Συστήματα Ανάκτησης Πληροφοριών

Προγραμματιστική Εργασία: Μέρος Α

Σοφία-Ζωή Σωτηρίου, p3210192

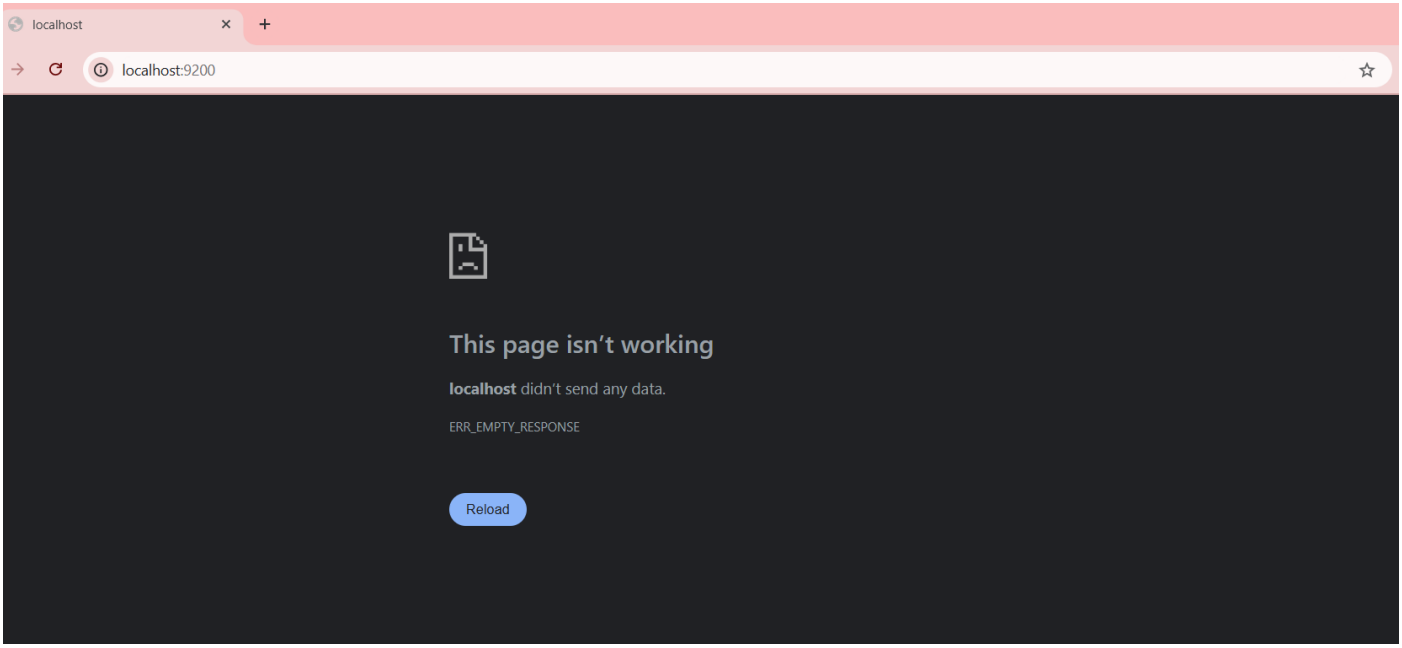
Εγκατάσταση απαραίτητων components

Πρώτο μέλημα ήταν η βεβαίωση ότι υπάρχουν και έχουν ενημερωθεί η python, το anaconda (ώστε να μπορέσει να χρησιμοποιηθεί το jupyter notebook) και το pip (ώστε να κατέβουν τα απαραίτητα libraries που θα χρησιμοποιηθούν κατά τη διάρκεια του project).

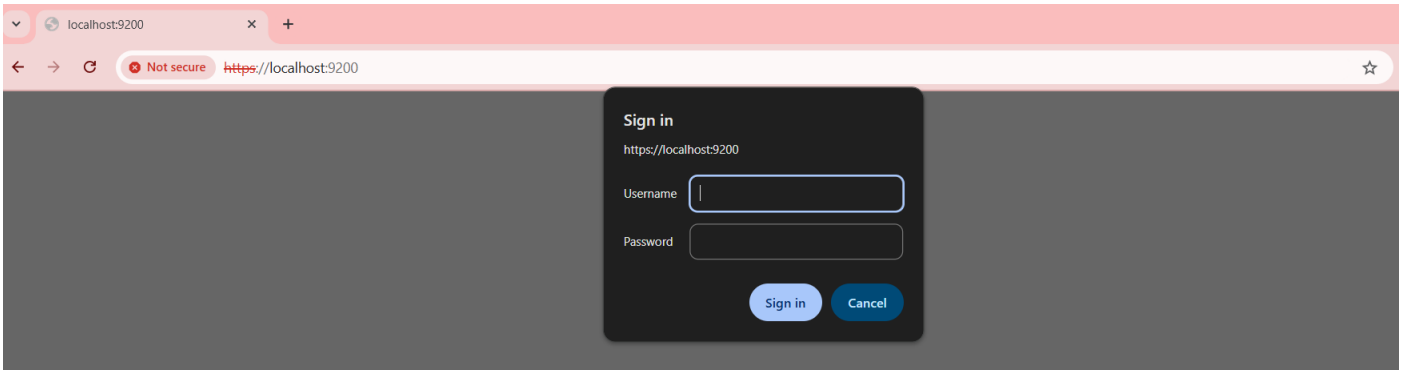
```
Microsoft Windows [Version 10.0.26100.3775]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sofia>python --version
Python 3.13.3

C:\Users\sofia>conda --version
conda 25.1.1

C:\Users\sofia>pip --version
pip 25.0.1 from C:\Users\sofia\AppData\Local\Programs\Python\Python313\Lib\site-packages\pip (python 3.13)
```

Έπειτα, έγινε η εγκατάσταση του ElasticSearch, όπου δημιουργήθηκε το εξής πρόβλημα σύνδεσης :

Με τη χρήση του http://localhost:9200 δεν έγινε η σύνδεση.



