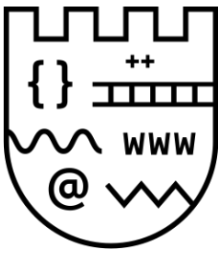
ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ



Εργασία στην Ανάκτηση Πληροφορίας

Ομιλίες Ελληνικού Κοινοβουλίου 1989-2020

Αλέξανδρος Μαυρομάτης

ΑΕΜ: 3132

Ειρήνη Ντελή

ΑΕΜ: 3393

[Φεβρουάριος] – [2022]

Εισαγωγή

Στα πλαίσια του μαθήματος της Ανάκτησης Πληροφορίας υλοποιήσαμε μία μηχανή αναζήτησης για την εύρεση σχετικών αποτελεσμάτων από τα δεδομένα ομιλιών της Βουλής των Ελλήνων.

Οι απαιτήσεις της εφαρμογής προέβλεπαν μεταξύ άλλων εύρεση ομοιοτήτων μεταξύ ομιλιών και εμφάνιση top-k ζευγών με τον υψηλότερο βαθμό ομοιότητας, τεχνικές LSI, και μία web-based εφαρμογή που θα αποτελούσε την διεπαφή του προγράμματος.

Δυστυχώς οι περισσότερες από τις παραπάνω απαιτήσεις δεν ικανοποιήθηκαν, καθώς εν τέλει υλοποιήσαμε μία απλή μηχανή αναζήτησης με έναν δεικτοδοτητή(indexer), και έναν επεξεργαστή ερωτημάτων (query processor). Η εφαρμογή επιστρέφει στον χρήστη τις σχετικότερες ομιλίες με βάση με μία αναζήτηση(query) που θα κάνει αυτός, χρησιμοποιόντας δεδομένα από αρχείο μορφής csv.

Περιορισμοί και Αρχεία

Ακολουθούν μερικές τεχνικές προϋποθέσεις που πρέπει να ληφθούν υπόψην για να λειτουργήσει το πρόγραμμα σωστά.

- Το πρόγραμμα δεν περιλαμβάνει υλοποιημένη διεπαφή, συνεπώς η χρήση του γίνεται μέσω κονσόλας.

- Το αρχείο csv που θα χρησιμοποιηθεί πρέπει να ονομάζεται parliament.csv

- Η πρώτη γραμμή στο αρχείο csv πρέπει να αφιερωθεί στο όνομα του χαρακτηριστικού που περιγράφει η κάθε στήλη πχ. member_name στην στήλη ονομάτων.

- Η στήλη των ομιλιών πρέπει να ονομάζεται speech

Οι δύο παραπάνω περιορισμοί βασίστηκαν στην μορφή του αρχείου *Greek_Parliament_Proceedings_1989_2020_DataSample.csv* , που περιέχει δείγμα των δεδομένων που θα χρησιμοποιηθούν, το οποίο πάρθηκε από το αποθετήριο github:

https://github.com/iMEdD-Lab/Greek_Parliament_Proceedings

που δώθηκε με την εκφώνηση της εργασίας.

- Τα κελιά στην στήλη των ομιλίων δεν πρέπει να είναι κενά.

- Το αρχείο parliament.csv πρέπει να βρίσκεται μέσα στον ίδιο φάκελο με τα αρχεία του κώδικα, καθώς και το αρχείο stopwords.txt

Το αρχείο stopwords περιέχει λέξεις στα Ελληνικά και στα Αγγλικά οι οποίες αποτελούν άρθρα, αντωνυμίες και άλλα μέρη του λόγου που δεν είναι ουσαστικά, είναι αρκετά συνηθισμένες και δεν πρέπει να χρησιμοποιηθούν στον αντιστραμμένο κατάλογο. Χρησιμοποιείται κατά την προεπεξεργασία των εγγραφών και των ερωτημάτων.

