ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

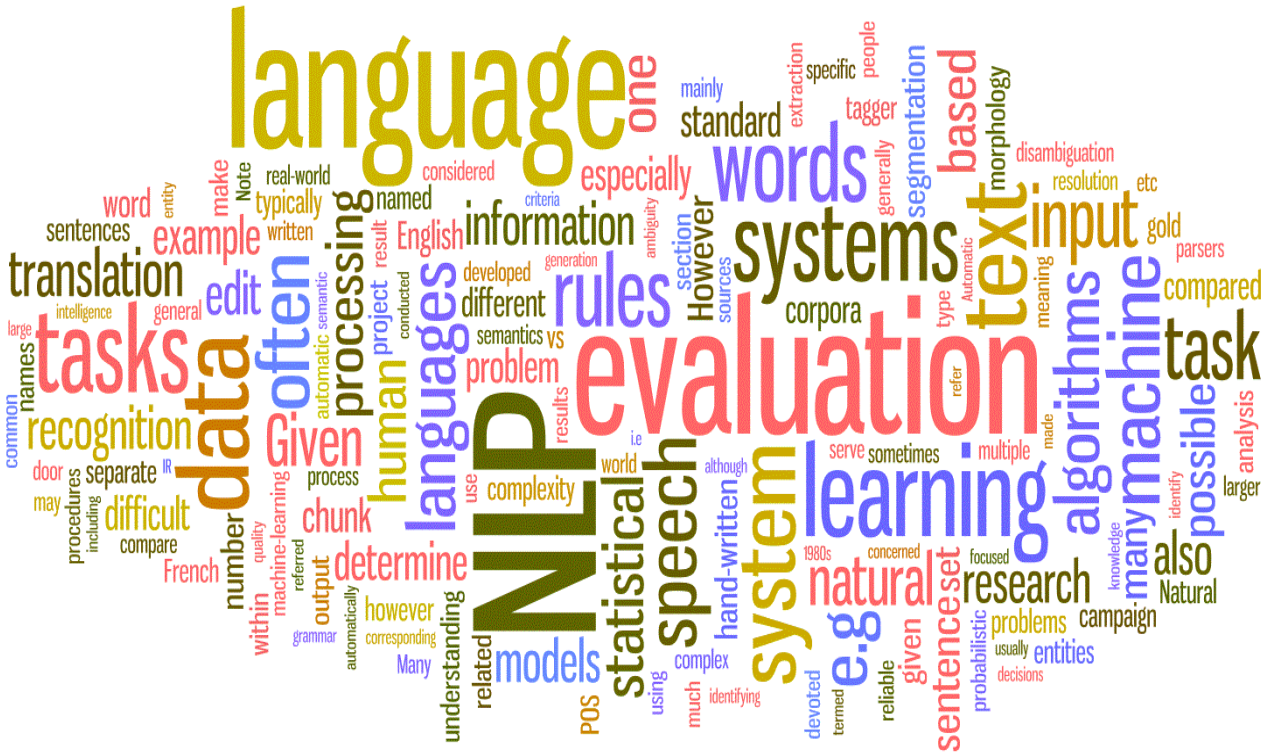
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΕΡΓΑΣΙΑ ΕΞΑΜΗΝΟΥ

Γλωσσική Τεχνολογία

Χειμερινό Εξάμηνο 2017-18



Ομάδα:

