Γλωσσική Τεχνολογία

Εργασία 2020-2021

έΥΑ ντουρου (παλαιοσ αμ 235854, νεοσ αμ 1041567)

2021

Περιεχόμενα

[Μέρος Α 2](#_Toc81926612)

[Προσκομιστής Ιστοσελίδων 2](#_Toc81926613)

[Προεπεξεργασία Δεδομένων 2](#_Toc81926614)

[**Politico** 3](#_Toc81926615)

[**Euronews** 3](#_Toc81926616)

[**BBC** 3](#_Toc81926617)

[**Aljazeera** 4](#_Toc81926618)

[**Eureporter** 4](#_Toc81926619)

[Μορφοσυντακτική Ανάλυση 4](#_Toc81926620)

[Αναπαράσταση ιστοσελίδων στο Μοντέλο Διανυσματικού Χώρου. 5](#_Toc81926621)

[Δημιουργία του ευρετηρίου 7](#_Toc81926622)

[Αναφορές 7](#_Toc81926623)

# Μέρος Α

## Προσκομιστής Ιστοσελίδων

Ο προσκομιστής ιστοσελίδων υλοποιήθηκε με τη χρήση του Scrapy (1), το οποίο λαμβάνει ως πηγές τα RSS feeds που φαίνονται στον πίνακα 1 και εξάγει το περιεχόμενό τους σε μορφοποίηση html, το οποίο και αποθηκεύει στο directory του project. Η υλοποίησή του φαίνεται στο αρχείο parker.py. (./scraper\_tintin/Scraper/NewsScrape/NewsScrape/spiders/parker.py)

Πίνακας 1 Πηγές του crawler

|  |  |
| --- | --- |
| **Όνομα πηγής** | **RSS feed Link** |
| Eureporter | https://www.eureporter.co/feed/ |
| Aljazeera | https://www.aljazeera.com/xml/rss/all.xml |
| Euronews | http://feeds.feedburner.com/euronews/en/home/ |
| BBC | http://feeds.bbci.co.uk/news/world/rss.xml |
| Washington Post | http://feeds.washingtonpost.com/rss/world |
| Politico | https://www.politico.eu/feed/ |
| NY Times | https://www.nytimes.com/svc/collections/v1/publish/https://www.nytimes.com/section/world/rss.xml |

Από τις παραπάνω ιστοσελίδες, οι NY times δεν επέτρεπαν πρόσβαση σε scraper μέσω του robots.txt.

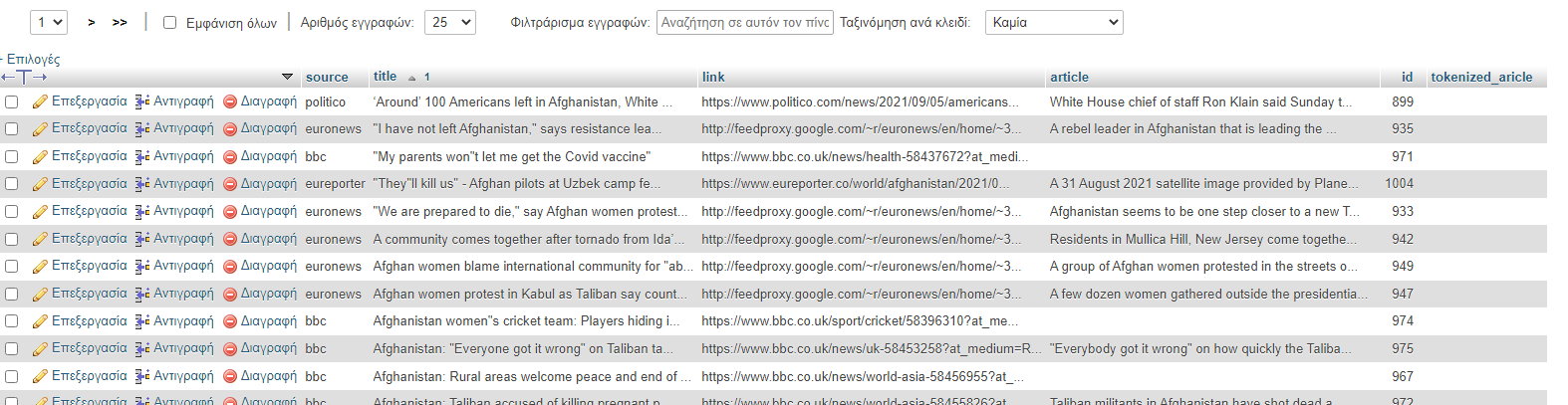
Οι σελίδες αυτές μετά το scrape, αποθηκεύονται σε αρχεία html, για την προεπεξεργασία.