Κοιτώντας τα logs του ElasticSearch, διαπιστώθηκε ότι δεν δεχόταν το http πρωτόκολλο, ως μη ασφαλές (δεν δεχόταν το αυτόματο -self-signed- certificate που δημιουργούταν). Επομένως, δοκιμάστηκε η χρήση της https://localhost:9200 διεύθυνσης, κατά τη οποία στη παρακάτω οθόνη τα default credentials (username: elastic, password: changeme) δεν έγιναν δεκτά. (Το στιγμιότυπο τραβήχτηκε αφού δόθηκαν τα credentials και πατήθηκε sign in)



Επομένως, αλλάχθηκε ο κωδικός του χρήστη elastic:

```
C:\Users\sofia\Downloads\elasticsearch-9.0.1-windows-x86_64\elasticsearch-9.0.1\bin>elasticsearch-reset-password -u elas
tic
This tool will reset the password of the [elastic] user to an autogenerated value.
The password will be printed in the console.
Please confirm that you would like to continue [y/N]y


Password for the [elastic] user successfully reset.
New value: VRid-7JuSK7w_UweAdT*
```

Αλλά το πρόβλημα του certificate παρέμενε και στο browser και στο command line (αφού εκεί έγιναν τα αρχικά πειράματα, πριν τη συγγραφή κώδικα στο jupyter), οπότε προστέθηκε το flag -k, με το οποίο ουσιαστικά το curl προσπερνάει τον έλεγχο του SSL certification, πετυχαίνοντας έτσι τη πρώτη είσοδο στο ElasticSearch

```
C:\Users\sofia\Downloads\elasticsearch-9.0.1-windows-x86_64\elasticsearch-9.0.1\bin>curl -u elastic:VRid-7JuSK7w_UweAdT*
 -k https://localhost:9200
{
  "name" : "LOLLYS",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "guGsRX0mTbqnmWP_U8liTg",
  "version" : {
    "number" : "9.0.1",
    "build_flavor" : "default",
    "build_type" : "zip",
    "build_hash" : "73f7594ea00db50aa7e941e151a5b3985f01e364",
    "build_date" : "2025-04-30T10:07:41.393025990Z",
    "build_snapshot" : false,
    "lucene_version" : "10.1.0",
    "minimum_wire_compatibility_version" : "8.18.0",
    "minimum_index_compatibility_version" : "8.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Σημειώνεται σε αυτό το σημείο ότι η συνέχεια της χρήσης του HTTPS πρωτοκόλλου είναι συνειδητή, καθώς παρά το επιπρόσθετο flag, με τη χρήση του HTTP πρωτοκόλλου η σύνδεση ήταν αδύνατη.

Κώδικας

(Με δεδομένο ότι ο αριθμός μητρώου μου είναι ζυγός, χρησιμοποιείται η συλλογή scidocs)

Μετά τις διαπιστώσεις που έγιναν κατά την εγκατάσταση του elasticsearch, τροποποιήθηκε αναλόγως και ο κώδικας, ώστε να προσπερνιέται ο έλεγχος του certificate και να μην εμφανίζονται τα σχετικά warnings, αλλά και σε περίπτωση σφάλματος να εμφανίζεται το ανάλογο exception, επιτυγχάνοντας τελικά το επιθυμητό αποτέλεσμα, δηλαδή επιτυχή σύνδεση, με πράσινο cluster status.

```
#test test one two
!curl -u elastic:VRid-7JuSK7w_UweAdT* -k https://localhost:9200
```

✓ 0.8s

```
{
  "name" : "LOLLYS",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "guGsRX0mTbqnmWP_U8liTg",
  "version" : {
    "number" : "9.0.1",
    "build_flavor" : "default",
    "build_type" : "zip",
    "build_hash" : "73f7594ea00db50aa7e941e151a5b3985f01e364",
    "build_date" : "2025-04-30T10:07:41.393025990Z",
    "build_snapshot" : false,
    "lucene_version" : "10.1.0",
    "minimum_wire_compatibility_version" : "8.18.0",
    "minimum_index_compatibility_version" : "8.0.0"
  },
  "tagline" : "You Know, for Search"
}
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed

  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100   530  100   530    0     0   1149      0 --:--:-- --:--:-- --:--:--  1152
```

```
import logging
import warnings
from elasticsearch.exceptions import AuthenticationException, NotFoundError

warnings.filterwarnings("ignore")
logging.basicConfig(level=logging.ERROR)

client = Elasticsearch("https://localhost:9200", basic_auth=("elastic", "VRid-7JuSK7w_UweAdT*"), verify_certs=False)

try:
    health = client.cluster.health()
    print("Connection succeeded, cluster health:", health)
except AuthenticationException:
    print("Username and passcode error")
except NotFoundError:
    print("Elasticsearch not found. Server is not at this address")
except Exception as e:
    print(f"An error occurred: {e}")
```

✓ 0.2s

```
Connection succeeded, cluster health: {'cluster_name': 'elasticsearch', 'status': 'green', 'timed_out': False, 'number_of_nodes': 1,
```

Ακολουθώντας τον κώδικα του εργαστηρίου, το scidocs/corpus.jsonl μετατρέπεται σε ένα dictionary, από το οποίο αρχικά για να υπάρχει μια εικόνα των δεδομένων και της δομής τους, εμφανίστηκαν το πλήθος των κειμένων, και τα κλειδιά του πρώτου κειμένου. Αφού διαπιστώθηκε ότι το κλειδί metadata είναι και αυτό λεξικό, τροποποιήθηκε ο κώδικας ώστε τελικά να εμφανίζει το πλήθος των κειμένων, όλα τα κλειδιά του κάθε κειμένου και τέλος να εμφανίζεται όλο το πρώτο κείμενο σε ζευγάρια κλειδί-τιμή.