Δεικτοδοτητής - Indexer

1. Προεπεξεργασία δεδομένων

Για την προεπεξεργασία των δεδομένων ορίζεται η μέθοδος preprocessing. Δέχεται σαν είσοδο το αρχείο δεδομένων από το csv, και επιστρέφει τα ίδια δεδομένα αλλά επεξεργασμένα.

Αρχικά κάνει όλα κεφαλαία γράμματα μικρά. Στην συνέχεια αφαιρεί τα σημεία στίξης και τα σημεία τονισμού. Τέλος, διαβάζει το αρχείο stopwords.txt, και διαγράφει όσες λέξεις των δεδομένων ταιριάζουν με λέξεις του stopwords.

```
# Function for data preprocessing
def preprocessing(data):

    # Make all letters lowercase
    data["speech"] = data["speech"].str.lower()

    # Remove punctuation
    data["speech"] = data["speech"].str.translate(str.maketrans('', '', string.punctuation))

    #Remove diacritics
    d = {ord('\N{COMBINING ACUTE ACCENT}'):None}
    for i in range (0, len(data["speech"])):
        s = ud.normalize('NFD',data["speech"][i]).translate(d)
        data["speech"][i] = s

    # Remove stopwords
    f = open("stopwords.txt", "r")
    stopwords = f.read()
    for i, row in data.iterrows():
        data.at[i,'speech'] = ' '.join([word for word in data.at[i,'speech'].split() if word not in stop

    return data
```

2. Υπολογισμός TF

Για τον υπολογισμό της μετρικής TF ενός όρου σε ένα έγγραφο ορίζεται η μέθοδος termFrequency. Δέχεται σαν είσοδο τον όρο, την λίστα των όρων του εγγράφου και την μέγιστη συχνότητα όρου στο έγγραφο, και επιστρέφει την τιμή TF, κανονικοποιημένη, χρησιμοποιόντας έναν απλό τύπο.

```
# Function to calculate the normalized TF of a word in a document
def termFrequency(term,all_words,max_frequency):

  return all_words.count(term) / float(max_frequency)
```

3. Καθολικό λεξικό όρων, λεξικά όρων εγγράφων

Για την δημιουργία του καθολικό λεξικού όρων και των λεξικών εγγράφων με τις TF τιμές, ορίζεται η μέθοδος TF_Process. Δέχεται σαν όρισμα το αρχείο δεδομένων(data), αφού προηγουμένως υπέστη προεπεξεργασία από την μέθοδο preprocessing. Επιστρέφει το καθολικό λεξικό όρων και την λίστα που περιλαμβάνει τα λεξικά όλων των εγγράφων.

Το καθολικό λεξικό όρων που αντιστοιχίζει κάθε όρο με λίστα των εγγράφων που τον περιέχουν. Κάθε έγγραφο θα διαθέτει το δικό του λεξικό όρων το οποίο θα αντιστοιχεί κάθε όρο του εγγράφου με την TF τιμή του. Ο δείκτης(index) των στοιχείων της λίστας αντιστοιχεί στον δείκτη(id) του εγγράφου από το αρχείο δεδομένων.

Αρχικά η επιλεγμένη εγγραφή μετατρέπεται σε μια λίστα όρων.

```
# Function to calculate TF for every term in every document
def TF_Process(data):
    globalDict = {}     # Global dictionary where:
                        # key = term
                        # value = documents where term can be found

    documentsTF = []    # List that will store dictionaries where:
                        # List index = document id
                            # key = term
                            # value = TF in the particular document

    for i, row in data.iterrows():  #For each document
        speech = data.at[i,'speech']

        # Take number of all non-unique words in a document
        all_words = speech.split()

        TF={}   # Dictionary where:
                # key = term,
                # value = TF in the document
```

