

ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ



- Πτυχιακή Εργασία -

Αξιοποίηση κοινωνικών δικτύων για την μελέτη των ηλεκτρονικών μέσων ενημέρωσης

Αθανάσιος Πιπλικάτσης

Επιβλέπων: Εμμανουήλ Μαραγκουδάκης

23 Μαρτίου 2022

Επιβλέπων

Εμμανουήλ Μαραγκουδάκης, Καθηγητής

Ιόνιο Πανεπιστήμιο

Τριμελής Επιτροπή

Εμμανουήλ Μαραγκουδάκης, Καθηγητής

Ιόνιο Πανεπιστήμιο

Κάτια - Λήδα Κερμανίδου, Αναπληρώτρια Καθηγήτρια

Ιόνιο Πανεπιστήμιο

Βασίλειος Καρυώτης, Αναπληρωτής Καθηγητής

Ιόνιο Πανεπιστήμιο

Περίληψη

Το παρών εγχείρημα προσπαθεί να εντοπίσει κατά πόσο τα ειδησεογραφικά στην Ελλάδα ακολουθούν τις τάσεις στο Twitter της Ελλάδας αλλά και το αντίστροφο. Οι τάσεις του Twitter αντικατοπτρίζουν όλες τις συζητήσεις που γίνονται γύρω από κάποια θέματα της επικαιρότητας. Με λίγα λόγια, μας φανερώνουν τι συζητιέται σε πανελλαδικό επίπεδο κάθε χρονική στιγμή. Έχοντας αυτή την πληροφορία διαθέσιμη καθίσταται δυνατό με τη χρήση μηχανικής μάθησης να βρεθούν για κάθε μία τάση τα αντίστοιχα άρθρα από τα ειδησεογραφικά. Έτσι δημιουργούνται συσχετίσεις μεταξύ των τάσεων του Twitter και των άρθρων από τα ειδησεογραφικά, σε σχέση πάντα με αυτά που γράφουν και οι δύο πλευρές.

Από την μία πλευρά το Twitter λειτουργεί χωρίς ιεραρχία και όλες οι πληροφορίες για όλα τα γεγονότα αποτελούνται από κάθε ξεχωριστό μεμονωμένο χρήστη. Κάθε χρήστης μπορεί να έχει την δικιά του υποκειμενική άποψη και να αντιλαμβάνεται τα γεγονότα έτσι όπως ορίζει ο ίδιος.

Από την άλλη πλευρά τα ειδησεογραφικά λειτουργούν με ιεραρχία και τα γεγονότα προβάλλονται μέσα από τα δικά τους ειδικά κατασκευασμένα γυαλιά. Κάθε ειδησεογραφικό έχει το δικό του κοινωνικοπολιτικό πλαίσιο που σύμφωνα με αυτό αντιλαμβάνεται τα γεγονότα και έτσι αρθρογραφεί.

Έχοντας αυτά κατά νου, οι συσχετίσεις που δημιουργούνται στο παρών εγχείρημα μπορεί να είναι δυνατές ή αδύναμες. Αν είναι δυνατή η συσχέτιση, αυτό σημαίνει ότι η τάση είναι συνυφασμένη με το άρθρο και αντίστροφα. Αν είναι αδύναμη, τότε είτε ο αλγόριθμος δεν κατάφερε να βρει κάποια μεγάλη ομοιότητα, είτε δεν υπάρχει αντίστοιχο άρθρο γιατί δεν γράφτηκε ποτέ. Αυτό είναι στο χέρι του αναγνώστη να το εντοπίσει με τα εργαλεία που παρέχονται στην διεπαφή του εγχειρήματος και με την δικιά του αντιληπτική ικανότητα.

Τέλος, το αποτέλεσμα είναι μια διαδικτυακή σελίδα διεπαφή, που παρέχονται διάφορες πληροφορίες σχετικά με τις συσχετίσεις που βρέθηκαν. Παρέχεται μηχανή αναζήτησης των άρθρων και των τάσεων για περαιτέρω διερεύνηση. Δίνεται η δυνατότητα προβολής της δημοτικότητας των τάσεων σε σχέση με το χρόνο. Καθώς και άλλες πληροφορίες που αυξάνονται και ανανεώνονται συνεχώς με το χρόνο προς εξερεύνηση.