Κυριακού Ανδρόνικος 5806

Ντενέζος Παναγιώτης 5853

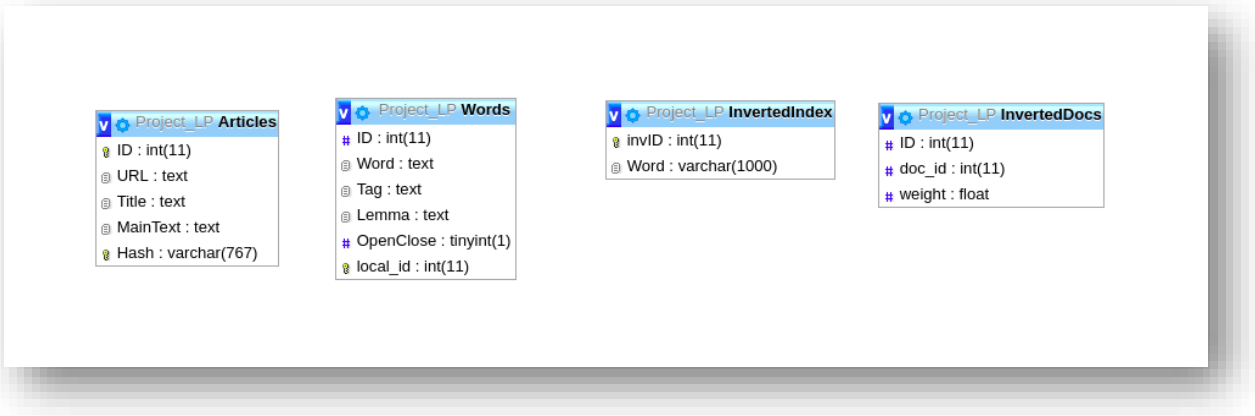
Περιεχόμενα

Μέρος Α

Η εργασία αυτή έχει σκοπό την δημιουργία ενός ολοκληρωμένου συστήματος συγκομιδής και δεικτοδότησης ιστοσελίδων. Η γλώσσα που χρησιμοποιήθηκε για την υλοποίηση ήταν η Python καθώς και επιμέρους εργαλεία που παρουσιάζονται παρακάτω. Για την παρουσίαση της εργασία θα αναφερθούμε στα αυτόνομα μέρη που υλοποιήθηκαν.

Βάση Δεδομένων

Για τους σκοπούς της άσκησης χρησιμοποιήθηκε μια σχεσιακή βάση δεδομένων MySQL η οποία περιέχει τους εξής πίνακες.



Προσκομιστής Ιστοσελίδων

Ο προσκομιστής ιστοσελίδων είναι ένα εργαλείο το οποίο αναλαμβάνει να παρακολουθεί μια σειρά από σελίδες και να κατεβάζει νέα άρθρα όταν αυτά είναι διαθέσιμα. Για τους σκοπούς της εργασίας, χρησιμοποιήθηκε ο προσκομιστής Scrapy[1] ο οποίος και ρυθμίστηκε προκειμένου να προσπελαύνει την κατηγορία των ειδήσεων του γνωστού ιστοτόπου reddit [2]. Οι ρυθμίσεις που έγιναν προκειμένου να λάβουμε ικανοποιητικό αριθμό δεδομένων ήταν ότι σε κάθε εκτέλεση ακολουθείται ο σύνδεσμος Next που υπάρχει σε μορφή κουμπιού στο τέλος της σελίδα και άρα προσπελαύνονται συνολικά 200 άρθρα κάθε φορά. Ο προσκομιστής αυτός εκτελέστηκε ανά τακτά χρονικά διαστήματα στην περίοδο από 27/12/2017 έως 10/01/2018 και τα αποτελέσματα αποθηκεύτηκαν σε επιμέρους αρχεία μορφής json τα οποία είχαν σαν όνομα την ημερομηνία εκτέλεσης. Για την οργανωμένη