Στην συνέχεια, κάθε όρος κάθε εγγραφής εντάσσεται στο καθολικό λεξικό, το οποίο ενημερώνει την λίστα με τις εγγραφές που τον περιέχουν. Επίσης για κάθε όρο υπολογίζεται η συχνότητα εμφάνισής του μέσα σε ένα έγγραφο, για να υπολογιστεί στο τέλος του βρόχου του εγγράφου η μέγιστη συχνότητα όρου του. Ύστερα υπολογίζεται η τιμή tf του κάθε όρου με την χρήση της μεθόδου termFrequency, η οποία μπαίνει στο λεξικό του εγγράφου TF. Στο τέλος, το λεξικό προστίθεται στην λίστα λεξικών εγγράφων.

```
        max_frequency = 0
        for term in all_words:

            # update global dictionary with all the documents where a term belongs
            if not term in globalDict:
                globalDict[term] = [i]
            elif not term in TF:
                doc_list = globalDict.get(term)
                doc_list.append(i)
                globalDict[term] = doc_list

            #Calculate maximum frequency of a term in the document
            if(max_frequency <= all_words.count(term)):
                max_frequency = all_words.count(term)

        # calculate the TF for each term
        for term in all_words:
            TF[term] = termFrequency(term,all_words,max_frequency)

        # Update the document TF dictionaries list
        documentsTF.append(TF)

    return globalDict, documentsTF
```

4. Υπολογισμός IDF

Για τον υπολογισμό της μετρικής IDF των όρων, ορίζεται η μέθοδος IDF_Process. Δέχεται σαν ορίσματα το καθολικό λεξικό όρων και τον συνολικό αριθμό των εγγράφων, και επιστρέφει το λεξικό IDF όρων. Για κάθε όρο του καθολικού λεξικού υπολογίζεται με έναν τύπο η τιμή IDF.

```
# Function to calculate IDF for every term
def IDF_Process(globalDict,total_docs):
    dictionaryIDF = globalDict.copy()
    for i in dictionaryIDF.keys():
        dictionaryIDF[i] = 1 + math.log(float(total_docs/len(dictionaryIDF[i])))
    return dictionaryIDF
```

Επεξεργαστής Ερωτημάτων

1. Προεπεξεργασία Ερωτήματος

Για την προεπεξεργασία του ερωτήματος ορίζεται η μέθοδος preprocessing. Δέχεται σαν είσοδο το ερώτημα και το επιστρέφει επεξεργασμένο.

Η διαδικασία είναι η ίδια όπως και στην προεπεξεργασία των δεδομένων, τα γράμματα γίνονται μικρά, αφαιρούνται τα σημεία στίξης και οι τόνοι, καθώς και τα stopwords.

```
# Function for query preprocessing
def query_Preprocessing(query):

    # Make all letters lowercase
    query = query.lower()

    # Remove punctuation
    query = query.translate(str.maketrans('', '', string.punctuation))

    # Remove diacritics
    query = query.split()
    d = {ord('\N{COMBINING ACUTE ACCENT}'):None}
    for i in range (0, len(query)):
        s = ud.normalize('NFD',query[i]).translate(d)
        query[i] = s

    # Remove stopwords
    f = open("stopwords.txt", "r")
    stopwords = f.read()
    query = ' '.join([word for word in query if word not in stopwords])

    return query.split()
```

2. Καθολικό λεξικό και λεξικό TF ερωτήματος

Η μέθοδος query_TF_Process δημιουργεί ένα καθολικό λεξικό ενημερωμένο με τους όρους του ερωτήματος, και ένα λεξικό TF όρων όπως θα είχε ένα έγγραφο. Η διαδικασία είναι ίδια με αυτήν της TF_Process, με την διαφορά όταν προστίθεται ένας όρος στο καθολικό λεξικό στην λίστα εγγράφων αντί της id που έμπαινε στα έγγραφα μπαίνει το string “query”.

```
# Function to calculate TF for every term in the query
def query_TF_Process(query, data, globalDict):

    queryGlobalDict = globalDict.copy()

    TF={}   # Dictionary where:
            # key = term,
            # value = TF in the query

    max_frequency = 0
    for term in query:

        # Update the query global dictionary
        if not term in queryGlobalDict:
            queryGlobalDict[term] = ["query"]
        elif not term in TF:
            doc_list = queryGlobalDict.get(term)
            doc_list.append("query")
            queryGlobalDict[term] = doc_list

        #Calculate maximum frequency of a term in the query
            if(max_frequency <= query.count(term)):
                max_frequency = query.count(term)

    # calculate the TF for each term
    for term in query:
        TF[term] = termFrequency(term,query,max_frequency)

    return queryGlobalDict, TF
```