Περιεχόμενα

Εισαγωγή	7
Παραπληροφόρηση στα Κοινωνικά Δίκτυα και στα Μέσα Μαζικής Ενημέρωσης	9
1. Συλλογή και Κατανόηση των Δεδομένων	11
1.1 Twitter.....	11
1.2 Οι τάσεις στο Twitter.....	12
1.3 Twitter API	12
1.4 Scraping Ειδησεογραφικών	13
2. Βάση Δεδομένων	16
2.1 Schema των μοντέλων	17
2.2 Οι CRUD διαδικασίες της Βάσης Δεδομένων.....	22
2.3 Ο ρόλος της βάσης δεδομένων στον client.....	24
3. Τα Δεδομένα	27
3.1 Διπλότυπα	27
3.2 Missing δεδομένα	28
3.3 Καθαρισμός δεδομένων	28
3.4 Stemming.....	29
4. Επεξεργασία των δεδομένων	31
4.1 Η αναπαράσταση των εγγράφων με Bag of Words	31
4.2 Κανονικοποίηση των BoW Διανυσμάτων	32
4.3 Συνημιτονοειδής ομοιότητα μεταξύ διανυσμάτων	33
4.4 Η προσπάθεια να ενσωματωθεί ο αλγόριθμος word2vec	34
4.5 Σφαιρική εικόνα όλης της διαδικασίας της επεξεργασίας δεδομένων.....	37
4.6 Διαχείριση των σφαλμάτων που προκύπτουν κατά την διάρκεια εκτέλεσης του προγράμματος	38
5. Ο διακομιστής NodeJS - express	40
5.1 Τρόπος λειτουργίας ενός διακομιστή ιστού.....	40
5.2 REST API ή αλλιώς ο τρόπος επικοινωνίας μεταξύ server και client.....	41
5.3 REST API με την express.js βιβλιοθήκη	41
5.4 Το REST API του εγχειρήματος.....	43
5.5 Ρυθμίσεις της NodeJS και express εφαρμογής	44
5.6 Το «πακετάρισμα» του όλου εγχειρήματος (Dockerize)	45

6. Οπτικοποίηση & Συμπεράσματα των δεδομένων.....	47
6.1 Περιήγηση στην εφαρμογή	47
6.2 Η επιλογή του Web Framework Angular.....	51
6.3 Τα χαρακτηριστικά της Angular εφαρμογής	51
6.4 Ερμηνεία αποτελεσμάτων της εφαρμογής.....	52
Βιβλιογραφία	55
Υπερσύνδεσμοι	57
Παράρτημα - User Manual	59
Παράρτημα – Deployment.....	63
Παράρτημα – Source Code	64

Εισαγωγή

Με την άνθιση του διαδικτύου τις τελευταίες δεκαετίες η εύκολη και γρήγορη μεταφορά πληροφοριών σε όλα τα μήκη και πλάτη της γης, μας έχει οδηγήσει στην εποχή της πληροφορίας. Υπολογίζεται ότι την χρονιά 2021 υπήρξαν 4,66 δισεκατομμύρια ενεργοί χρήστες του διαδικτύου¹ ικανοί να καταναλώσουν περιεχόμενο καθώς και να παράγουν. Η ροή της πληροφορίας στο διαδίκτυο γίνεται με υψηλούς ρυθμούς και αυτό οφείλεται εν μέρη με τον ερχομό του Web 2.0 (Ιστός 2.0). Η νέα γενιά του παγκόσμιου ιστού βασίζεται στην όλο και μεγαλύτερη δυνατότητα των χρηστών του διαδικτύου να παράγουν περιεχόμενο. Γνωστές εφαρμογές του Web 2.0 είναι τα γνωστά σε όλους μας μέσα κοινωνικής δικτύωσης (social media) Facebook, YouTube, Twitter κ.α., όπου μέσα από αυτά δίνεται η δυνατότητα στους χρήστες να γράψουν ό,τι αφορά την καθημερινότητα τους, την άποψη και κριτική τους θέση πάνω σε κάποιο ζήτημα καθώς και ό,τι μπορεί να συζητάνε στην πραγματική τους ζωή να το μοιράζονται με άλλους χρήστες στα κοινωνικά δίκτυα. Πέρα από οποιαδήποτε πληροφορία στον παγκόσμιο ιστό έχει ενδιαφέρον να δούμε κατά πόσο και αν επηρέασε το Web 2.0 την διάχυση των ειδήσεων στις κοινωνίες. Περισσότερα από 3 δισεκατομμύρια άτομα από όλο τον κόσμο χρησιμοποιούν τα social media κάθε μήνα, πολλοί από αυτούς τα χρησιμοποιούν για να ενημερώνονται για το τι συμβαίνει τόσο τοπικά αλλά και παγκόσμια. Ακόμα και οι παραδοσιακές εφημερίδες έχουν μεταφερθεί στα social media για να εναγκαλισθούν αυτήν την αλλαγή. Οι ειδήσεις στο διαδίκτυο, δηλαδή γεγονότα που συμβαίνουν σε μια κοινωνία, μπορούν να βρεθούν από ειδησεογραφικούς φορείς μέσα από τα άρθρα που δημοσιεύουν. Κάθε άνθρωπος έχει την δική του αντίληψη, έτσι οι ειδησεογραφικοί φορείς έχουν συντάκτες οι οποίοι μεταδίδουν τα γεγονότα έτσι όπως τα ερμηνεύουν οι ίδιοι. Με παρόμοιο τρόπο αλλά όχι επαγγελματικά οι χρήστες των social media μπορεί να δημοσιεύσουν κάποια είδηση, η οποία δεν έχει δημοσιευθεί από τα ειδησεογραφικά. Αυτό μπορεί να έχει συμβεί για πολλούς λόγους, μια προφανής αιτία είναι τα ειδησεογραφικά να μην γνωρίζουν για αυτήν την είδηση. Άλλη αιτία είναι να γνωρίζουν για την ύπαρξη της και να μην την δημοσίευσαν διότι μπορεί να μην άξιζε να δημοσιευθεί ή για οποιοδήποτε άλλο λόγο που μπορεί να έχει ένας ειδησεογραφικός φορέας.

