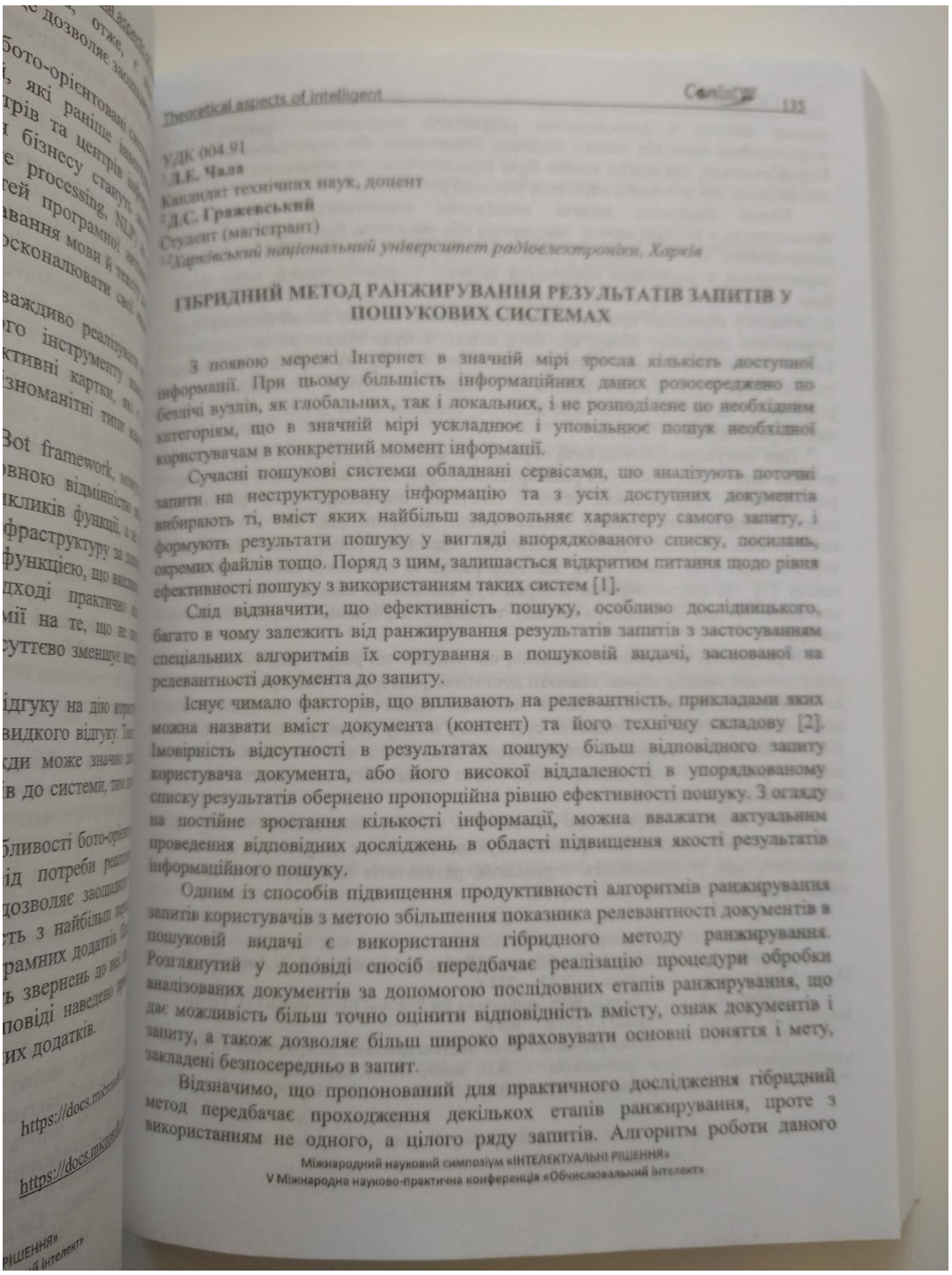


Рисунок Г.4 – Частина змісту матеріалів конференції



Theoretical aspects of intelligent Computer 135

УДК 004.91

[1]Л.Е. Чала
Кандидат технічних наук, доцент
[2]Д.С. Гражевський
Студент (магістрант)
[1,2]Харківський національний університет радіоелектроніки, Харків

ГІБРИДНИЙ МЕТОД РАНЖИРУВАННЯ РЕЗУЛЬТАТІВ ЗАПИТІВ У ПОШУКОВИХ СИСТЕМАХ

З появою мережі Інтернет в значній мірі зросла кількість доступної інформації. При цьому більшість інформаційних даних розосереджено по безлічі вузлів, як глобальних, так і локальних, і не розподілене по необхідним категоріям, що в значній мірі ускладнює і уповільнює пошук необхідної користувачам в конкретний момент інформації.