3. Υπολογισμός IDF

Η μέθοδος αυτή υπολογίζει το IDF των όρων του ερωτήματος και δημιουργεί ένα ενημερωμένο λεξικό IDF όρων. Ο υπολογισμός του IDF γίνεται όπως και στον indexer.

```
# Function to calculate IDF for every term
def query_IDF_Process(query, queryGlobalDict, dictionaryIDF, total_docs):
    queryDictionaryIDF = dictionaryIDF.copy()
    for i in query:
        queryDictionaryIDF[i] = 1 + math.log(float(total_docs/len(queryGlobalDict[i])))

    return queryDictionaryIDF
```

4. Υπολογισμός TF-IDF διανυσμάτων εγγράφων

Η μέθοδος create_vector υπολογίζει τα βάρη των όρων ενός εγγράφου και τα αποθηκεύει σε ένα λεξικό TF-IDF όρων. Δέχεται ως όρισμα ένα έγγραφο, το ανανεωμένο λεξικό IDF και το ανανεωμένο καθολικό λεξικό όρων, και επιστρέφει το λεξικό TF-IDF. Ο υπολογισμός του βάρους γίνεται με γινόμενο των TF και IDF τιμών.

```python
# Function to create the tf-idf vector of a document
def create_vector(single_doc, queryDictionaryIDF, queryGlobalDict):
    doc_tfidf = {}

    # Every weight is initially 0
    for term in queryGlobalDict.keys():
        doc_tfidf[term] = 0

    for term in queryGlobalDict.keys():
        if term in single_doc.keys():
            doc_tfidf[term] = single_doc[term]* queryDictionaryIDF[term]

    return doc_tfidf
```

5. Υπολογισμός ομοιότητας συνημιτόνου

Η μέθοδος cosine_formula δέχεται σαν όρισμα διανύσματα βαρών, ένα του ερωτήματος και ένα ενός εγγράφου, και υπολογίζει την ομοιότητα συνημιτόνου μεταξύ τους. Η ομοιότητα προκύπτει από το γινόμενο των διανυσμάτων προς το γινόμενο των μέτρων τους.

```python
# Function to perform cosine similarity
def cosine_formula(query,doc):

    dot_product = np.dot(query,doc)
    query_norm = np.linalg.norm(query)
    doc_norm = np.linalg.norm(doc)

    return (dot_product/query_norm*doc_norm)
```

6. Υπολογισμός πίνακα ομοιότητας

Η μέθοδος cosine_similarity δημιουργεί τον πίνακα ομοιότητας των εγγράφων, χρησιμοποιόντας την μέθοδο cosine_formula. Δέχεται σαν όρισμα το διάνυσμα βαρών του ερωτήματος και την λίστα των TF-IDF λεξικών των εγγράφων.

```python
# This is the function that builds the similarity matrix between the query and the documents
def cosine_similarity(query_vector_list,tfidf_list):
    similarity_matrix = [cosine_formula(query_vector_list,list(doc.values())) for doc in tfidf_list]

    return similarity_matrix
```

7. Βασική μέθοδος εύρεσης σχετικών αποτελεσμάτων

Η μέθοδος query_search χρησιμοποιεί τις προηγούμενες μεθόδους για να υλοποιήσει την αναζήτηση σχετικών αποτελεσμάτων με βάση το ερώτημα του χρήστη.