Όσο αφορά τα κοινωνικά δίκτυα, έχουν υπάρξει έρευνες που ελέγχουν την αξιοπιστία των δημοσιεύσεων από τους χρήστες του Twitter, με σκοπό να αναγνωρίσουν μέσω της μηχανικής μάθησης εάν υπάρχει ορθότητα στα μηνύματα τους. Οι Castillo κ.α. ήταν η πρώτη ομάδα που εργάστηκε πάνω στο θέμα της αξιοπιστίας αυτών των δημοσιεύσεων (Castillo et al, 2011). Είχαν προτείνει ένα μοντέλο που με αυτόματο τρόπο κατηγοριοποιούσε τα tweets βάσει των χαρακτηριστικών που εξάγονταν από αυτά. Αυτά τα χαρακτηριστικά εκπαιδεύτηκαν σε διάφορους κατηγοριοποιητές όπως τον SVM, Bayesian μοντέλο και δέντρα απόφασης, παρουσίασαν σχεδόν 86% ακρίβεια χρησιμοποιώντας το J48 αλγόριθμο δέντρο απόφασης. Η ίδια μελέτη ανέφερε επίσης ότι τα καλύτερα χαρακτηριστικά ήταν συσχετισμένα με χρήστες που χρησιμοποιούσαν περισσότερο την πλατφόρμα Twitter (μεγαλύτερος χρόνος που ξόδευαν για να ασχολούνται με αυτήν και υψηλό αριθμό followers, δηλαδή το κοινό τους). Άλλη έρευνα ασχολήθηκε με την αξιοπιστία φημολογημάτων που υπήρχαν στο Twitter (Zubiaga et al, 2017). Χρησιμοποιώντας tweets που επισημείωσαν δημοσιογράφοι με ετικέτες που δήλωναν αν είναι φήμες ή όχι κατασκεύασαν ένα μοντέλο κατηγοριοποίησης μέσω του στατιστικού μοντέλου CRF (Conditional Random Fields). Τα χαρακτηριστικά που χρησιμοποίησαν ήταν η διανυσματοποίηση των κειμένων μέσω του Word2Vec αλγόριθμου, αριθμό λέξεων καθώς και σημεία στίξης. Το αποτέλεσμα ήταν 68% του μέτρου ακριβείας και έδειξαν ότι πέρα των μεταδεδομένων των tweets τα χαρακτηριστικά των κειμένων των ίδιων των tweets είναι καλοί δείκτες.