Сучасні пошукові системи обладнані сервісами, що аналізують поточні запити на неструктуровану інформацію та з усіх доступних документів вибирають ті, вміст яких найбільш задовольняє характеру самого запиту, і формують результати пошуку у вигляді впорядкованого списку, посилань, окремих файлів тощо. Поряд з цим, залишається відкритим питання щодо рівня ефективності пошуку з використанням таких систем [1].

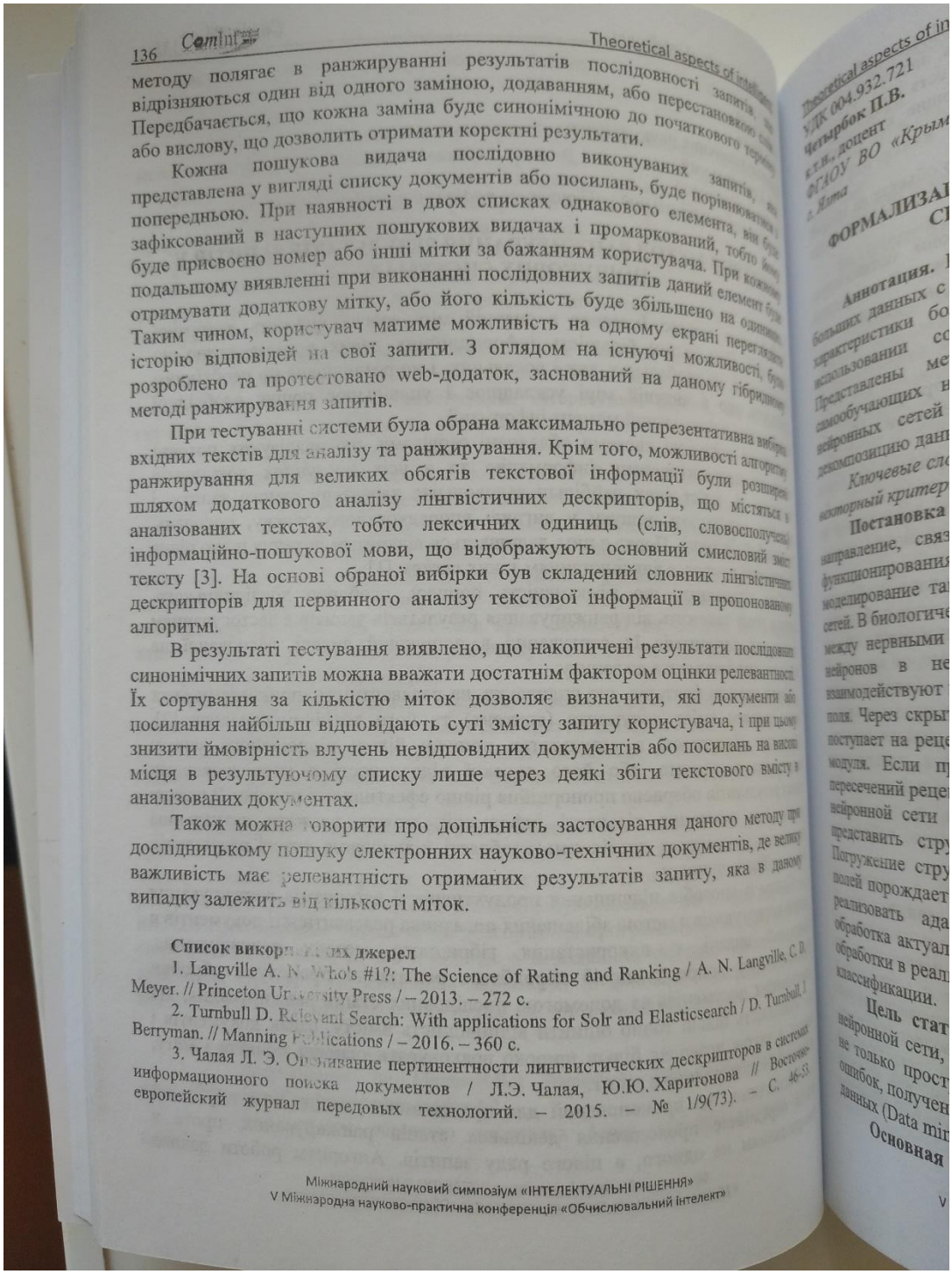
Слід відзначити, що ефективність пошуку, особливо дослідницького, багато в чому залежить від ранжирування результатів запитів з застосуванням спеціальних алгоритмів їх сортування в пошуковій видачі, заснованої на релевантності документа до запиту.