[1] https://scrapy.org/
[2] https://www.reddit.com/r/news/

αποθήκευση χρησιμοποιήθηκαν τρεις πληροφορίες οι οποίες ήταν ο τίτλος ενός άρθρου, το περιεχόμενο του καθώς και η ιστοσελίδα από την οποία ανακτήθηκε. Τα κείμενα που συλλέχθηκαν αποθηκεύτηκαν στη βάση δεδομένων. Προκειμένου να αποφευχθεί η διπλή εισαγωγή ενός άρθρου, το URL κωδικοποιήθηκε με base64 κωδικοποίηση και αποθηκεύτηκε σαν μοναδικό χαρακτηριστικό. Παράλληλα, πριν την εισαγωγή ενός κειμένου στην βάση έγινε ένα πρώτο στάδιο προεπεξεργασίας στο οποίο απομονώθηκε το κειμενικό περιεχόμενο (αφαίρεση HTML tags, συνδέσμων κτλ). Προκειμένου να το επιτύχουμε αυτό χρησιμοποιήσαμε το πακέτο Goose Extractor[3] το οποίο εκτελέστηκε μέσα σε ένα virtual environment της python. Σημαντικό είναι να αναφερθεί ότι κατά την εκτέλεση του προσκομιστή, έγινε η κατάλληλη ρύθμιση προκειμένου να υπακούεται το αρχείο robots.txt του reddit και ο αριθμός των αιτημάτων να μην προκαλεί κατάχρηση των πόρων του. Συνολικά συγκεντρώθηκαν 1.158 ξεχωριστά άρθρα.

Προεπεξεργασία Δεδομένων

Η προεπεξεργασία των δεδομένων γίνεται πριν την αποθήκευση στην βάση δεδομένων και περιγράφεται στο αρχείο *store.py* και συγκεκριμένα στην συνάρτηση *load_database()*. Πιο συγκεκριμένα αντικαθιστούνται στο κείμενο οι χαρακτήρες /, \, *, @, ., =, ^, % με κενά και όλα τα γράμματα μετατρέπονται σε πεζά.

Μορφοσυντακτική Ανάλυση

Πριν από το βήμα αυτό εισάγουμε στον πίνακα Articles τα δεδομένα που έχουμε προς το παρόν για το κάθε κείμενο προκειμένου να το αποθηκεύσουμε στην αρχική του εκδοχή. Τα δεδομένα που έχουμε επιλέξει να αποθηκεύσουμε είναι το URL του (έτσι ώστε να μπορούμε να το προσπελάσουμε ξανά αν είναι απαραίτητο), τον τίτλο του, το κυρίως κείμενο και το hash που αναφέρθηκε παραπάνω. Προκειμένου να εκτελεστεί επιτυχώς η μορφοσυντακτική ανάλυση γίνεται πρώτα ένα tokenization (χρήση *nltk.word_tokenize()*) των κειμένων και έπειτα αντιστοιχίζεται συντακτική ετικέτα σε κάθε λέξη (χρήση *nltk.pos_tag()*). Η υλοποίηση της αντιστοίχισης γίνεται με χρήση του Natural Language Toolkit (NLTK)[4]. Έπειτα, ακολουθεί η εύρεση του λήμματος (χρήση *lmtzr.lemmatize()*) από το οποίο

[3] https://pypi.python.org/pypi/goose-extractor/
[4] http://www.nltk.org/

προέρχεται η κάθε λέξη. Για να το πετύχουμε αυτό, αποθηκεύουμε σε μια λίστα τα tags τα οποία αντιστοιχούν σε ρήματα {"VB","VBD","VBG","VBN","VBP","VBZ"} και αν η προς λημματοποίηση λέξη είναι ρήμα χρησιμοποιούμε τον WordNetLemmatizer με την επιλογή "v" ενώ διαφορετικά την προκαθορισμένη υλοποίηση που αναφέρεται σε ουσιαστικά. Τέλος, έγινε ο διαχωρισμός των τερματικών όρων (stop words). Ανακτήθηκε από την σελίδα infogistics[5] η λίστα με τις ετικέτες που αντιστοιχούν σε τερματικούς όρους και η κάθε λέξης χαρακτηρίστηκε ανάλογα. Μετά από την εκτέλεση των παραπάνω βημάτων οι πληροφορίες για κάθε λέξη ενός κειμένου αποθηκεύτηκε στον πίνακα Words.