Σκοπός αυτού του εγχειρήματος είναι να προσφέρει ένα εργαλείο το οποίο ερευνητές μπορούν να χρησιμοποιήσουν ώστε να εντοπίσουν την εγκυρότητα άρθρων σε σχέση με τις τάσεις του Twitter. Στην πλατφόρμα του Twitter μία είδηση εξιστορείται μέσα από τα tweets των τάσεων. Τα ειδησεογραφικά μπορεί να παραλείψουν κάποια είδηση, η οποία στην πλατφόρμα Twitter μπορεί να λάβει δημοσιότητα μέσα από τις δημοσιεύσεις των χρηστών της. Το αντίστροφο σίγουρα συμβαίνει, δηλαδή είναι δεδομένο ότι οι τάσεις του Twitter σίγουρα θα παραλείπουν ειδήσεις μιας και δεν είναι αυτός ο σκοπός τους. Αν όμως όταν έχει συμβεί ένα γεγονός που εμφανίζεται

¹ <https://www.statista.com/statistics/617136/digital-population-worldwide/>

στις τάσεις του Twitter, και τα ειδησεογραφικά δεν αναφέρουν κάτι, αυτό ενδεχομένως να σημαίνει ότι τα ειδησεογραφικά συνειδητά δεν ανέρτησαν κάτι για αυτό. Επιπλέον, τα social media έχουν την ικανότητα να ασκούν πιέσεις στο σύνολο της κοινωνίας και κατ' επέκταση στα ειδησεογραφικά ώστε να λάβει ένα θέμα την ανάλογη προσοχή (Μουζακίτη, 2012). Πολλές φορές οι τάσεις που εμφανίζονται στο Twitter είναι στην ουσία τους τροποποιημένες αναδημοσιεύσεις που έχουν προέλθει από τα ειδησεογραφικά, ενίοτε όμως συμβαίνει το αντίθετο. Όταν συμβαίνει, λοιπόν, οι τάσεις του Twitter να εμφανίζονται πρώτες σαν είδηση από τα ειδησεογραφικά μπορεί να έχει ασκηθεί πίεση από τη διαδικτυακή κοινότητα ώστε να αναφερθεί στα ειδησεογραφικά. Όταν τα ειδησεογραφικά και οι τάσεις του Twitter μιλάνε για μια είδηση, το περιεχόμενο που έχουν αυτά τα δύο μεταξύ τους μπορεί να είναι τελείως διαφορετικό. Δηλαδή μπορεί οι ιστορίες που περιγράφονται να είναι αντικρουόμενες, τα γεγονότα να σχολιάζονται με θετικό ή αρνητικό πρόσημο, καθώς και να παραλείπονται ή να προστίθενται διάφορες πτυχές που μπορεί να έχουν τα γεγονότα. Θεωρούμε ότι είναι σκόπιμο να υπάρχει ένα εργαλείο ικανό να καταγράφει τέτοια φαινόμενα ώστε να είναι δυνατόν να εξασκηθεί η περαιτέρω διερεύνηση από οποιονδήποτε ενδιαφερόμενο.

Έτσι σχεδιάστηκε μια εφαρμογή, η οποία, συλλέγει tweets και άρθρα από ειδησεογραφικά με σκοπό να δημιουργήσει συσχετίσεις ομοιότητας μεταξύ τους. Η ομοιότητα εδώ ορίζεται περισσότερο στο κατά πόσο τα άρθρα και tweets ταιριάζουν μεταξύ τους με βάση τις κοινές τους λέξεις, παρά στο ότι μοιράζονται κοινές απόψεις και γνώμες. Αυτό είναι στο χέρι του χρήστη της εφαρμογής να ανακαλύψει με βάση όλα τα στοιχεία που δίνονται από την εφαρμογή. Η εφαρμογή είναι σε θέση να συλλέγει τις τάσεις του Twitter την στιγμή που συμβαίνουν και να ψάχνει στα ειδησεογραφικά το πιο σχετικό με αυτές άρθρο. Καταγράφεται η ημερομηνία που πρωτοεμφανίζεται μία τάση και η εξέλιξη της στον χρόνο καθώς και το πιο σχετικό άρθρο που βρέθηκε γι' αυτή. Παρέχεται λειτουργία αναζήτησης πίσω στον χρόνο όπως συνέβαινε εκείνη την στιγμή η εξέλιξη της διαδικασίας εύρεσης ομοιότητας μεταξύ άρθρων και τάσεων. Στο σύνολο της η εφαρμογή είναι ένα εργαλείο για τους ερευνητές που στόχο έχει να τους βοηθήσει να κατανοήσουν τα σύγχρονα ελαττώματα των ειδησεογραφικών στην Ελλάδα καθώς και από την άλλη πλευρά την εξέταση και κατανόηση των δημοσιεύσεων που πραγματοποιούνται στην πλατφόρμα του Twitter.