```python
all_json_data = []
with open('read/scidocs/corpus.jsonl', "r", encoding="utf-8") as file:
    for line in file:
        all_json_data.append(json.loads(line))

example = all_json_data[0]

print(f" number of texts: {len(all_json_data)}")
print("keys: ", end=' ')
for key in example.keys() :
    print(key, end=' ')

print("\nkeys in metadata: ", end=' ')
for key in example['metadata'].keys() :
    print(key, end=' ')

print('\n')
for i, (key, value) in zip(range(3), example.items()):
    print(f"{key}: {value}")

for key, value in example["metadata"].items() :
    print(f"{key}: {value}")
```

✓ 1.6s

```
 number of texts: 25657
keys:  _id title text metadata
keys in metadata:  authors year cited_by references

_id: 632589828c8b9fca2c3a59e97451fde8fa7d188d
title: A hybrid of genetic algorithm and particle swarm optimization for recurrent network design
text: An evolutionary recurrent network which automates the design of recurrent neural/fuzzy networks using a new evolutionary learning algorithm is proposed
authors: ['1725986']
year: 2004
cited_by: ['93e1026dd5244e45f6f9ec9e35e9de327b48e4b0', '870cb11115c8679c7e34f4f2ed5f469badedee37', '7ee0b2517cbda449d73bacf83c9bb2c96e816da7', '97ca96b2a60b0
references: ['57fdc130c1b1c3dd1fd11845fe86c60e2d3b7193', '51317b6082322a96b4570818b7a5ec8b2e330f2f', '2a047d8c4c2a4825e0f0305294e7da14f8de6fd3', 'e105fa3107
```

Όπως φαίνεται και από την εικόνα, κάθε κείμενο περιέχει ένα λεξικό με τα metadata, εκ των οποίων τα δύο πεδία περιέχουν πολλή πληροφορία και ίσως και λίγο περιττή, οπότε ο κώδικας του format των δεδομένων αρχικά τα αφαιρούσε, αλλά με πειράματα που έγιναν κατά τα queries, φάνηκε ότι όταν τα δεδομένα αυτά χρησιμοποιήθηκαν στην αναζήτηση τα αποτελέσματα ήταν καλύτερα. Ο τρόπος αναζήτησης θα εξηγηθεί πιο αναλυτικά παρακάτω, αλλά εδώ παρατίθενται τα αποτελέσματα της αναζήτησης χωρίς τα πεδία cited_by και references (αριστερά) και έπειτα με αυτά τα πεδία (δεξιά). Όπως φαίνεται, όλες οι μετρικές είχαν χειρότερη απόδοση χωρίς αυτά τα πεδία

```
num_rel                 all     4928
num_rel_ret             all     1142
map                     all     0.0449
gm_map                  all     0.0019
```

```
P_5                     all     0.0374
P_10                    all     0.0441
P_15                    all     0.0504
P_20                    all     0.0571
```

```
num_rel                 all     4928
num_rel_ret             all     1828
map                     all     0.0829
gm_map                  all     0.0109
```

```
P_5                     all     0.0652
P_10                    all     0.0748
P_15                    all     0.0826
P_20                    all     0.0914
```

Έτσι, το τελικό formatting που έγινε στα δεδομένα ήταν απλά να καταργηθεί το υπο-λεξικό metadata για να διευκολυνθεί η αναζήτηση, διατηρώντας όλα τα πεδία όπως ήταν. Έτσι,στο επόμενο κελί δημιουργείται μια νέα λίστα formatted_data, με τα πεδία uniqueID, title, text, authors, year, cited_by και references. Το πεδίο uniqueID μετονομάστηκε έτσι από _id γιατί κατά τη δημιουργία ευρετηρίου δημιουργούταν σφάλμα γιατί κάθε doc στο λεξικό έχει αυτόματα ένα πεδίο με όνομα _id.

Επομένως, για τη διευκόλυνση της διαδικασίας και για ένα πιο σίγουρο και ευανάγνωστο αποτέλεσμα, δημιουργήθηκε μια νέα λίστα με αυτά τα πεδία. Τέλος, γίνεται εμφάνιση του πρώτου κειμένου για να επιβεβαιωθεί η επιτυχία της μετατροπής.