Αναπαράσταση Ιστοσελίδων στο Μοντέλο Διανυσματικού Χώρου

Ο υπολογισμός των βαρών για κάθε όρο έγινε με χρήση ερωτημάτων στην βάση δεδομένων. Η αποθήκευση των βαρών έγινε σε μια δομή δεδομένων της Python που ονομάζεται λεξικό και που αντιστοιχίζει κλειδιά με τιμές. Ως κλειδιά χρησιμοποιήθηκαν οι μοναδικοί όροι της συλλογής των κειμένων (λήμματα). Αρχικά, για κάθε κείμενο της συλλογής υπολογίστηκε το κανονικοποιημένο term frequency για κάθε όρο από τον τύπο $tf = \frac{tf_{όρου}}{μέγιστο\ tf\ στο\ κείμενο}$.
Έπειτα κάθε όρος εισήχθηκε στο λεξικό, ενώ αν υπήρχε ήδη καταχώρηση για κάποιον, η λίστα με το ζευγάρι (αναγνωριστικό κειμένου, βάρος) προσαρτήθηκε ανάλογα.

Σε ένα δεύτερο βήμα υπολογίστηκε το inverse document frequency κάθε όρου από τον τύπο:

$$idf = \frac{Αριθμός\ κειμένων\ συλλογής}{Αριθμός\ κειμένων\ που\ εμφανίζεται\ ο\ όρος}$$

Και ανανεώθηκε κατάλληλα το λεξικό.

Οι κώδικες για τα παραπάνω βρίσκονται στις συναρτήσεις *calculate_tf()* και *calculate_tf_idf()* αντίστοιχα.

[5] http://www.infogistics.com/tagset.html

Αποθήκευση του ανεστραμμένου ευρετηρίου

Για την αποθήκευση του ανεστραμμένου ευρετηρίου υλοποιήθηκαν δυο μεθοδολογίες. Αρχικά, έγινε εξαγωγή σε μορφή XML (συνάρτηση *export_to_xml()*). Για την εξαγωγή χρησιμοποιήθηκε το πακέτο *xml.etree.ElementTree*[6] το οποίο δημιούργησε την XML οντότητα και με χρήση του *xml.dom.minidom*[7] μορφοποιήθηκε σε κατάλληλη προς ανάγνωση μορφή και αποθηκεύτηκε στο αρχείο output.xml το οποίο και επισυνάπτεται. Παράλληλα με παρόμοια λογική αναπτύχθηκε κώδικας για το άνοιγμα ενός αρχείου xml και την αποθήκευση των περιεχομένων του σε μορφή ανεστραμμένου ευρετηρίου (αρχείο *load.py*).