Παραπληροφόρηση στα Κοινωνικά Δίκτυα και στα Μέσα Μαζικής Ενημέρωσης

Η ποιοτική δημοσιογραφία καθώς και το πλήθος ελεύθερων και ισχυρών ειδησεογραφικών είναι απαραίτητα στοιχεία μιας δημοκρατικής κυβέρνησης. Οι ακαδημαϊκοί έχουν δώσει έμφαση στην σημαντικότητα των ελεύθερων και ισχυρών ειδησεογραφικών ώστε αυτά να ενημερώνουν μαζικά τους πολίτες μιας κοινωνίας, να παρουσιάζουν διαφορετικές σκοπιές, να ενθαρρύνουν το δημόσιο διάλογο και την κοινωνική ενσυναίσθηση. Οι διαφορετικές σκοπιές σε μία είδηση έχουν μεγάλο αντίκτυπο αν αναλογιστούμε την σημασία τους στην επίπτωση των δημόσιων σχέσεων. Είναι σημαντικό να υπάρχουν διαφορετικές οπτικές πάνω σε ένα ζήτημα ώστε να προβλεφθεί η πιθανή κατάχρηση των ειδησεογραφικών μονοπωλίων, ιδιωτικών ή δημόσιων (R. K Nielsen, 2012). Τί συμβαίνει όμως σε περιπτώσεις όπου δεν χρησιμοποιείται ο κώδικας ποιοτικής δημοσιογραφίας;

Λόγω της ελεύθερης διάδοσης της πληροφορίας υπάρχει ο κίνδυνος της λανθασμένης ή παραπλανητικής πληροφορίας. Η λανθασμένη ή η παραπλανητική πληροφορία που παρουσιάζεται σαν γεγονός εκούσια ή ακούσια αποτελεί κακή πληροφορία (misinformation). Ο όρος της παραπληροφόρησης (disinformation) ορίζεται με την σκοπίμη διάδοση, διοχέτευση ψευδών πληροφοριών με στόχο ή με αποτέλεσμα την παραπλάνηση και την σύγχυση του αποδέκτη³. Οι τεχνικές της, για να παραποιηθεί η πληροφορία ποικίλουν, όπως η αξιολόγηση μιας ασήμαντης είδησης σαν σημαντική αλλά και το αντίθετο. Η αποσιώπησή θεμελιώδους σημασίας ειδήσεων, η αποσπασματικοποίηση μιας ιστορίας, η δήθεν ιδεολογική ακεραιότητα των γεγονότων, η υπερπληροφόρηση, η δραματοποίηση των ειδήσεων κ.α. αποτελούν εργαλεία της παραπληροφόρησης για την διαστρέβλωση μιας είδησης.

Η ηλεκτρονική μελέτη σε 28 χώρες διάρκειας 17 χρόνων και με περισσότερους από 33.000 πολίτες (Edelman Trust Barometer 2017)⁴ φανερώνει ότι η εμπιστοσύνη σε τέσσερα κλειδιά θεσμούς (επαγγελματικός, κυβερνητικός, ΜΚΟ, ΜΜΕ) όσο αφορά την ενημέρωση όλο και μειώνεται. Σε αρκετά μέρη παγκόσμια, η εμπιστοσύνη στα ΜΜΕ και γενικότερα στην δημοσιογραφία είναι ευάλωτες και σε πτώση. Βρέθηκε ότι οι περισσότεροι πολίτες βασίζονται σε μηχανές αναζήτησης στο διαδίκτυο για να ενημερώνονται (64%). Ακολουθούν τα παραδοσιακά μέσα ενημέρωσης (57%), ηλεκτρονικά μέσα ενημέρωσης (51%) (πχ ειδησεογραφικές σελίδες), ιδιωτικά μέσα ενημέρωσης (43%) και τα κοινωνικά δίκτυα (41%).

Υπάρχουν κάποιες ηθικές αρχές που πρέπει να προκαταβάλει ένα ειδησεογραφικό μέσο και αυτές είναι, η αποφυγή προβολής γεγονότων με υπερβολές ή ακόμα και κατασκευασμένες ιστορίες που στόχο έχουν την έξαψη της περιέργειας για την προσέλκυση του αναγνωστικού κοινού (κίτρινος τύπος). Πιο σοβαρό πλήγμα μπορεί να αποφέρουν τα ειδησεογραφικά από την στιγμή που έχουν την δύναμη να καταστείλουν την πληροφορία για πολιτικές σκοπιμότητες. Δεν είναι καινούργιο φαινόμενο η ανάμιξη της πολιτικής δραστηριότητας με την χρήση του λόγου στην δημόσια συζήτηση κάτι που μπορεί να επηρεάζει τον τύπο. Αυτό μπορεί να γίνει με πολλούς τρόπους από μια κυβέρνηση για παράδειγμα όπως μας ενημερώνει η Κλειώ Κεντερελίδου :