## Προεπεξεργασία Δεδομένων

Για την προεπεξεργασία δεδομένων παίρνουμε ξεχωριστά την κάθε αποθηκευμένη σελίδα και με χρήση του εργαλείου Beautiful Soup, εξάγουμε από αυτή τους τίτλους. Συγκεκριμένα, ο τρόπος εξαγωγής για κάθε σελίδα φαίνεται στην αντίστοιχη παράγραφο παρακάτω.

Στο τέλος αυτής της διαδικασίας έχουμε για κάθε σελίδα τρεις λίστες titles, links, articles οι οποίες περιλαμβάνουν αντίστοιχα τους τίτλους, τα urls προς τα άρθρα, τα οποία χρησιμοποιούνται ως μοναδικά κλειδιά στις επόμενες φάσεις, για να ξεχωρίζουμε τα άρθρα και τέλος τα άρθρα.

Στη συνέχεια, εισάγουμε με τη σειρά τις παραπάνω λίστες σε localhost database, με χρήση MySQL και phpmyadmin. Η βάση δεδομένων έχει τη μορφή που φαίνεται στην εικόνα 1 και περιλαμβάνει και στήλη για την αποθήκευση του tokenized άρθρου, το οποίο γίνεται σε επόμενα ερωτήματα.



Εικόνα 1

### **Politico**

Κάθε άρθρο περιλαμβάνεται υπό ένα <item> tag.

Οι τίτλοι και τα links των άρθρων βρίσκονται στα αντίστοιχα tags.

To RSS του politico περιλαμβάνει και ολόκληρα τα άρθρα, στο tag <content:encoded>, το οποίο το Beautiful Soup μετατρέπει απλά σε <encoded>. Το εξάγουμε μέσω της findAll() και στη συνέχεια το μετατρέπουμε σε καλύτερη μορφή, αφαιρώντας τα tags :

article = re.sub('<.\*?>', '', article)

και τα newlines:

article = ''.join(article.strip().split('\\n'))

Αξίζει να σημειωθεί πως ανάλογη επεξεργασία γίνεται και στα links, καθώς περιλαμβάνουν ειδικούς χαρακτήρες \r, \n οι οποίοι δυσκολεύουν την χρήση τους.

### **Euronews**

Αντίστοιχα με το Politico, οι τίτλοι και τα links εξάγονται από τα αντίστοιχα tags.

Το Euronews δεν περιλαμβάνει ολόκληρα άρθρα στο RSS feed του, επομένως χρησιμοποιούμε το κάθε link για να κάνουμε εκ νέου request και στη συνέχεια να εξάγουμε το άρθρο από τη σελίδα του μέσω του Beautiful Soup.

r = requests.get(link)  
good\_soup = BeautifulSoup(r.content, 'html.parser')

Συγκεκριμένα εξάγουμε όλα τα div του class 'c-article-content' και στη συνέχεια από αυτά όλα τα p tags, δηλαδή τις παραγράφους του άρθρου. Έπειτα της συνδέουμε όλες για να αποθηκεύσουμε ολόκληρο το άρθρο.

divs = good\_soup.find\_all('div', class\_ = 'c-article-content')

for div in divs:  
 for p in div.find\_all('p'):  
 article.append(p.text)

### **BBC**

To RSS feed γίνεται parse ακριβώς όπως στο Euronews.