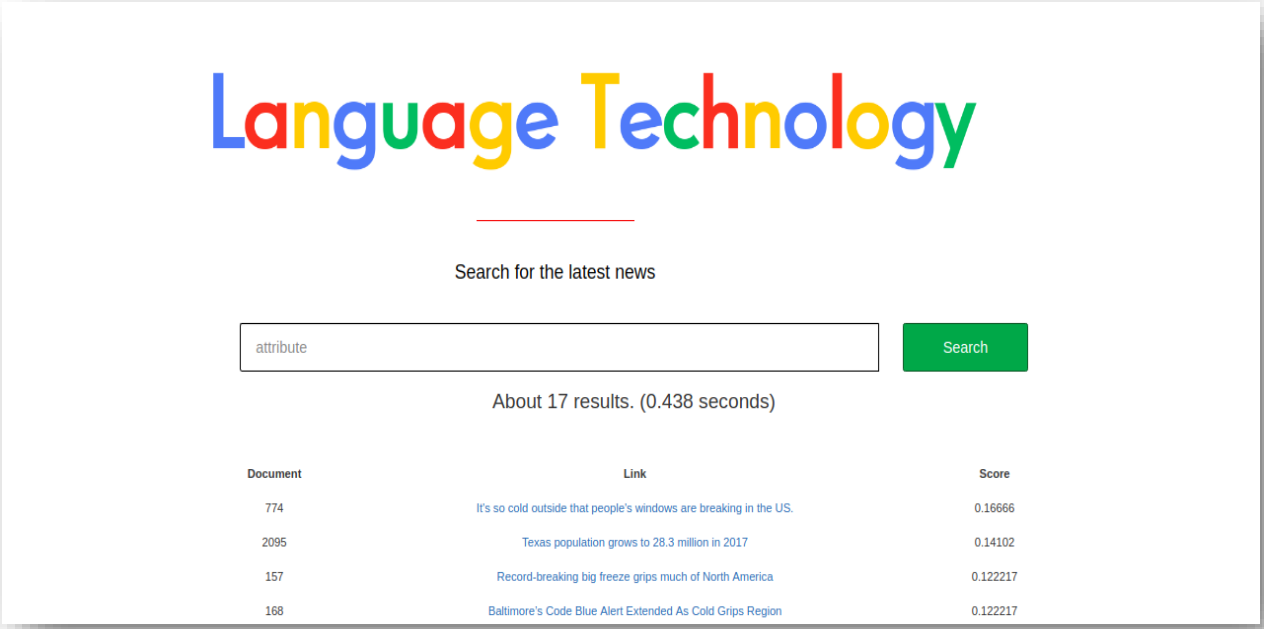
Η δεύτερη στρατηγική που ακολουθήθηκε, ήταν η αποθήκευση στην βάση δεδομένων προκειμένου να είναι διαθέσιμη από το σύστημα το οποίο θα δώσει γραφικό περιβάλλον και την δυνατότητα στον χρήστη να εκτελεί ερωτήματα (συνάρτηση *store_in_db()*). Ο τρόπος που έγινε η αποθήκευση μιμείται την μορφή του ανεστραμμένου ευρετηρίου καθώς δημιουργήθηκαν δύο πίνακες. Ο πρώτος με όνομα InvertedIndex αποθηκεύει όλες τις λέξεις του ευρετηρίου και ο δεύτερος με όνομα InvertedDocs αποθηκεύει τα ζευγάρια (αναγνωριστικό κειμένου, βάρος στο κείμενο). Είναι εμφανές ότι η σχέση που τους συνδέει είναι 1-N.

Μηχανισμός Υποβολής Ερωτημάτων

Για την δημιουργία του μηχανισμού υποβολής αναπτύχθηκε γραφικό περιβάλλον χρήστη. Χρησιμοποιήθηκαν γλώσσες του Web όπως HTML, CSS και Javascript για την εμφάνιση ενώ για το back-end αναπτύχθηκε κώδικας σε PHP. Για το λειτουργικό κομμάτι, αρχικά, λαμβάνεται η είσοδος του χρήστη και διαχωρίζεται στους επιμέρους όρους. Έπειτα, γίνεται ένα ερώτημα στην βάση και συγκεκριμένα στους πίνακες InvertedIndex και InvertedDocs όπου χρησιμοποιείται η δεσμευμένη λέξη LIKE της SQL με όρισμα τον κάθε όρο προκειμένου να επιτευχθεί string matching (χρησιμοποιείται ο χαρακτήρας % τόσο πριν όσο και μετά την λέξη έτσι ώστε να επιστραφούν όσο γίνεται περισσότερα αποτελέσματα). Η απάντηση της βάσης αποθηκεύεται σε ένα associative array με κλειδί το κάθε αναγνωριστικό κειμένου και τιμή το βάρος του όρου. Αν σε επόμενη αναζήτηση για κάποιο όρο επιστραφεί το ίδιο κείμενο τότε το βάρος προστίθεται στο ήδη υπολογισμένο. Τέλος, όταν

[6] https://docs.python.org/2/library/xml.etree.elementtree.html
[7] https://docs.python.org/2/library/xml.dom.minidom.html

καταναλωθούν όλοι οι όροι της εισόδου, γίνεται ένα ερώτημα στον πίνακα Articles και τα κείμενα που επιστρέφονται κωδικοποιούνται σε JSON μορφή για να επιστραφούν ασύγχρονα στην σελίδα αναζήτησης που βλέπει ο χρήστης. Ένα παράδειγμα αναζήτησης και επιστροφής αποτελεσμάτων φαίνεται στην παρακάτω εικόνα:



Αξιολόγηση Ευρετηρίου

Για την αξιολόγηση του ευρετηρίου υποβλήθηκαν 20 ερωτήματα της μιας λέξης, 20 ερωτήματα των δύο λέξεων, 30 ερωτήματα των τριών λέξεων, 30 ερωτήματα των τεσσάρων λέξεων και 30 ερωτήματα των πέντε λέξεων. Οι λέξεις επιλέχθηκαν τυχαία με χρήση της ιστοσελίδας textfixer[8]. Τα αποτελέσματα παρουσιάζονται στον παρακάτω πίνακα:

[8] https://www.textfixer.com/tools/random-words.php

	Ερωτήματα 1 Λέξης	Ερωτήματα 2 Λέξεων	Ερωτήματα 3 Λέξεων	Ερωτήματα 4 Λέξεων	Ερωτήματα 5 Λέξεων
1	0.626	0.711	0.993	1.404	1.72
2	0.53	0.717	1.032	1.33	1.642
3	0.467	0.7	1.021	1.359	1.69
4	0.516	0.701	1.049	1.354	1.683
5	0.373	0.696	0.997	1.345	1.662
6	0.365	0.684	0.994	1.35	1.678
7	0.353	0.821	1.157	1.332	1.663
8	0.437	0.741	1	1.343	1.657
9	0.356	0.702	1.09	1.338	1.695
10	0.353	0.701	1.008	1.361	1.683
11	0.371	0.79	1.038	1.374	1.658
12	0.361	0.678	1.01	1.366	1.741
13	0.378	0.718	1.024	1.372	1.668
14	0.349	0.663	1.001	1.333	1.652
15	0.341	0.689	1.014	1.321	1.659
16	0.402	0.73	1.01	1.328	1.662
17	0.365	0.685	1.001	1.335	1.687
18	0.378	0.679	1.027	1.361	1.685
19	0.364	0.686	1.017	1.347	1.677
20	0.358	0.701	0.998	1.369	1.651
21			1.018	1.38	1.724
22			1.042	1.37	1.675
23			1.005	1.338	1.654
24			0.999	1.319	1.625
25			1.028	1.37	1.687
26			1.083	1.38	1.727
27			1.004	1.328	1.657
28			0.992	1.344	1.681
29			1.017	1.373	1.672
30			0.986	1.334	1.671
ΜΟ	0.40215	0.70965	1.021833333	1.351933333	1.6762

Ο συνολικός μέσος χρόνος απόκρισης όπως μετρήθηκε είναι 1.1 δευτερόλεπτα.

Μέρος Β

Ο κώδικας για την υλοποίηση του μέρους Β βρίσκεται στον φάκελο Part2 και συγκεκριμένα στο αρχείο store.py.

Προεπεξεργασία των συλλογών

Αρχικά, επιλέχθηκαν δέκα συλλογές και απομακρύνθηκαν τα κείμενα τα οποία είχαν χαρακτήρες που δεν ήταν δυνατόν να αποθηκευτούν σε αρχείο XML (λόγω των κανόνων του προτύπου). Συνοπτικά αυτά ήταν τα κείμενα 58854, 53644, 53516, 61129, 176936, 178464, 58988, 7615,9170, 9174,52826, 38424. Έπειτα, για τα υπόλοιπα κείμενα έγινε μια παρόμοια διαδικασία με το Μέρος Α, όπου απομακρύνθηκαν περισσότεροι ειδικοί χαρακτήρες, όπως οι /, \, *, @, ., =, ^, %, ‘,’ , +, _, ‘, -, ! και έπειτα εισήχθησαν τα κείμενα και οι λέξεις στους αντίστοιχους πίνακες Mails και Words.

Ο κώδικας βρίσκεται στην συνάρτηση *load_database()*.