Існує чимало факторів, що впливають на релевантність, прикладами яких можна назвати вміст документа (контент) та його технічну складову [2]. Імовірність відсутності в результатах пошуку більш відповідного запиту користувача документа, або його високої віддаленості в упорядкованому списку результатів обернено пропорційна рівню ефективності пошуку. З огляду на постійне зростання кількості інформації, можна вважати актуальним проведення відповідних досліджень в області підвищення якості результатів інформаційного пошуку.

Одним із способів підвищення продуктивності алгоритмів ранжирування запитів користувачів з метою збільшення показника релевантності документів в пошуковій видачі є використання гібридного методу ранжирування. Розглянутий у доповіді спосіб передбачає реалізацію процедури обробки аналізованих документів за допомогою послідовних етапів ранжирування, що дає можливість більш точно оцінити відповідність вмісту, ознак документів і запиту, а також дозволяє більш широко враховувати основні поняття і мету, закладені безпосередньо в запит.

Відзначимо, що пропонований для практичного дослідження гібридний метод передбачає проходження декількох етапів ранжирування, проте з використанням не одного, а цілого ряду запитів. Алгоритм роботи даного

Рисунок Г.5 – Тези доповіді «Гібридний метод ранжирування результатів запитів у пошукових системах»



136 CamInt Theoretical aspects of intelligent

методу полягає в ранжируванні результатів послідовності запитів, що відрізняються один від одного заміною, додаванням, або перестановкою слів. Передбачається, що кожна заміна буде синонімічною до початкового терміну або вислову, що дозволить отримати коректні результати.

Кожна пошукова видача послідовно виконуваних запитів, що представлена у вигляді списку документів або посилань, буде порівнюватися з попередньою. При наявності в двох списках однакового елемента, він буде зафіксований в наступних пошукових видачах і промаркований, тобто йому буде присвоєно номер або інші мітки за бажанням користувача. При кожному подальшому виявленні при виконанні послідовних запитів даний елемент буде отримувати додаткову мітку, або його кількість буде збільшено на одиницю. Таким чином, користувач матиме можливість на одному екрані переглядати історію відповідей на свої запити. З оглядом на існуючі можливості, було розроблено та протестовано web-додаток, заснований на даному гібридному методі ранжирування запитів.

При тестуванні системи була обрана максимально репрезентативна вибірка вхідних текстів для аналізу та ранжирування. Крім того, можливості алгоритму ранжирування для великих обсягів текстової інформації були розширені шляхом додаткового аналізу лінгвістичних дескрипторів, що містяться в аналізованих текстах, тобто лексичних одиниць (слів, словосполучень) інформаційно-пошукової мови, що відображують основний смисловий зміст тексту [3]. На основі обраної вибірки був складений словник лінгвістичних дескрипторів для первинного аналізу текстової інформації в пропонованому алгоритмі.

В результаті тестування виявлено, що накопичені результати послідовних синонімічних запитів можна вважати достатнім фактором оцінки релевантності. Їх сортування за кількістю міток дозволяє визначити, які документи або посилання найбільш відповідають суті змісту запиту користувача, і при цьому знизити ймовірність влучень невідповідних документів або посилань на високі місця в результуючому списку лише через деякі збіги текстового вмісту в аналізованих документах.

Також можна говорити про доцільність застосування даного методу при дослідницькому пошуку електронних науково-технічних документів, де велику важливість має релевантність отриманих результатів запиту, яка в даному випадку залежить від кількості міток.

Список використаних джерел

1. Langville A. N. Who's #1?: The Science of Rating and Ranking / A. N. Langville, C. D. Meyer. // Princeton University Press / – 2013. – 272 с.

2. Turnbull D. Relevant Search: With applications for Solr and Elasticsearch / D. Turnbull, J. Berryman. // Manning Publications / – 2016. – 360 с.

3. Чалая Л. Э. Оценивание пертинентности лингвистических дескрипторов в системах информационного поиска документов / Л.Э. Чалая, Ю.Ю. Харитонова // Восточно-европейский журнал передовых технологий. – 2015. – № 1/9(73). – С. 46-53.