³ https://www.greek-language.gr/greekLang/modern_greek/tools/lexica/triantafyllides/search.html?lq=παραπληροφόρηση&dq

⁴ <https://www.slideshare.net/EdelmanInsights/2017-edelman-trust-barometer-global-results-71035413>

[Οι κυβερνητικές ανακοινώσεις και τα δελτία τύπου συντάσσονται με τρόπο που εξυπηρετούν τις μιντιακές επιταγές και το δημοσιογραφικό λόγο ενός δελτίου ειδήσεων, αλλά και επιδοτούν, στην ουσία, με θέματα τα Μέσα Επικοινωνίας. Αποτέλεσμα αυτού είναι η κυβέρνηση να μπορεί να ελέγχει, στο μέτρο του εφικτού, το περιεχόμενο των ειδήσεων που ανακοινώνει και το ότι οι ειδήσεις θα μεταδίδονται ως έχουν, χωρίς την παρέμβαση του δημοσιογράφου, του αρχισυντάκτη και του διευθυντή ειδήσεων του τηλεοπτικού σταθμού.]

(Κεντερελίδου: 2009, παρ 3.3)

Μέσω των κυβερνητικών ανακοινώσεων και των συνεντεύξεων τύπου με τον κυβερνητικό εκπρόσωπο μια κυβέρνηση μπορεί να διαχειριστεί τα Μέσα Επικοινωνίας. Οι πολιτικοί μαζί με τον κυβερνητικό εκπρόσωπο κάνουν δηλώσεις οι οποίες μεταφέρονται στα ειδησεογραφικά μέσα. Αυτό δημιουργεί την ευκαιρία για την κυβέρνηση να θέσει την δημόσια συζήτηση καθώς και την ενημέρωση στους πολίτες που παρακολουθούν τα Μέσα Επικοινωνίας. Δηλαδή τα ειδησεογραφικά όταν εφαρμόζουν τέτοιες πρακτικές σε συνδυασμό με την έλλειψη επαγγελματισμού τείνουν να ετεροκαθορίζονται από διάφορες πολιτικές σκοπιμότητες.

Από την άλλη παρόμοια φαινόμενα έχουν εκκολληθεί μέσω της πλατφόρμας κοινωνικών δικτύων στο διαδίκτυο. Η ροή περιεχομένου στις πλατφόρμες αυτές δημιουργούνται από τους ίδιους τους χρήστες της, δίχως διαμεσολάβηση. Έτσι με τρόπο ανώνυμο θα μπορούσε οποιοσδήποτε κακόβουλος φορέας να διαχύσει στοχευμένες ψευδείς ειδήσεις με τέτοιο τρόπο ώστε να επωφελή τον εκάστοτε κακόβουλο φορέα. Όλοι όσοι χρησιμοποιούν το Facebook, το Twitter κ.α. είναι πολύ πιθανό να έχουν πέσει πάνω σε τέτοια παραπλανητικά μηνύματα. Οι κακόβουλοι φορείς μπορούν με την χρήση μεγάλου ανθρωπίνου δυναμικού που έχει πρόσβαση σε αυτές τις πλατφόρμες, να διοχετεύσουν μηνύματα μέσα σε διάφορες υπάρχουσες θεματολογίες που είναι υπό συζήτηση με σκοπό να πολώσουν το πεδίο. Ιδίως με την χρήση των bots⁵ αυτή η διαδικασία μπορεί να γίνει πολύ πιο φτηνή και αποδοτική με αποτέλεσμα όλο και περισσότεροι κακόβουλοι φορείς να χρησιμοποιούν τέτοιου είδους τακτικές. Το περιεχόμενο τέτοιων παραπλανητικών μηνυμάτων συνήθως λειτουργεί ως δόλωμα για το αναγνωστικό κοινό με την χρήση ψευδών οπτικών στοιχείων ή στατιστικών, η χρήση ψευδούς πηγής που επινοείται λαμβάνει χώρα ώστε να προσθέτει εγκυρότητα στο περιεχόμενο. (Ireton C. , Posetti J., 2021)

Τέλος, είναι σαφές ότι η παραπληροφόρηση τόσο στα κοινωνικά δίκτυα όσο και στα ΜΜΕ έχουν επιπτώσεις στην υγεία και στην διατήρηση μιας δημοκρατικής κοινωνίας. Είναι στο χέρι των πολιτών με την αρωγή ερευνητών της δημοσιογραφίας, της επιστήμης και της σωστής παιδείας, να αναγνωρίζουν τον κακό σπόρο και να κρατάνε τον καλό.