Στη συνέχεια, πρέπει να πάρουμε το άρθρο από το link του, με εκ νέου scrape του. Συγκεκριμένα γίνεται ένα request στο link για το περιεχόμενο της σελίδας κι έπειτα λαμβάνουμε με χρήση του Beautiful Soup το άρθρο, εξάγοντας αρχικά το article tag με class "ssrcss-xalfp3-ArticleWrapper e1nh2i2l6", το οποίο περιέχει το άρθρο μαζί με πιθανές εικόνες κι έπειτα στο div με class 'ssrcss-uf6wea-RichTextComponentWrapper e1xue1i87' υπό το article tag εξάγουμε όλα τα paragraphs (p tags), τα οποία και περιέχουν το κείμενο του άρθρου.

### **Aljazeera**

Το parsing γίνεται όπως στο Euronews, με την εξαίρεση ότι το όνομα του class που αναζητούμε στο div tag είναι 'wysiwyg wysiwyg--all-content css-1vsenwb'.

### **Eureporter**

Το RSS feed αυτής της σελίδας περιλαμβάνει και άρθρα, επομένως το parsing γίνεται όπως και του Politico.

Για την αποθήκευση των άρθρων επιλέχθηκε SQL database γιατί έτσι η ανάκτησή τους γίνεται με ένα query και το link, δεν χρειάζεται να διατηρείται ανοιχτό αρχείο json ή xml και να γίνεται συνέχεια scrape, καθώς και είναι εύκολο να προστεθεί απλά στην εγγραφή του αντίστοιχου link το tokenized άρθρο στην συνέχεια.

### Μορφοσυντακτική Ανάλυση

Στη συνέχεια, με χρήση του εργαλείου NLTK, γίνεται tokenization και pos tagging για κάθε κείμενο της βάσης δεδομένων μας. Αρχικά, συλλέγουμε τα links και τα αντίστοιχα άρθρα από τη βάση, με ένα SELECT ερώτημα. Έπειτα, για κάθε link και άρθρο:

Αφαιρούνται οι χαρακτήρες \r,\t,\n οι οποίοι προκύπτουν από το scraping:

article = cleaner.clean\_link(article)

Το άρθρο γίνεται tokenize:

tokens = nltk.word\_tokenize(article)

και στη μεταβλητή tokens έχουμε λίστα από τις λέξεις tokens που εμφανίζονται σε αυτό. Το αποτέλεσμα αυτό επεξεργαζόμαστε ώστε να αφαιρέσουμε τα σημεία στίξης ως εξής:

punctuations = "?:!.,;"

for tkn in tokens:  
 if tkn in punctuations:  
 tokens.remove(tkn)

Τέλος αφαιρούνται τα stopwords:

no\_stopwords = [word.lower() for word in tokens if not word in stopwords.words()]

Σημειώνεται πως το tokenized άρθρο, πριν από την επεξεργασία για σημεία στίξης και stopwords αποθηκεύεται στη βάση δεδομένων:

def insert\_tokenized(tokens, key):  
 db = MySQLdb.connect('localhost', 'root', '', 'articleDB', charset='utf8')  
 crs = db.cursor()  
 crs.execute(f"""UPDATE articles SET tokenized\_article = {tokens.replace("'", '"').encode("ascii", "replace").decode()} WHERE id = {key}""")

Τέλος, γίνεται μορφοσυντακτικός σχολιασμός μέσω του pos\_tag() του NLTK και τα tokens, με τα pos tags σε μορφή διανύσματος αποθηκεύονται σε αρχείο json.

tagged = nltk.pos\_tag(no\_stopwords)  
entities = nltk.chunk.ne\_chunk(tagged)  
tokenized\_articles.append(tagged)

Το διάνυσμα για κάθε άρθρο είναι της μορφής:

[[“token”, “POS\_TAG”], [“token”, “POS\_TAG”],…,[“token”, “POS\_TAG”]]

και συνολίκα το αρχείο json στο οποίο αποθηκεύονται είναι της μορφής:

{ “link” : [[“token”, “POS\_TAG”], [“token”, “POS\_TAG”],…,[“token”, “POS\_TAG”]],

…..,

“link”: [[“token”, “POS\_TAG”], [“token”, “POS\_TAG”],…,[“token”, “POS\_TAG”]] }

Σε επίπεδο κώδικα, έχουμε μια λίστα links, η οποία περιλαμβάνει τα links και αντίστοιχα μια λίστα λιστών tokenized\_articles, η οποία, με αντιστοιχία στις θέσεις των links έχει την αντίστοιχη λίστα tokens, με το POS σχολιασμό τους, ως tuples.