Δημιουργία Χώρου Χαρακτηριστικών

Όπως και παραπάνω για κάθε κείμενο υπολογίστηκε το tf-idf βάρος για κάθε όρο (συνάρτηση *calculate_tf_idf()*) και δημιουργήθηκε ένα ανεστραμμένο αρχείο το οποίο αποθηκεύτηκε τόσο σε μορφή XML (συνάρτηση *export_to_xml()*) στο αρχείο output2.xml όσο και στην βάση δεδομένων για μελλοντική χρήση (συνάρτηση *save_weights()*).

Παράλληλα υλοποιήθηκε η συνάρτηση *load_xml()*, η οποία διαβάζει ένα xml αρχείο και δημιουργεί το ανεστραμμένο ευρετήριο στην μνήμη.

Δημιουργία Διανυσματικών Χαρακτηριστικών

Η επιλογή των 8000 όρων με το μεγαλύτερο βάρος tf-idf γίνεται στην συνάρτηση *create_vectors()*, η οποία δημιουργεί ένα αραιό μητρώο στο οποίο η μια διάσταση έχει τα κείμενα και η άλλη τους όρους (συνολικές διαστάσεις 5528×8000).

Σύγκριση Διανυσμάτων Χαρακτηριστικών

Η συνάρτηση *calculate_similarity()* προσπελαύνει όλα τα κείμενα ελέγχου των κατηγοριών που έχουν επιλεγεί και για το καθένα κάνει την εξής διαδικασία. Αρχικά αφαιρούνται οι ειδικοί χαρακτήρες και βρίσκεται η συντακτική ετικέτα του κάθε όρου. Έπειτα, γίνεται λημματοποίηση των όρων και υπολογίζεται η συχνότητα εμφάνισης τους στο κάθε κείμενο. Για τον υπολογισμό της idf τιμής του όρου ανακτάται πληροφορία από την βάση και στο τελικό αποτέλεσμα γίνεται κανονικοποίηση ώστε να μην εξαρτάται από το μέγεθος του κειμένου. Ουσιαστικά, ο τύπος που χρησιμοποιείται για τον όρο k είναι :

$$w_k = \frac{tf_k \log(\frac{N}{n_k})}{\sqrt{\sum_{k=1}^{t}(tf_k)^2[\log(\frac{N}{n_k})]^2}}$$

Ουσιαστικά δημιουργούμε ένα ψευδοδιάνυσμα στον οποίο οι όροι – διαστάσεις ταυτίζονται με τους 8000 πιο σημαντικούς του ανεστραμμένου ευρετηρίου.