⁵ Σ.τ.Σ. με την λέξη bots, αναφέρεται σε Τεχνητές Νοημοσύνες που χρησιμοποιούν πολλούς ψεύτικους λογαριασμούς στις πλατφόρμες κοινωνικών δικτύων και μέσω της μαζικής αυτόματης δημιουργίας κειμένων πραγματοποιούν αναρτήσεις σε θεματολογίες που έχουν προγραμματιστεί να προσβάλλουν.

1. Συλλογή και Κατανόηση των Δεδομένων

Σε αυτό το κεφάλαιο θα εξηγήσουμε όλη την βασική λειτουργία του προγράμματος που επιτρέπει την εξαγωγή αποτελεσμάτων. Το βασικό πρόβλημα είναι η εύρεση συσχετίσεων μεταξύ άρθρων των ειδησεογραφικών και τάσεων του twitter. Εδώ με τη λέξη “συσχέτιση” εννοείτε η μαθηματική απόσταση μεταξύ δύο ή περισσότερων εγγράφων. Για την επίτευξη αυτού του σκοπού έπρεπε να λυθούν ορισμένα προβλήματα. Για να γίνει οποιαδήποτε συσχέτιση μεταξύ εγγράφων, πρέπει να υπάρχουν τα έγγραφα. Σε αυτό το πρόβλημα χρειάζονται άρθρα και tweets. Θα αναλυθεί πως ακριβώς εξάγονται τα άρθρα κατευθείαν από τα ειδησεογραφικά, πως με τη βοήθεια του twitter API εξάγονται τα tweets καθώς και άλλες πληροφορίες που προσφέρει η πλατφόρμα στους developers. Στην συνέχεια ακολουθεί ο “καθαρισμός” των δεδομένων και τέλος ο αλγόριθμος της μαθηματικής συσχέτισης των εγγράφων.

1.1 Twitter

Όπως θα έχει γίνει αντιληπτό μέχρι τώρα, η πλατφόρμα κοινωνικής δικτύωσης που χρησιμοποιήθηκε είναι αποκλειστικά το Twitter. Αυτή η απόφαση βασίστηκε σε δύο λόγους. Αρχικά το Twitter παρέχει το δικό του API, πράγμα πολύ βολικό και εύχρηστο όπως αναλύεται παρακάτω. Ο δεύτερος λόγος, είναι πιο πολύ στον τρόπο που χρησιμοποιείται η πλατφόρμα από τους χρήστες και πως οι πληροφορίες που διαχέονται είναι πολύ πιο χρήσιμες σε σχέση με άλλες πλατφόρμες.

Από την εμφάνιση του Web 2.0 οι χρήστες του διαδικτύου από απλοί παρατηρητές, πλέον έχουν την δυνατότητα, χωρίς κάποια ιδιαίτερη τεχνογνωσία να παράγουν περιεχόμενο, να το μοιράσουν και να μείνουν συνδεδεμένοι σε ένα αλληλεπιδραστικό περιβάλλον. Συγκεκριμένα η πλατφόρμα Twitter στεγάζει εκατομμύρια χρήστες παγκοσμίως καθώς και χιλιάδες Έλληνες χρήστες. Συγκεκριμένα για την Ελλάδα, στο Twitter υπάρχει και στεγάζεται η ελληνική κοινότητα. Έτσι η κοινότητα όπως τη γνωρίζουμε στο φυσικό χώρο μεταφέρθηκε με τα δικά της χαρακτηριστικά στο διαδίκτυο. Διαπερνώντας με αυτό το τρόπο τους γεωγραφικούς περιορισμούς όσον αφορά την επικοινωνία. Επιτρέποντας σε πολίτες όλου του ελλαδικού χώρου και όχι μόνο να συμμετέχουν σε κοινές συζητήσεις ανεξάρτητα του τόπου.