Αρχικά, φορτώνει τα συνολικά δεδομένα από το αρχείο “parliament.csv”. Χρησιμοποιεί την μέθοδο preprocessing για να φέρει τα δεδομένα σε κατάλληλη μορφή. Στην συνέχεια, με τα δεδομένα αυτά δημιουργεί το καθολικό λεξικό όρων και την λίστα λεξικών TF των εγγράφων με την μέθοδο TF_Process. Μετά δημιουργεί και το λεξικό IDF με την μέθοδο IDF_Process. Η διαδικασία δημιουργίας των λεξικών επαναλαμβάνεται παρακάτω για το ερώτημα με τις αντίστοιχες μεθόδους.

```
# This is the main query search function
def query_search(query):

    data = pd.read_csv("parliament.csv")

    # Preprocess data
    data = preprocessing(data)

    # globalDict: dictionary with all the terms (keys) and the documents where they can be found (values
    # documentsTF: list of dictionaries with terms and their frequency in a document
    globalDict, documentsTF = TF_Process(data)

    # dictionaryIDF: dictionary with all the terms (keys) and their respective IDF (values)
    dictionaryIDF = IDF_Process(globalDict,data.shape[0])

    # Preprocess query
    query = query_Preprocessing(query)
    # Same as the dicionaries above, with the addition of the query terms in them
    queryGlobalDict, queryTF = query_TF_Process(query,data,globalDict)
    queryDictionaryIDF = query_IDF_Process(query, queryGlobalDict, dictionaryIDF, data.shape[0])
```

Ύστερα ακολουθεί η δημιουργία των διανυσμάτων βαρών. Πριν τον πολλαπλασιασμό των διανσυμάτων TF και IDF, η λίστα των λεξικών TF των εγγράφων ενημερώνεται με μηδενικά για τους όρους που αυτά δεν περιέχουν, για να είναι τα διανύσματα TF και IDF του ίδιου βαθμού. Με την create_vector υπολογίζεται το γινόμενο και ενημερώνεται η λίστα διανυσμάτων βαρών tfidf_list.

Η αντίστοιχη διαδικασία εφαρμόζεται και για το ερώτημα, μόνο που αρχικά δημιουργείται ένα λεξικό βαρών και όχι κατευθείαν διάνυσμα, και το γινόμενο υπολογίζεται επί τόπου χωρίς την χρήση μεθόδου.

```
#Fill the tf of terms not included in a document with zeroes
for i in queryGlobalDict.keys():
    for j in range(len(documentsTF)):
        if not i in documentsTF[j]:
            documentsTF[j][i]=0

# Create tf-idf vectors for all the documents and put them on a list
tfidf_list = []
for single_doc in documentsTF:
    tfidf_list.append(create_vector(single_doc, queryDictionaryIDF, queryGlobalDict))


# Create a dictionary for all terms tf-idf
queryTFIDF={}
for term in queryGlobalDict.keys():
    queryTFIDF[term] = 0

for i in query:
    queryTFIDF[i] = queryDictionaryIDF[i]*queryTF[i]
```

Τέλος, το λεξικό βαρών του ερωτήματος μετρέπεται σε διάνυσμα, και χρησιμοποιείται η μέθοδος cosine_similarity για τον υπολογισμό του πίνακα ομοιότητας. Στα αρχικά δεδομένα προστίθεται σαν στήλη Score το διάνυσμα ομοιότητας, και ταξινομούνται με βάση αυτήν.

```
    # Turn the query TFIDF dictionary to a list
    query_vector_list = list(queryTFIDF.values())
    # Compute the similarity matrix between our query and our documents
    similarity_matrix = cosine_similarity(query_vector_list,tfidf_list)
    query_data = data.copy()
    #query_data['TFIDF'] = tfidf_list  # you can put the tfidf data in your final results if you want
    query_data['Score'] = np.array(similarity_matrix)
    query_data.sort_values(by=['Score'], inplace=True, ascending=False)
    return query_data, queryTFIDF
```

Τελικό αρχείο εφαρμογής (Main)

Η Main αποτελείται ουσιαστικά από έναν βρόχο while με μόνιμη συνθήκη true που εκτελεί συνεχώς ερωτήματα του χρήστη και σταματάει όταν αυτός κλείσει το τερματικό.