```python
formatted_data = []
for text in all_json_data:
    placeholder = {}
    placeholder["uniqueID"] = text["_id"]
    placeholder["title"] = text["title"]
    placeholder["text"] = text["text"]
    placeholder["authors"] = text["metadata"]["authors"]
    placeholder["year"] = text["metadata"]["year"]
    placeholder["cited_by"] = text["metadata"]["cited_by"]
    placeholder["references"] = text["metadata"]["references"]

    formatted_data.append(placeholder)

example = formatted_data[0]

print("keys: ", end=' ')
for key in example.keys() :
    print(key, end=' ')

print('\n')
for key, value in example.items():
    print(f"{key}: {value}")
```

✓ 0.0s

```
keys:  uniqueID title text authors year cited_by references

uniqueID: 632589828c8b9fca2c3a59e97451fde8fa7d188d
title: A hybrid of genetic algorithm and particle swarm optimization for recurrent network design
text: An evolutionary recurrent network which automates the design of recurrent neural/fuzzy networks using a new evolutic
authors: ['1725986']
year: 2004
cited_by: ['93e1026dd5244e45f6f9ec9e35e9de327b48e4b0', '870cb11115c8679c7e34f4f2ed5f469badedee37', '7ee0b2517cbda449d73bac
references: ['57fdc130c1b1c3dd1fd11845fe86c60e2d3b7193', '51317b6082322a96b4570818b7a5ec8b2e330f2f', '2a047d8c4c2a4825e0f6
```

Στη συνέχεια ακολουθούν δυο κελιά με σκοπό τη δημιουργία του ευρετηρίου με βάση τα υπάρχοντα πεδία και τη διαγραφή του ευρετηρίου (και έλεγχο ότι δεν υπάρχουν ευρετήρια στον client) αν αυτό χρειαστεί. Στο πείραμα διαγραφής του Index, φαίνεται ότι διαγράφηκε επιτυχώς, αφού μετά το μόνο Index που βρέθηκε είναι το security-7 που είναι της ElasticSearch και όχι προς χρήση από το κοινό .

```python
mapping = {
    "settings": {
        "similarity": {
            "default": {
                "type":"BM25",
                "k1" : 1.4,
                "b" : 0.6
            }
        },
        "analysis": {
            "analyzer": {
                "default": {
                    "type": "english"
                },
                "default_search": {
                    "type": "english"
                }
            }
        }
    },
    "mappings": {
        "properties": {
        "title": {
            "type": "text",
            "copy_to": "allContent",
            "similarity": "default"
        },
        "text": {
            "type": "text",
            "copy_to": "allContent",
            "similarity": "default"
        },
        "authors": {
            "type": "text",
            "copy_to": "allContent",
            "similarity": "default"
        },
        "year": {
            "type": "date",
            "format": "yyyy"
        },
        "cited_by": {
            "type": "text",
            "copy_to": "allContent",
            "similarity": "default"
        },
        "references": {
            "type": "text",
            "copy_to": "allContent",
            "similarity": "default"
        },
        "allContent": {
            "type": "text",
            "similarity": "default"
        }
        }
    }
}

# create an index with the configuration above
client.indices.create(index='indexino', body=mapping)
```

✓ 1.5s

```
ObjectApiResponse({'acknowledged': True, 'shards_acknowledged': True, 'index': 'indexino'})
```

```python
client.indices.delete(index="scidocs")

indices = client.indices.get_alias(index="*")

if indices:
    print("Indices found:")
    for index_name in indices:
        print(f" - {index_name}")
else:
    print("No indices found.")
```

✓ 0.3s

```
Indices found:
 - .security-7
```

Στη συνέχεια τοποθετήθηκαν όλα τα κείμενα στο ευρετήριο και μετά εμφανίστηκε το πλήθος των κειμένων του ευρετηρίου, που ήταν όσα και τα κείμενα του corpus.jsonl, οπότε δεν υπήρχαν απώλειες

```python
from elasticsearch.helpers import bulk

documents = []
for doc in formatted_data:
    docID = doc.get('uniqueID')
    doc.pop('uniqueID')
    documents.append({
        "_index": "indexino",
        "_id": docID,
        "_source": doc
    })

# Indexing με helpers.bulk
bulk(client, documents, refresh=True)

count = client.count(index='indexino')["count"]

print(f"Index '{'indexino'}' contains {count} documents.")
```

✓ 1m 45.7s

```
Index 'indexino' contains 25657 documents.
```

Και μετά σύμφωνα με τον κώδικα του εργαστηρίου εμφανίστηκαν όλα τα αποτελέσματα με έναν πιο τυποποιημένο τρόπο ώστε να φανεί και ότι έχουν τη σωστή δομή και ότι έχουν περαστεί σωστά (δεν έγινε κάποιο σφάλμα κατά τη προεπεξεργασία). (Τα αποτελέσματα έχουν κοπεί για λόγους αναγνωσιμότητας).

```python
def pretty_search_response(response):
    if len(response["hits"]["hits"]) == 0:
        print("Your search returned no results.")
    else:
        for hit in response["hits"]["hits"]:
            id = hit["_id"]
            score = hit["_score"]
            title = hit["_source"]["title"]
            text = hit["_source"]["text"]
            authors = hit["_source"]["authors"]
            year = hit["_source"]["year"]
            cited_by = hit["_source"]["cited_by"]
            references = hit["_source"]["references"]

            pretty_output = f"\nID: {id}\nScore: {score}\ntitle: {title}\ntext: {text}\nauthors: {authors}\nyear: {year}\ncited_by: {cited_by}\nreferences: {references}"

            print(pretty_output)
```

✓ 0.0s

```python
search_results = client.search(index="indexino", query={"match_all": {}}, size=30)