## Αναπαράσταση ιστοσελίδων στο Μοντέλο Διανυσματικού Χώρου.

Όπως αναφέρθηκε στην παραπάνω παράγραφο, η αναπαράσταση των άρθρων ως διάνυσμα και η αποθήκευσή τους σε αρχείο για μετέπειτα χρήση γίνεται μαζί με τη μορφοσυντακτική ανάλυση τους, στο αρχείο tokenizer.py.

Στη συνέχεια, στο αρχείο indexer.py γίνεται η αφαίρεση των closed class categories, το lemmatization και η κατασκευή του ευρετηρίου.

Η συνάρτηση που αφαιρεί τα closed class categories είναι η εξής:

def remove\_closedclasscategories(tagged\_dic):  
 closedclasscategories = ['.', ',', ':', 'CD', 'CC', 'DT', 'EX', 'IN', 'LS', 'MD', 'PDT', 'POS', 'PRP', 'PRP$', 'RP', 'TO', 'UH', 'WDT', 'WP', 'WP$', 'WRB', ".", ",", ":", "D" "C" "T" "X" "N" "S" "D" "DT" "OS" "RP" "RP$" "P" "O" "H" "DT" "P" "P$" "RB"]  
 for key in tagged\_dic:  
 article = tagged\_dic[key]  
 for pair in article:  
 if pair[1] in closedclasscategories:  
 pair.clear()  
 tagged\_dic[key] = article  
 return tagged\_dic

Δέχεται ένα dictionary με pos tagged tokens και διατηρεί σε λίστα τα tags που αποτελούν closed class categories. Στη συνέχεια κάνει iterate το dictionary, ελέγχοντας εάν το tag του κάθε token βρίσκεται στη λίστα και αν βρίσκεται το αφαιρεί από το άρθρο.

Στη συνέχεια αποθηκεύεται και το άρθρο χωρίς τα closed class categories σε αρχείο json για μετέπειτα χρήση:

with open('Results/pos\_tags\_noclosed.json', 'w')as json\_out:  
 json.dump(tagged\_dictionary, json\_out)  
lemmas = get\_lemmas(tagged\_dictionary)

Το lemmatization γίνεται με χρήση του WordNetLemmatizer, διότι παρέχει την ικανότητα να παράγουμε λήμμα από τα tokens που έχουμε στο κάθε άρθρο δεδομένου του context του, δηλαδή της χρήσης της λέξης στο άρθρο, το οποίο φαίνεται από το POS tag του. Για να λειτουργήσει, θα πρέπει να μετατρέψουμε τα tags που έχουμε ήδη σε tag που δέχεται ο WordNetLemmatizer (adj, verb, noun, adv):

def get\_wordnet\_pos(treebank\_tag):  
 if treebank\_tag.startswith('J'):  
 return wordnet.ADJ  
 elif treebank\_tag.startswith('V'):  
 return wordnet.VERB  
 elif treebank\_tag.startswith('N'):  
 return wordnet.NOUN  
 elif treebank\_tag.startswith('R'):  
 return wordnet.ADV  
 else:  
 return wordnet.NOUN