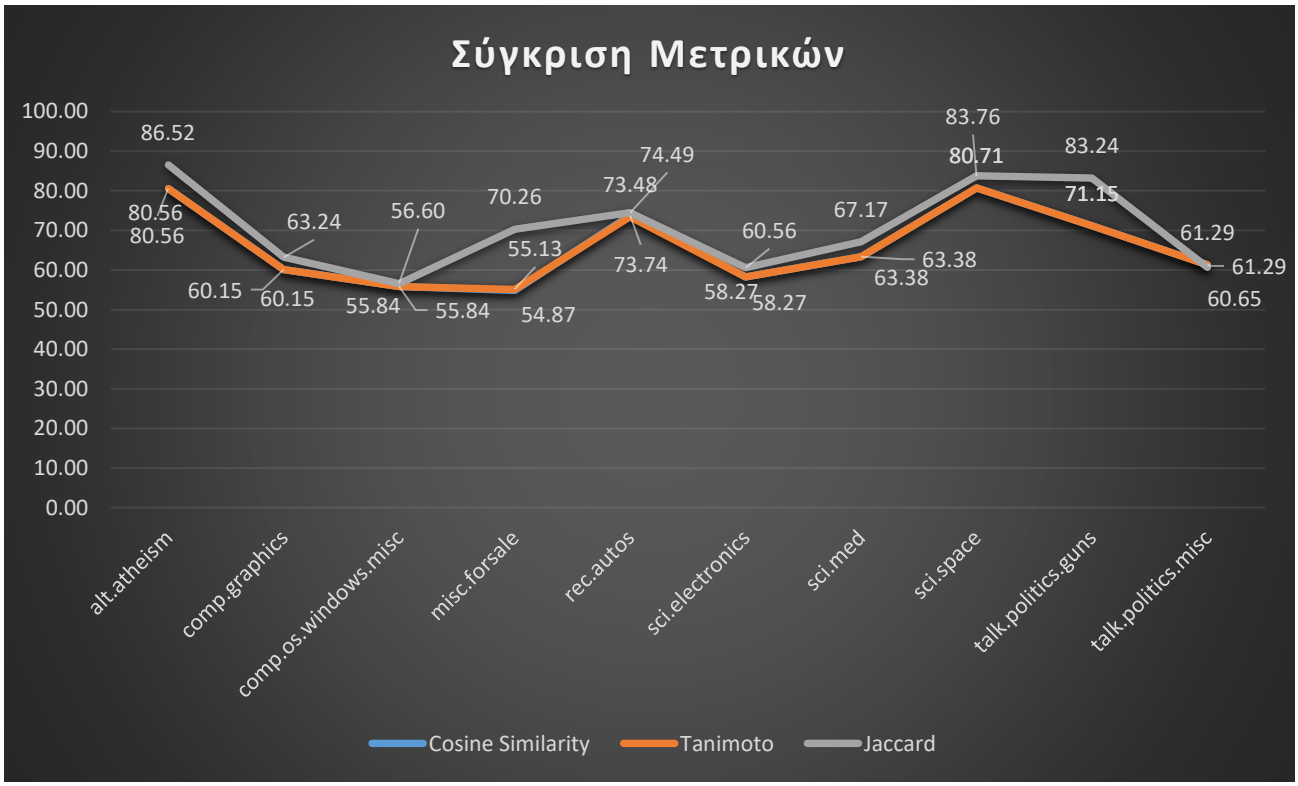
Στη συνέχεια υλοποιήθηκαν τρεις μετρικές:

- $Cosine\ Similarity(q,d) = \frac{g \times d}{|q||d|}$
- $Tanimoto(q,d) = \frac{q \times d}{q \times q + d \times d - q \times d}$
- $Jaccard(q,d) = \frac{|q \cap d|}{|q \cup d|}$

Οι μετρικές αυτές περιγράφονται στις αντίστοιχες συναρτήσεις *cosine_similarity()*, *tanimoto()* και *jaccard()*.

Τα αποτελέσματα από τα πειράματα με τα κατάλληλα έγγραφα παρουσιάζονται παρακάτω:

Κατηγορία	**# Κειμένων**	**Cosine Similarity**		**Tanimoto**		**Jaccard**	
		Ορθά	*Ποσοστό*	*Ορθά*	*Ποσοστό*	*Ορθά*	*Ποσοστό*
alt.atheism	319	257	80.56	257	80.56	276	86.52
comp.graphics	389	234	60.15	234	60.15	246	63.24
comp.os.windows.misc	394	220	55.84	220	55.84	223	56.60
misc.forsale	390	214	54.87	215	55.13	274	70.26
rec.autos	396	292	73.74	291	73.48	295	74.49
sci.electronics	393	229	58.27	229	58.27	238	60.56
sci.med	396	251	63.38	251	63.38	266	67.17
sci.space	394	318	80.71	318	80.71	330	83.76
talk.politics.guns	364	259	71.15	259	71.15	303	83.24
talk.politics.misc	310	190	61.29	190	61.29	188	60.65
Συνολικά	***3745***	***2464***	***65.79***	***2464***	***65.79***	***2639***	***70.47***



Συμπερασματικά, παρατηρείται ότι η μετρική του Jaccard αποδίδει καλύτερα από τις μετρικές Cosine Similarity και Tanimoto οι οποίες δίνουν πολύ κοντινά αποτελέσματα.