# Print search results
pretty_search_response(search_results)
```

✓ 0.4s

```
ID: c44c877ad48f587baa49c79d528d1be6448256b1
Score: 1.0
title: Derivation of Human Trophoblast Stem Cells.
text: Trophoblast cells play an essential role in the interactions between the fetus and mother. Mouse trophoblast stem (TS) cells have been derived and used as the best in vitro
authors: ['5106786', '4588977', '2052070', '49147928', '31224071', '5247562', '49643722', '34935883', '2420587', '4295772']
year: 2018
cited_by: ['06f568a3d9cb3567a287b1ba5f187ed115a9ee25', 'a83d453875087a33e9a174c33ac50685d67f2f6b', '177b0a19af318258ce0e36b956926c10a119e394', 'cfc753b35a0a2e29981ad459cb28302c187
references: ['296f4da149e30efeef4e73e21a85186ce9addc01', '8200759086b02d5923f6b02d5319ce435da769b0', '9a78cc5dcf8b1bf2e7a557d8282bc26657a124e2', '681b62e08f942a6cbd9911a02fb26d76c

ID: 4cd6e2a4e9212c7e47e50bddec92f09095b67c2d
Score: 1.0
title: A Bidirectional-Switch-Based Wide-Input Range High-Efficiency Isolated Resonant Converter for Photovoltaic Applications
text: Modular photovoltaic (PV) power conditioning systems (PCSs) require a high-efficiency dc-dc converter stage capable of regulation over a wide input voltage range for maximum
authors: ['32306799', '9042262', '2411413', '2674816', '32808650']
```

Το επόμενο βήμα είναι η επεξεργασία των queries, οπότε στο επόμενο κελί γίνεται το import των queries, εμφανίζονται τα πεδία τους και εμφανίζονται έπειτα και όλα τα queries ώστε να υπάρχει μια εικόνα του συνόλου

```
queries = []
with open('read/scidocs/queries.jsonl', "r", encoding="utf-8") as file:
    for line in file:
        queries.append(json.loads(line))

example = queries[0]
print(f" number of texts: {len(queries)}")
print("keys: ", end=' ')
for key in example.keys() :
    print(key, end=' ')

print("\nkeys in metadata: ", end=' ')
for key in example['metadata'].keys() :
    print(key, end=' ')


for example in queries :

    print('\n')
    for i, (key, value) in zip(range(2), example.items()):
        print(f"{key}: {value}")

    for key, value in example["metadata"].items() :
        print(f"{key}: {value}")
```

✓ 0.2s

```
 number of texts: 1000
keys:  _id text metadata
keys in metadata:  authors year cited_by references

_id: 78495383450e02c5fe817e408726134b3084905d
text: A Direct Search Method to solve Economic Dispatch Problem with Valve-Point Effect
authors: ['50306438', '15303316', '1976596']
year: 2014
cited_by: ['38e78343cfd5c013decf49e8cf008ddf6458200f']
references: ['632589828c8b9fca2c3a59e97451fde8fa7d188d', '4cf296b9d4ef79b838dc565e6e84ab9b089613de', '86e8


_id: 7dcb308b9292a8bc87d6f7793d2ca5e0e19dfa40
text: Bearish-Bullish Sentiment Analysis on Financial Microblogs
authors: ['2243444', '32946276', '3349721']
year: 2017
cited_by: ['3a3f23919a04b4ff400f6a83a72eaff80bbdfb94', '4f3a038f1850f99fd169a7d703463592cb9e5582']
references: ['a08b72b072a3bf6256ddef222f6eacf014cc2b59', 'b1de806bf1d2f824bdb5d34ac96b062ce206e14e', '15b9
```

Βλέποντας όλα τα queries, διακρίθηκαν κάποιες ανωμαλίες στα κείμενα όπως κάποιες εμφανίσεις ειδικών χαρακτήρων, οι οποίοι πρέπει να ήταν απομεινάρια κάποιας κωδικοποίησης, οπότε έγινε μια μικρή προεπεξεργασία των queries και έπειτα, όπως και για τα δεδομένα (αφού και στις δύο περιπτώσεις υπήρχε ένα υπολεξικό metadata), μπήκαν σε ένα νέο σετ, με σκοπό την απομάκρυνση του υπολεξικού.

```
from nltk.corpus import stopwords
import re

for doc in queries :
    text = doc["text"]
    text = re.sub(r'[/()@|:?!.\-,+\d]', ' ', text)
    text = text.lower()
    doc["text"] = text
```

✓ 0.0s

```python
formatted_queries = []
for text in queries:
    placeholder = {}
    placeholder["uniqueID"] = text["_id"]
    placeholder["text"] = text["text"]
    placeholder["authors"] = text["metadata"]["authors"]
    if text["metadata"]["year"] :
        placeholder["year"] = text["metadata"]["year"]
    else :
        placeholder["year"] = "3000"
    placeholder["cited_by"] = text["metadata"]["cited_by"]
    placeholder["references"] = text["metadata"]["references"]
    formatted_queries.append(placeholder)

example = formatted_queries[1]

print("keys: ", end=' ')
for key in example.keys() :
    print(key, end=' ')

print('\n')
for key, value in example.items():
    print(f"{key}: {value}")
```

✓ 0.0s

```
keys:  uniqueID text authors year cited_by references

uniqueID: 7dcb308b9292a8bc87d6f7793d2ca5e0e19dfa40
text: bearish bullish sentiment analysis on financial microblogs
authors: ['2243444', '32946276', '3349721']
year: 2017
cited_by: ['3a3f23919a04b4ff400f6a83a72eaff80bbdfb94', '4f3a038f1850f99fd1
references: ['a08b72b072a3bf6256ddef222f6eacf014cc2b59', 'b1de806bf1d2f824
```

Πριν γίνουν τα queries, δημιουργείται μια συνάρτηση safe_query η οποία εφαρμόζεται στα πεδία με πολλά στοιχεία (authors, cited_by, references) για να ενώνει όλα τα στοιχεία σε ένα ενιαίο string.