1. Έλεγχος για σωστή είσοδο ερωτήματος

Γίνεται έλεγχος για να μην δώσει ο χρήστης κενή είσοδο, είτε πατώντας κατευθείαν enter είτε βάζοντας μόνο spaces και tabs.

```
    # Loop for correct query input
    while(true):
        query = input("\nInsert your query: ")
        if query.strip():
            break
        print("\nPlease do not insert an empty query")
```

2. Έλεγχος για σωστή είσοδο top-k αποτελεσμάτων και εκτέλεση

Μετά το ερώτημα ο χρήστης ζητήται να εισάγει τον αριθμό των πιο σχετικών αποτελεσμάτων που θέλει να του επιστρέψει το πρόγραμμα, δηλαδή πόσες σχετικές εγγραφές θα του επιστρέψει. Μετά τον έλεγχο, εκτελεί την μέθοδο search query.

```
    # Loop for correct top-k results input
    while(true):
        num = input("\nInsert the number of top relevant results you want: ")
        if not(num.isnumeric()):
            print("\nPlease give a valid integer number.")
            continue
        num = int(num)
        if(num >= 1):
            break
        print("\nPlease give a valid integer number.")

    query_data, full_query_vector = query_search(query)

    print("\n")
    print(query_data[0:num])
```

Παραδείγματα ερωτημάτων

Τα παραδείγματα αυτά βασίζονται στο αρχείο *Greek_Parliament_Proceedings_1989_2020_DataSample.csv* από την σελίδα https://github.com/iMEdD-Lab/Greek_Parliament_Proceedings

1. Δοκιμές λανθασμένων εισόδων

```
Insert your query:

Please do not insert an empty query

Insert your query: ερντογαν αγια σοφια τζαμι μητσοτακης

Insert the number of top relevant results you want: -1

Please give a valid integer number.

Insert the number of top relevant results you want: 0

Please give a valid integer number.

Insert the number of top relevant results you want: εεεε

Please give a valid integer number.

Insert the number of top relevant results you want: 10
```