Рисунок Г.6 – Продовження тез доповіді «Гібридний метод ранжирування результатів запитів у пошукових системах»

ДОДАТОК Ґ

Тези доповіді «Hybrid Method of Ranking Query Results in Search Engines», Proceedings of International Conference on Software Engineering

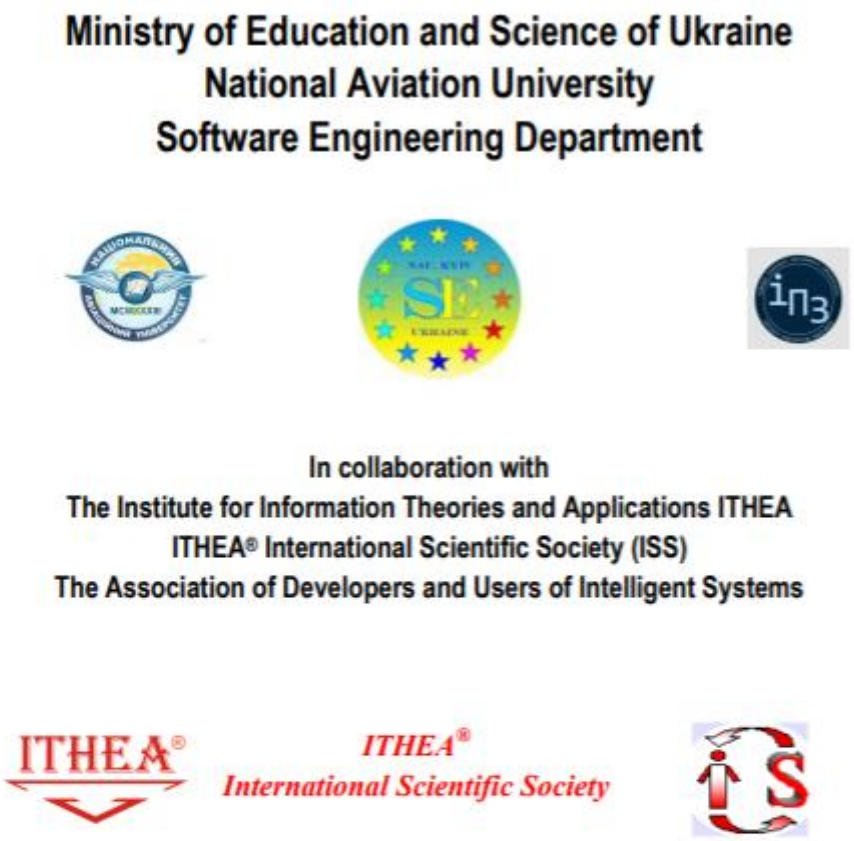


Рисунок Ґ.1 – Титульна сторінка матеріалів конференції

Рисунок Ґ.2 – Частина змісту матеріалів конференції

Section: Software Engineering Application Domains

HYBRID METHOD OF RANKING QUERY RESULTS IN SEARCH ENGINES

L. Chala, D. Hrazhevskyi

Kharkiv National University of Radioelectronics, Ukraine

Introduction

With the advent of the Internet, the available information amount has greatly increased. At the same time, most information data are dispersed in a variety of nodes, both global and local, and is not divided into the necessary categories, meaning, purpose, which greatly complicates and slows down the search for the necessary information at a particular time.

Search engines have been established to resolve these issues. Such services, when creating a request for unstructured information to them, choose from all available documents those whose content most satisfies the content of the request itself, and return them as results – in the form of a list, links, individual files.

At the same time, a meaningful ordering of the results positions is required. Its absence similarly leads to an increase in the amount of time required for the user to find the required documents. It is also necessary to consider the size of the documents collections involved in the information search. In the case of large collections, it is not practical to view all the results and choose the most appropriate ones. Therefore, we can conclude about the openness of the question about the level of search efficiency, using search engines.

Ranking

The effectiveness of search [Langville, 2013], especially research, largely depends on special algorithms for sorting query results, based on the degree of compliance of each document directly to the query content – so-called relevance. This process is called ranking. The probability of the absence in the search results of a more appropriate user request document, or its high remoteness in an ordered list of results is inversely proportional to the level of search efficiency. Considering the constant increase in the amount of information, it can be considered relevant research and development in this area.

Рисунок Ґ.3 – Тези доповіді «Hybrid Method of Ranking Query Results in Search Engines»

Obviously, the use of only one indicator will not be enough to ensure high relevance. There are many factors that affect the relevance, examples of which can be called the content of the document, its technical component [Turnbull, 2016], the frequency of its opening, location. For example, the well-known measure TF-IDF, used to determine the importance of a word in the context of a document, uses both the frequency of the term and the frequency of the document – the number of documents containing the term [Robertson, 2004]. Also, using this measure, it is possible to represent documents in the form of vectors (vector search model), compare and rank them using commonly known metrics - Euclidean distance, cosine measure [Shahzad and Ramsha, 2018].