```python
def safe_query(value):
    if isinstance(value, list):
        return " ".join(map(str, value))
    return str(value or "")
```

✓ 0.0s

Queries (όλα τα πειράματα έγιναν με size=20 και αφού αποφασίστηκε η τελική μορφή έγιναν τα queries για size = 30, 50)

Η αρχική μορφή των queries που χρησιμοποιήθηκε ήταν η ακόλουθη, με τη λογική ότι όλα τα πεδία πρέπει να είναι στο σωστό κείμενο και ότι έτσι θα αυξανόταν η ακρίβεια των αποτελεσμάτων, μια υπόθεση η οποία δεν επαληθεύτηκε από το αποτέλεσμα, καθώς δεν βρέθηκε κανένα αποτέλεσμα οπότε το αρχείο που δόθηκε στο trec_eval ήταν άδειο και δεν παρήγαγε κανένα αποτέλεσμα.

```
for query in formatted_queries :
            search_results = client.search(
                index="indexino",
                size=20,
                query={
                    "bool": {
                        "must": [
                            {
                                "multi_match": {
                                        "query": query.get("text", ""),
                                        "fields": ["title", "text"]
                                }
                            },
                            {
                                "multi_match": {
                                        "query": safe_query(query.get("authors", "")),
                                        "fields": ["authors"]
                                }
                            },
                            {
                                "match": {
                                    "year": {
                                        "query": query.get("year", ""),
                                    }
                                }
                            },
                            {
                                "multi_match": {
                                        "query": safe_query(query.get("cited_by", "")),
                                        "fields": ["cited_by"]
                                }
                            },
                            {
                                "multi_match": {
                                        "query": safe_query(query.get("references", "")),
                                        "fields": ["references"]
                                }
                            }
                        ],
                    }
                }
            )
```

```
trec_eval.get_results: Cannot read results file 'results.test'
trec_eval: Quit in file 'results.test'
```

Τα επόμενα πειράματά έγιναν με σκοπό να φανεί αν χρειάζονται όλα τα πεδία της των κειμένων, ή αν προκαλούν «θόρυβο» και εν τέλει επηρεάζουν αρνητικά τις μετρικές. Το πρώτο πείραμα είναι το ακόλουθο, στο οποίο γίνεται αναζήτηση μόνο στα πεδία title και text, θεωρώντας ότι αυτά είναι τα πιο σχετικά και από μόνα τους θα δίνουν το πιο «καθαρό» αποτέλεσμα.

```
"must": [
    {
        "multi_match": {
                "query": query.get("text", ""),
                "fields": ["title", "text"]
        }
    }
],
```

```
num_rel                 all     4928
num_rel_ret             all     1030
map                     all     0.0407
gm_map                  all     0.0013
```

```
P_5                     all     0.0350
P_10                    all     0.0402
P_15                    all     0.0454
P_20                    all     0.0515
```

Όπως φαίνεται, ανακτήθηκε σχεδόν το 1/5 σχετικών κειμένων με χαμηλό precision σε όλα τα επίπεδα. Επομένως, η αναζήτηση χρειάζεται και άλλα πεδία για να είναι πιο ακριβής. Ωστόσο, αφού με όλα τα πεδία σε must το αποτέλεσμα δεν βελτιώνεται, αποφασίστηκε ότι πεδίο εισαχθεί να είναι σε έναν κόμβο should. Πρώτα έγινε αναζήτηση με 4 πεδία (must: title, text και should: authors, year με τη λογική ότι τα πεδία authors και year είναι πιο σχετικά και άρα θα βοηθήσουν την αναζήτηση περισσότερο σε σχέση τα πεδία cited_by και references) και έπειτα με όλα τα πεδία (must: title, text και should: authors, year, cited_by, references). Τα αποτελέσματα έχουν παρατεθεί και στο κελί που περιγράφει το formatting των κειμένων, καθώς έτσι αποφασίστηκε και η τελική μορφή των ευρετηρίων, αλλά παρατίθενται εκ νέου, αριστερά με τέσσερα πεδία και δεξιά με έξι, για την πιο αποτελεσματική σύγκριση τους.

```
num_rel                 all     4928
num_rel_ret             all     1142
map                     all     0.0449
gm_map                  all     0.0019
```

```
P_5                     all     0.0374
P_10                    all     0.0441
P_15                    all     0.0504
P_20                    all     0.0571
```

```
num_rel                 all     4928
num_rel_ret             all     1828
map                     all     0.0829
gm_map                  all     0.0109
```

```
P_5                     all     0.0652
P_10                    all     0.0748
P_15                    all     0.0826
P_20                    all     0.0914
```

Όπως είναι εμφανές, η χρήση όλων των πεδίων βελτιώνει σημαντικά την απόδοση την αναζήτησης. Έγιναν και πειράματα σχετικά με το ποια πεδία θα πρέπει να είναι στον κόμβο must και ποια στον κόμβο should. Η βασική ιδέα ήταν αυτή που χρησιμοποιήθηκε και από πάνω, δηλαδή ότι στον κόμβο must είναι σίγουρα τα πεδία title και text, και έγινε μια προσπάθεια εξακρίβωσης αν βελτιώνει την απόδοση η μεταφορά κάποιου άλλου πεδίου (author, year, cited_by, references) από τον κόμβο should στον κόμβο must. Η συνολική απόδοση δεν βελτιώθηκε, αντίθετα χειροτέρεψε οπότε για λόγους μήκους και ευαναγνωσιμότητας δεν θα παρατεθούν όλα τα αποτελέσματα, παρά μόνο ενδεικτικά το αποτέλεσμα της μεταφοράς του πεδίου authors στον κόμβο must, το οποίο έφερε το καλύτερο αποτέλεσμα.