Στη συνέχεια δημιουργούμε τα λήμματα εφαρμόζοντας lemmatizer σε κάθε token του κάθε άρθρου και αποθηκεύουμε τη λίστα λημμάτων σε txt αρχείο για μετέπειτα χρήση:

def get\_lemmas(tagged\_dictionary):  
 lemmatizer = WordNetLemmatizer()  
 lemmas = []  
 for key in tagged\_dictionary:  
 for pair in tagged\_dictionary[key]:  
 if len(pair) == 0:  
 continue  
 else:  
 if lemmatizer.lemmatize(pair[0]) not in lemmas:  
 lemmas.append(lemmatizer.lemmatize(pair[0], pos=get\_wordnet\_pos(pair[1])))

Η συνάρτηση δέχεται dictionary το οποίο περιέχει τα άρθρα και τα αντίστοιχα token vectors με τα POS tags τους. Ελέγχοντας ανα link (key), δημιουργεί λήμματα και τα προσθέτει στη λίστα λημμάτων, η οποία στη συνέχεια θα επιστραφεί συνάρτηση που δημιουργεί το ευρετήριο.

Επιπλέον, αφαιρούνται τυχόν λήμματα που προκύπτουν ως μοναδικοί χαρακτήρες:

# remove single character lemmas  
for lemma in lemmas:  
 if len(lemma) == 1:  
 lemmas.remove(lemma)

## Δημιουργία του ευρετηρίου

Έπειτα από τις παραπάνω διεργασίες, έχουμε στα χέρια μας μια λίστα λημμάτων, και ένα dictionary που περιλαμβάνει εγγραφές της μορφής {link : <POS tagged tokenized article vector>}, επομένως μπορούμε να μετρήσουμε τις συχνότητες των όρων στα κείμενα και να δημιουργήσουμε το ανεστραμμένο ευρετήριο. Για τον υπολογισμό της μετρικής TF-IDF δημιουργούνται οι συναρτήσεις term\_frequency(lemma, tagged\_dictionary[key]) και inverse\_document\_frequency(lemma, tagged\_dictionary), οι οποίες χρησιμοποιούν αντίστοιχα τις σχέσεις (1) και (2) για τον υπολογισμό του βάρους του κάθε λήμματος στο κείμενο. Ο έλεγχος γίνεται για κάθε λήμμα και παράλληλα με χρήση της βιβλιοθήκης cElementTree του εργαλείου xml για python, γίνεται εγγραφή του ευρετηρίου στο output file”inverted\_index.xml”. Η διαδικασία αυτή φαίνεται στις γραμμές 25-52 του αρχείου indexer.py

Στην εικόνα 2 φαίνεται κομμάτι του ευρετηρίου inverted\_index.py που δημιουργήθηκε:



Εικόνα 2

## Αξιολόγηση Ευρετηρίου

### Ερωτήματα της μιας λέξης

|  |  |  |  |
| --- | --- | --- | --- |
| **Α/Α** | **ΕΡΩΤΗΜΑ** | **ΑΠΟΚΡΙΣΗ** | **ΑΡΙΘΜΟΣ ΑΠΟΤΕΛΕΣΜΑΤΩΝ** |
| 1 | "COVID" | 15.300552129745483 | 9 |
| 2 | "afghanistan" | 13.360227108001709 | 14 |
| 3 | “rights” | 16.6549813747406 | 20 |
| 4 | “Europe” | 20.230428457260132 | 15 |
| 5 | “DELTA” | 18.191984176635742 | 1 |
| 6 | “variant” | 16.870831966400146 | 0 |
| 7 | “strain” | 13.528109788894653 | 3 |
| 8 | “staTES” | 15.041808128356934 | 0 |
| 9 | “states” | 18.209105730056763 | 0 |
| 10 | “London” | 12.826951026916504 | 5 |
| 11 | “terror” | 13.561873197555542 | 0 |
| 12 | “terrorist” | 17.764453411102295 | 5 |
| 13 | “China” | 16.07418918609619 | 5 |
| 14 | “Africa” | 15.38338327407837 | 0 |
| 15 | “president” | 15.147238492965698 | 22 |
| 16 | “minister” | 18.38479256629944 | 19 |
| 17 | “tech” | 14.844147443771362 | 9 |
| 18 | “covid-19” | 16.081592321395874 | 7 |
| 19 | “health” | 14.131364822387695 | 11 |
| 20 | “hospitals” | 17.312322854995728 | 6 |
| ΑΠΟΚΡΙΣΗ: | | 318.900337458/20 = **15.9450168729** | |