2. Δοκιμές ερωτημάτων που αναμένεται αποτέλεσμα

```
Insert your query: ερντογαν αγια σοφια τζαμι μητσοτακης

Insert the number of top relevant results you want: 15


                                member_name sitting_date  ... Unnamed: 11      Score
71         βοριδης χρηστου μαυρουδης (μακης)   24/07/2020  ...         NaN  21.830573
39                  πασχαλιδης αδαμ ιωαννης   24/07/2020  ...         NaN  20.363559
61              μανωλακου εμμανουηλ διαμαντω   24/07/2020  ...         NaN  18.564133
37                μπουκωρος γεωργιου χρηστος   24/07/2020  ...         NaN  17.362976
19               αραχωβιτης γεωργιου σταυρος   24/07/2020  ...         NaN  15.824714
47                ανδριανος σωτηριου ιωαννης   24/07/2020  ...         NaN  11.628322
49              σκονδρα κωνσταντινου ασημινα   24/07/2020  ...         NaN  10.909433
27  χρυσομαλλης παναγιωτη μιλτιαδης (μιλτος)   24/07/2020  ...         NaN  10.560294
6         μαραβεγιας αριστοτελη κωνσταντινος   24/07/2020  ...         NaN   8.689603
41                αραμπατζη αθανασιου φωτεινη   24/07/2020  ...         NaN   8.243624
23                χητας αχιλλεως κωνσταντινος   24/07/2020  ...         NaN   7.894636
10   μπαραλιακος ελευθεριου ξενοφων (φωντας)   24/07/2020  ...         NaN   4.193311
16      δημοσχακης σπυρου αναστασιος (τασος)   24/07/2020  ...         NaN   3.655347
0               σκρεκας θεοδωρου κωνσταντινος   24/07/2020  ...         NaN   2.537273
31                γρηγοριαδης γεωργιου κλεων   24/07/2020  ...         NaN   0.986587

[15 rows x 13 columns]
```

```
Insert your query: αλιεια μηκος σκαφους φορος

Insert the number of top relevant results you want: 10


                              member_name sitting_date  ... Unnamed: 11      Score
21             αραχωβιτης γεωργιου σταυρος   24/07/2020  ...         NaN  21.827631
12         κεγκερογλου αλεξανδρου βασιλειος   24/07/2020  ...         NaN  10.297177
14         κεγκερογλου αλεξανδρου βασιλειος   24/07/2020  ...         NaN   6.925721
61             μανωλακου εμμανουηλ διαμαντω   24/07/2020  ...         NaN   4.896406
55            βιλιαρδος διονυσιου βασιλειος   24/07/2020  ...         NaN   3.012037
0              σκρεκας θεοδωρου κωνσταντινος   24/07/2020  ...         NaN   0.000000
66  κωνσταντινοπουλος κωνσταντινου οδυσσεας   24/07/2020  ...         NaN   0.000000
74              τελιγιοριδου ιωαννη ολυμπια   24/07/2020  ...         NaN   0.000000
73        βοριδης χρηστου μαυρουδης (μακης)   24/07/2020  ...         NaN   0.000000
72             αραχωβιτης γεωργιου σταυρος   24/07/2020  ...         NaN   0.000000

[10 rows x 13 columns]
```

```
Insert your query: χωραφια αγροκτημα νομοσχεδιο φορολογια

Insert the number of top relevant results you want: 10


                              member_name sitting_date  ... Unnamed: 11      Score
49             σκονδρα κωνσταντινου ασημινα   24/07/2020  ...         NaN  11.495687
71        βοριδης χρηστου μαυρουδης (μακης)   24/07/2020  ...         NaN  10.208803
27  χρυσομαλλης παναγιωτη μιλτιαδης (μιλτος)   24/07/2020  ...         NaN   8.042412
43              αραμπατζη αθανασιου φωτεινη   24/07/2020  ...         NaN   7.942106
37               μπουκωρος γεωργιου χρηστος   24/07/2020  ...         NaN   6.920297
12         κεγκερογλου αλεξανδρου βασιλειος   24/07/2020  ...         NaN   4.110134
6         μαραβεγιας αριστοτελη κωνσταντινος   24/07/2020  ...         NaN   3.665741
35    χειμαρας αθανασιου θεμιστοκλης (θεμης)   24/07/2020  ...         NaN   3.268087
41              αραμπατζη αθανασιου φωτεινη   24/07/2020  ...         NaN   2.660727
47               ανδριανος σωτηριου ιωαννης   24/07/2020  ...         NaN   2.502169

[10 rows x 13 columns]
```

3. Δοκιμές ερωτημάτων που δεν αναμένεται αποτέλεσμα

```
Insert your query: τσιπουρα γκαζακι instagram league of legends

Insert the number of top relevant results you want: 10


                              member_name sitting_date  ... Unnamed: 11 Score
0              σκρεκας θεοδωρου κωνσταντινος   24/07/2020  ...         NaN   0.0
63                 πανας νικολαου αποστολος   24/07/2020  ...         NaN   0.0
73        βοριδης χρηστου μαυρουδης (μακης)   24/07/2020  ...         NaN   0.0
72             αραχωβιτης γεωργιου σταυρος   24/07/2020  ...         NaN   0.0
71        βοριδης χρηστου μαυρουδης (μακης)   24/07/2020  ...         NaN   0.0
70  κωνσταντινοπουλος κωνσταντινου οδυσσεας   24/07/2020  ...         NaN   0.0
69             μανωλακου εμμανουηλ διαμαντω   24/07/2020  ...         NaN   0.0
68  κωνσταντινοπουλος κωνσταντινου οδυσσεας   24/07/2020  ...         NaN   0.0
67               σενετακης γεωργιου μαξιμος   24/07/2020  ...         NaN   0.0
66  κωνσταντινοπουλος κωνσταντινου οδυσσεας   24/07/2020  ...         NaN   0.0

[10 rows x 13 columns]
```