```
num_rel                 all     3054
num_rel_ret             all     289
map                     all     0.0552
gm_map                  all     0.0003
```

```
P_5                     all     0.0754
P_10                    all     0.0451
P_15                    all     0.0306
P_20                    all     0.0234
```

Αφού καθορίστηκε η ανάγκη χρήσης όλων των πεδίων χωρίς όμως να είναι όλα σε έναν κόμβο must, ο επόμενος συνδυασμός που δοκιμάστηκε, είναι να είναι υποχρεωτική η αναζήτηση στα πεδία title και text, ενώ τα υπόλοιπα τέσσερα να είναι προαιρετικά. Η λογική πίσω από αυτό είναι ότι ο καλύτερος δείκτης σχετικότητας ενός κειμένου είναι το match στο κείμενο (στον τίτλο ή στο σώμα κειμένου),

Επομένως, καθορίστηκε ο βασικός σκελετός όπως αναζήτησης ως εξής.

```
for query in formatted_queries :
    search_results = client.search(
        index="indexino",
        size=20,
        query={
                    "bool": {
                        "must": [
                            {
                                "multi_match": {
                                    "query": query.get("text", ""),
                                    "fields": ["title", "text"]
                                }
                            }
                        ],
                        "should" : [
                            {
                                "multi_match": {
                                    "query": safe_query(query.get("authors", "")),
                                    "fields": ["authors"]
                                }
                            },
                            {
                                "match": {
                                    "year": {
                                        "query": query.get("year", ""),
                                    }
                                }
                            },
                            {
                                "multi_match": {
                                    "query": safe_query(query.get("cited_by", "")),
                                    "fields": ["cited_by"]
                                }
                            },
                            {
                                "multi_match": {
                                    "query": safe_query(query.get("references", "")),
                                    "fields": ["references"]
                                }
                            }
                        ],
                    }
                }
    )
```

Παρατίθενται εκ νέου τα αποτελέσματα που αποφέρει αυτό το configuration, τα οποία είναι τα μέχρι στιγμής καλύτερα:

```
num_rel                 all     4928
num_rel_ret             all     1828
map                     all     0.0829
gm_map                  all     0.0109
```

```
P_5                     all     0.0652
P_10                    all     0.0748
P_15                    all     0.0826
P_20                    all     0.0914
```

Τα μέχρι στιγμής πειράματα είχαν γίνει με similarity function BM25 με πειραματικές τιμές στις παραμέτρους k1=1.4 και b=0.6, οπότε στα επόμενα πειράματα διατηρείται το configuration των queries, αλλά αλλάζουν οι παράμετροι του similarity function:

K1=1.2, b=1.75 (default τιμές)

```
num_rel                 all     4928
num_rel_ret             all     1822
map                     all     0.0812
gm_map                  all     0.0115
```

```
P_5                     all     0.0638
P_10                    all     0.0730
P_15                    all     0.0826
P_20                    all     0.0911
```

K1= 1.6, b=0.5

num_rel	all	4928
num_rel_ret	all	1824
map	all	0.0828
gm_map	all	0.0109

P_5	all	0.0658
P_10	all	0.0742
P_15	all	0.0825
P_20	all	0.0912

Όπως φαίνεται, οι αλλαγές είναι αμελητέες, οπότε τα επόμενα πειράματα θα συνεχίσουν με τις προηγούμενες τιμές των παραμέτρων (k1=1.4 και b=0.6)

Τα επόμενα πειράματα αφορούν τη τοποθέτηση βαρών στα πεδία της αναζήτησης με τη λογική ότι παρόλο που η χρήση όλων των πεδίων βελτιώνει την απόδοση του ευρετηρίου, δεν έχουν όλα την ίδια βαρύτητα.

Επομένως, τοποθετήθηκαν τα εξής βάρη: 4 στο πεδίο title, 3 στο πεδίο text και 1 στο πεδίο authors. Τα αποτελέσματα είναι τα εξής, τα οποία είναι εμφανώς χειρότερα από το πείραμα χωρίς τα βάρη, όπως φαίνεται παρακάτω.

num_rel	all	4928
num_rel_ret	all	1634
map	all	0.0712
gm_map	all	0.0073

P_5	all	0.0542
P_10	all	0.0637
P_15	all	0.0731
P_20	all	0.0817

Η μεγάλη απόκλιση μεταξύ των πεδίων (τρία πεδία με μηδενικό βάρος και τρία με αρκετά υψηλό) προφανώς δεν βοήθησε στη βελτίωση του αποτελέσματος, οπότε για το επόμενο πείραμα προστέθηκαν τα εξής βάρη και στα υπόλοιπα πεδία: 2 στο authors, 1,5 στο cited_by και 1,2 στο references, με βελτιωμένα αποτελέσματα.

num_rel	all	4928
num_rel_ret	all	1733
map	all	0.0763
gm_map	all	0.0091

P_5	all	0.0592
P_10	all	0.0687
P_15	all	0.0778
P_20	all	0.0866

Μετά από μερικά πειράματα ακόμα με τα βάρη, φάνηκε ότι κανένας συνδυασμός βαρών δεν ξεπερνάει το αρχικό αποτέλεσμα χωρίς τα βάρη, οπότε για το επόμενο πείραμα χρησιμοποιήθηκε το configuration χωρίς τα βάρη, με τη προσθήκη της εντολής minimum_should_match: 1 για τα πεδία authors, year, cited_by, references, η οποία όπως φαίνεται βελτίωσε σημαντικά τα αποτελέσματα. Φαίνεται ότι με την εντολή, τα πιο έγκυρα αποτελέσματα εμφανίζονται πιο ψηλά, αλλά χάνονται κάποια σχετικά κείμενα, ενώ χωρίς την εντολή χάνεται λίγο precision αλλά υπάρχουν περισσότερα σχετικά κείμενα.

num_rel	all	4888
num_rel_ret	all	1648
map	all	0.0962
gm_map	all	0.0082

P_5	all	0.0903
P_10	all	0.0880
P_15	all	0.0847
P_20	all	0.0831

Φαίνεται ότι με την εντολή, τα πιο έγκυρα αποτελέσματα εμφανίζονται πιο ψηλά, αλλά χάνονται κάποια σχετικά κείμενα, ενώ χωρίς την εντολή χάνεται λίγο precision αλλά υπάρχουν περισσότερα σχετικά κείμενα.

Μέχρι στιγμής σε όλα τα πειράματα το rank των αποτελεσμάτων στο αρχείο που δεχόταν ως είσοδος το Trec_val ήταν πάντα 0, τάρα δοκιμάστηκε η τοποθέτηση rank, η οποία επέφερε πολύ καλά αποτελέσματα όπως φαίνεται παρακάτω, οπότε κρατήθηκε σαν αλλαγή.