### Ερωτήματα των δύο λέξεων

|  |  |  |  |
| --- | --- | --- | --- |
| **Α/Α** | **ΕΡΩΤΗΜΑ** | **ΑΠΟΚΡΙΣΗ** | ΑΡΙΘΜΟΣ ΑΠΟΤΕΛΕΣΜΑΤΩΝ |
| 1 | "COVID pandemic" | 24,5572957992553 | 17 |
| 2 | "Taliban Afghanistan" | 14,6520757675170 | 14 |
| 3 | "women's rights" | 20,1897675991058 | 20 |
| 4 | "Europian news" | 15,3587985038757 | 16 |
| 5 | "DELTA variant" | 15,9176502227783 | 1 |
| 6 | "new variant" | 20,0518517494201 | 0 |
| 7 | "virus strain" | 14,5852038860321 | 3 |
| 8 | "UniTED staTES" | 15,1919443607330 | 13 |
| 9 | "united states" | 13,9692251682281 | 13 |
| 10 | "London calling" | 15,0472526550292 | 5 |
| 11 | "terror ensues" | 17,9639163017272 | 0 |
| 12 | "terrorist attacks" | 15,1701669692993 | 5 |
| 13 | "China bans" | 16,9010603427886 | 5 |
| 14 | "Africa news" | 15,2066268920898 | 16 |
| 15 | "president announces" | 14,3326847553253 | 22 |
| 16 | "prime minister" | 19,6260535717010 | 24 |
| 17 | "tech university" | 16,1781138267517 | 14 |
| 18 | "covid-19 outbreak" | 14,0862631797705 | 8 |
| 19 | "health risk" | 18,7839462757110 | 16 |
| 20 | "hospitals capacity" | 16,6018626689910 | 6 |
| ΑΠΟΚΡΙΣΗ: | | 334,37176049613000/20 =**16,71858802480650** | |

### Ερωτήματα των τριών λέξεων

|  |  |  |  |
| --- | --- | --- | --- |
| **Α/Α** | **ΕΡΩΤΗΜΑ** | **ΑΠΟΚΡΙΣΗ** | **ΑΡΙΘΜΟΣ ΑΠΟΤΕΛΕΣΜΑΤΩΝ** |
| 1 | COVID pandemic outbreak | 16,67433142662040 | 17 |
| 2 | Taliban overtake Afghanistan | 17,42877650260920 | 14 |
| 3 | women's rights fight | 19,77321195602410 | 28 |
| 4 | Europian political news | 15,89884638786310 | 29 |
| 5 | DELTA variant vaccine | 14,95789742469780 | 5 |
| 6 | covid vaccination presentage | 21,04274678230280 | 11 |
| 7 | new variant discovered | 14,28748750686640 | 0 |
| 8 | new variant deadly | 15,00512385368340 | 2 |
| 9 | novel virus strain | 14,80378365516660 | 3 |
| 10 | virus strain contagious | 17,75211143493650 | 3 |
| 11 | UniTED staTES politics | 15,07178068161010 | 14 |
| 12 | united states military | 14,06390142440790 | 20 |
| 13 | terror in east | 15,11944913864130 | 17 |
| 14 | terrorist attacks city | 27,82066440582270 | 16 |
| 15 | China bans site | 17,20829010009760 | 5 |
| 16 | Africa news world | 16,26133847236630 | 29 |
| 17 | president new announcement | 17,80672764778130 | 25 |
| 18 | prime minister virus | 19,54525232315060 | 24 |
| 19 | tech univeristy america | 16,37763953208920 | 19 |
| 20 | covid-19 new outbreak | 16,33763265609740 | 8 |
| 21 | high health risk | 25,41943311691280 | 25 |
| 22 | low health risk | 19,26779222488400 | 16 |
| 23 | hospitals maximum capacity | 15,89395546913140 | 6 |
| 24 | intensive care unit | 22,11041879653930 | 32 |
| 25 | authorities take action | 17,96369910240170 | 28 |
| 26 | war in east | 17,54981493949890 | 17 |
| 27 | NATO announcement afghanistan | 15,78588581085200 | 18 |
| 28 | send aid war | 22,82213997840880 | 4 |
| 29 | Kabul street protests | 17,13792276382440 | 23 |
| 30 | US investigate report | 15,58383631706230 | 19 |
| ΑΠΟΚΡΙΣΗ: | | 532,77189183235000/30=**17,75906306107830** | |