Different from the vector model of search and ranking based on metrics is ranked search, focused on the Internet and based on links. Considering the reference structure of the Internet when ranking can give a significant improvement in search results [Langville and Meyer, 2004]. The reference structure consists of documents submitted by sites, their pages, and hyperlinks between them. It can be represented as a directed graph with web pages as nodes and hyperlinks as edges. The link analysis methods in the web graph is based on two assumptions that the text of the link that pointing to page B is a good description of page B. and a hyperlink from page A to page B is the recognition of the authority of page B by the creator of page A.

The main methods of reference ranking [Patel and Patel, 2015], based on which create new methods, or use modified, as, for example, global search engines Google, Baidu, are RankPage and Hypertext Induced Topic Search (HITS).

Hybrid method of ranking

One of the most effective ways to increase the productivity of ranking is the use of a hybrid ranking method [Чала, 2019]. When using this method, it is assumed that the collections of documents pass through several successive stages of ranking. As a result, the resulting lists become much more accurate, because with each such pass, the conformity assessment of the content, features of the documents and the request is clarified. Also, it is possible to cover more widely the concepts and purpose laid down directly in the request, to consider new factors.

The hybrid method proposed for practical research also involves the passage of several stages of ranking, but using not one autonomous, but several queries. The idea of this method is to rank the results of a sequence of queries that differ from each other by replacing, adding, or rearranging words. It is assumed that each change will be synonymous to its original expression to obtain more correct results.

A web application was created that allows the user to use a hybrid ranking method when searching. This application was written using .NET and ASP.NET MVC. The algorithm of the program, as well as the method, begins with sending a http request with the content in the form of a proposal to the server of

Рисунок Ґ.4 – Продовження тез доповіді «Hybrid Method of Ranking Query Results in Search Engines»

a remote search engine in order to receive an http-response. The user has the option to enter his query in the search bar of the application. In this application the library Google Custom Search API was used, which allows to use Google search servers, and with which you can send up to a hundred requests a day.

Most search engines use the PageRank reference ranking method to search for the required documents, often modified to consider some other factors as well as vector models. The resulting http response in the form of a ranked list of results is parsed and recorded in the NoSQL database in the form of links that were received first. After that, the links from the database are displayed to the user, and user enters a query similar to the previous one in meaning into the search bar of the application, but the content of which is slightly changed, in order to obtain a result list from both new and previously obtained elements.

Each new search output – http response in the form of a list – of similar consecutive queries will be compared with the previous one in the database. If the two lists have the same item, it will be recorded in the subsequent search results, and marked – it will be assigned a number or other labels at the request of the user, which is also reflected in the database. Each time a subsequent query is detected, the item will receive an additional label, or its number will be increased by one. Thus, the user will be able to view the history of responses to their queries on one screen. Such links will be given to the user first, and ordered by the number of their labels, number, which makes it possible to immediately see the most suitable result.

Conclusion

It was found that the accumulated results of successive synonymous queries can be considered indicators of increasing the productivity of the ranking, and enough factors of relevance. Sorting them by the number of labels allows to see which documents or links most correspond to a certain meaning, the essence of the content of the user's request, and at the same time reduce the likelihood of inappropriate documents or links to high places in the resulting list only because of some coincidence of the text content. We can also talk about the feasibility of this method in the researches, where the relevance of the results of the query is of great importance, which, in this case, depends on the number of labels.

Bibliography

[Langville, 2013] A.N. Langville. Who's #1?: The Science of Rating and Ranking. Ed. A.N. Langville and C.D. Meyer. Princeton University Press, 2013.

[Turnbull, 2016] D. Turnbull. Relevant Search: With applications for Solr and Elasticsearch. Ed. D. Turnbull and J. Berryman. Manning Publications, 2016.

Рисунок Ґ.5 – Кінець тез доповіді «Hybrid Method of Ranking Query Results in Search Engines»

ДОДАТОК Д

Відомість атестаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	133 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів __

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Гражевський Д.С.		06.05	Дослідження методів ранжування результатів запитів у пошукових системах	Шифр групи	Код напр./спец.
Перевірив		Чала Л.Е		11.05		СШІм-18-2	122
Н.контр.				12.05		ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.		18.05			