```
num_rel                 all     4908
num_rel_ret             all     1692
map                     all     0.0955
gm_map                  all     0.0091
```

```
P_5                     all     0.0900
P_10                    all     0.0889
P_15                    all     0.0859
P_20                    all     0.0849
```

Το τελευταίο πείραμα που έγινε ήταν η προσθήλη raw fields ώστε τα πεδία authors, cited_by και references να μπορούν να βρεθούν και ως keywords το οποίο θεωρητικά είναι πιο αποτελεσματικό για πολλαπλά αλφαριθμητικά στοιχεία όπως είναι η περίπτωση των τριών αυτών πεδίων, αλλά στη πράξη βλέπουμε ότι τα αποτελέσματα δεν βελτιώθηκαν, οπότε η καλύτερη λύση παραμένει η προηγούμενη. Επομένως, παρατίθεται το τελικό configuration και τα αποτελέσματα για size=20, 30, 50

```python
with open("write/results.test", "w") as file :
    for query in formatted_queries :
        search_results = client.search(
            index="indexino",
            size=20,
            query={
                "bool": {
                    "must": [
                        {
                            "multi_match": {
                                "query": query.get("text", ""),
                                "fields": ["title", "text"]
                            }
                        }
                    ],
                    "should" : [
                        {
                            "multi_match": {
                                "query": safe_query(query.get("authors", "")),
                                "fields": ["authors"]
                            }
                        },
                        {
                            "match": {
                                "year": {
                                    "query": query.get("year", ""),
                                }
                            }
                        },
                        {
                            "multi_match": {
                                "query": safe_query(query.get("cited_by", "")),
                                "fields": ["cited_by"]
                            }
                        },
                        {
                            "multi_match": {
                                "query": safe_query(query.get("references", "")),
                                "fields": ["references"]
                            }
                        }
                    ],
                    "minimum_should_match": 1

                }
            }
        )
```

```
            c = 1
            for hit in search_results["hits"]["hits"] :
                doc_id = hit["_id"]
                score = hit.get("score", None)
                run_name = "STANDARD"
                rank = c
                c+=1

                line = f"{query["uniqueID"]} Q0 {doc_id} {rank} {score} {run_name}"
                file.write(line + "\n")
```

Με αυτόν τον κώδικα συλλέγονται τα τελικά αποτελέσματα και εισάγονται σε ένα αρχείο results.test με την απαραίτητη μορφή ώστε να μπορεί να διαβαστεί από το trec_eval.

Αποτελέσματα αναζήτησης για size=20

```
num_rel                 all     4908
num_rel_ret             all     1692
map                     all     0.0955
gm_map                  all     0.0091
```

```
P_5                     all     0.0900
P_10                    all     0.0889
P_15                    all     0.0859
P_20                    all     0.0849
```

Αποτελέσματα αναζήτησης για size=30

```
num_rel                 all     4908
num_rel_ret             all     1832
map                     all     0.0874
gm_map                  all     0.0096
```

```
P_5                     all     0.0799
P_10                    all     0.0751
P_15                    all     0.0689
P_20                    all     0.0648
```

Αποτελέσματα αναζήτησης για size=50

```
num_rel                 all     4908
num_rel_ret             all     1956
map                     all     0.0810
gm_map                  all     0.0097
```

```
P_5                     all     0.0757
P_10                    all     0.0656
P_15                    all     0.0601
P_20                    all     0.0552
```

Και τέλος, υπάρχει ένα κομμάτι κώδικα το οποίο παίρνει το αρχείο με τα σωστά κείμενα και αλλάζει τη δομή του γιατί το αρχείο qrels για να διαβαστεί από το trec_eval θέλει 4 στήλες, εκ των οποίων η μια να είναι 0. Αυτή η στήλη έλειπε από το αρχείο, οπότε ο κώδικας μορφοποιεί τη δομή του ώστε να ταιριάζει με τις προδιαγραφές του trec_eval

```
with open('read/scidocs/qrels/test.tsv') as infile, open('write/qrels.test', "w") as outfile:
    next(infile)
    for line in infile:
        parts = line.strip().split("\t")
        if len(parts) == 3:
            query_id, doc_id, relevance = parts
            outfile.write(f"{query_id} 0 {doc_id} {relevance}\n")
```

✓ 0.2s