### Ερωτήματα των τεσσάρων λέξεων

|  |  |  |  |
| --- | --- | --- | --- |
| **Α/Α** | **ΕΡΩΤΗΜΑ** | **ΑΠΟΚΡΙΣΗ** |  |
| 1 | COVID pandemic outbreak danger | 17,26794004440300 | 22 |
| 2 | Taliban overtake Afghanistan Kabul | 16,30406379699700 | 17 |
| 3 | women rights fight protest | 16,55181288719170 | 37 |
| 4 | Europian political news England | 15,24763941764830 | 29 |
| 5 | DELTA variant vaccine efficient | 20,13634443283080 | 5 |
| 6 | covid vaccination presentage low | 16,74112582206720 | 11 |
| 7 | new variant discovered danger | 14,88418984413140 | 7 |
| 8 | new variant deadly young | 15,00618100166320 | 7 |
| 9 | novel virus strain delta | 20,41378211975090 | 4 |
| 10 | virus strain contagious younger | 15,96265029907220 | 3 |
| 11 | UniTED staTES politics NEWS | 14,76692271232600 | 26 |
| 12 | united states military war | 24,16243672370910 | 20 |
| 13 | terror in middle east | 16,23118472099300 | 19 |
| 14 | terrorist attacks city Kabul | 15,93382310867300 | 25 |
| 15 | China bans web site | 24,49193906784050 | 5 |
| 16 | Africa news world today | 18,70172309875480 | 33 |
| 17 | president new covid announcement | 16,80822491645810 | 30 |
| 18 | prime minister virus lockdown | 23,54892182350150 | 24 |
| 19 | texas tech univeristy america | 19,20056796073910 | 19 |
| 20 | covid-19 new outbreak contagion | 17,73958039283750 | 8 |
| 21 | high health risk old | 19,05057597160330 | 25 |
| 22 | low health risk younger | 21,60628390312190 | 16 |
| 23 | hospitals maximum capacity vaccine | 16,66109728813170 | 9 |
| 24 | healthcare intensive care unit | 20,68185043334960 | 33 |
| 25 | authorities take action law | 21,00982093811030 | 38 |
| 26 | war in east US | 19,23667573928830 | 17 |
| 27 | NATO announcement afghanistan aid | 26,39075541496270 | 18 |
| 28 | send aid war work | 21,85352635383600 | 4 |
| 29 | Kabul street protests rights | 25,25489521026610 | 38 |
| 30 | US investigate report war | 17,82209324836730 | 19 |
| ΑΠΟΚΡΙΣΗ: | | 569,66862869262500/30=  18,98895428975420 | |

Μέση απόκριση: **17.5571261848** seconds

# Αναφορές

1. **Scrapy.** Scrapy. [Ηλεκτρονικό] https://scrapy.org/.

2. **Crummy.** Beautiful Soup 4. [Ηλεκτρονικό] https://www.crummy.com/software/BeautifulSoup/.

3. **NLTK.** Natural Language Toolkit. [Ηλεκτρονικό] https://www.nltk.org/.

4. **Dustman, Andy.** Welcome to MySQLdb's documentation! *mysqlclient.* [Ηλεκτρονικό] [Παραπομπή: 4 9 2021.] https://mysqlclient.readthedocs.io/.

5. **XML Processing Modules. *Python.* [Ηλεκτρονικό] Python Software Foundation. [Παραπομπή: 7 9 2021.] https://docs.python.org/3/library/xml.html.**