Στο Twitter υπάρχει η έννοια του microblogging, η διαφορά του με το blogging είναι ότι είναι πολύ πιο σύντομο και περιεκτικό. Το περιεχόμενο συνήθως αφορά την ενημέρωση των χρηστών εντός του συγκεκριμένου κοινωνικού δικτύου, για τις δραστηριότητες, απόψεις και καταστάσεις σχετικά με την ζωή τους. Ο περιορισμός στους 280 χαρακτήρες ανά δημοσίευση (το tweet) ωθεί τους χρήστες να χρησιμοποιούν την πλατφόρμα αρκετά συχνά, διότι η ανάρτηση απαιτεί ελάχιστο χρόνο να συγγραφθεί. Το ίδιο το Twitter ρωτάει τους χρήστες του «Τι συμβαίνει;» και εκείνοι απαντούν αναλόγως, σχετικά για την επικαιρότητα και το ζήτημα που απασχολεί τους περισσότερους, για ένα συμβάν που υπήρξε στον τόπο τους, τί σκέφτονται και τί κάνουν. Δημιουργώντας έτσι ο καθένας ξεχωριστά, αλλά και αλληλοεπιδρώντας με τους υπόλοιπους στο κοινωνικό δίκτυο, μια πλατφόρμα που δημοσιεύονται και σχολιάζονται ειδήσεις. Καθώς και ο διαμοιρασμός πληροφοριών μεταξύ των χρηστών ανάλογα με τα ενδιαφέροντα τους, αλλά και γενικότερα τη συζήτηση κάποιου θέματος από χιλιάδες χρήστες ταυτόχρονα.

Όλα τα παραπάνω συντελούν το Twitter, μια πλατφόρμα κοινωνικού δικτύου κατάλληλη για το σκοπό μας. Ενώ άλλες πλατφόρμες είναι πολύ πιο δημοφιλείς με περισσότερους χρήστες, δεν παρέχουν τα ίδια εργαλεία στον χρήστη, ώστε να τους ωθούν να γράφουν περιεχόμενο που αφορά τα γεγονότα της επικαιρότητας. Ένα τέτοιο εργαλείο – καινοτομία του Twitter είναι οι τάσεις (hashtags).

1.2 Οι τάσεις στο Twitter

Οι τάσεις στο Twitter είναι λέξεις οι οποίες γράφονται από πολλούς ξεχωριστούς χρήστες στα tweets που δημοσιεύουν. Συνήθως χρησιμοποιείτε το σύμβολο «#» (hashtag) στην αρχή της λέξης για να δημιουργηθεί μια θεματολογία. Όταν και αν γραφτούν πολλά tweets που εμπεριέχουν την ίδια λέξη κλειδί, τότε αυτή η λέξη γίνεται τάση. Ανάλογα το πόσα πολλά tweets και σε πόσο χρόνο δημοσιευθούν θα οριστεί και η κατάταξη της λέξης κλειδί. Δηλαδή θα οριστεί πόσο δημοφιλής είναι αυτή η λέξη άρα και η θεματολογία που διέπει όλα τα tweets που έχουν γραφτεί για αυτήν. Ανά δεδομένη στιγμή οι κορυφαίες τάσεις δημοσιεύονται στην πλατφόρμα και είναι διαθέσιμες για προβολή στην αρχική σελίδα της εφαρμογής.

Αυτό τα τελευταία χρόνια έχει αλλάξει συθέμελα τον τρόπο διάχυσης της πληροφορίας στην κοινωνία μας. Στο σύνολο των χιλιάδων εκατομμυρίων tweets αποκαλύπτονται παγκόσμια ενδιαφέροντα, γεγονότα, τάσεις και όλα αυτά σε σχεδόν πραγματικό χρόνο. Λόγω του ταχύτατου ρυθμού δημοσιευμάτων στην πλατφόρμα, οι τάσεις όπως συμβαίνει και στον πραγματικό κόσμο ανανεώνονται συνεχώς. Αυτό συνεπάγεται την πλατφόρμα Twitter σαν μια πολύτιμη πηγή εξερεύνησης της μαζικής προσοχής των χρηστών σε κάποιο θέμα. Οι κορυφαίες τάσεις του Twitter σε αυτήν της εργασία συλλέγονται περίπου ανά ώρα. Να σημειωθεί ότι το Twitter API επιτρέπει την συλλογή των πρώτων πενήντα τάσεων ανά δεδομένη στιγμή. Για λόγους υπολογιστικής οικονομίας αποφασίστηκε να περιοριστεί στις πρώτες πέντε.

Οι τάσεις όπως ειπώθηκε πιο πάνω συνδέονται με tweets (δημοσιεύσεις χρηστών), όλα αυτά τα tweets νοηματοδοτούν την τάση. Ένα παράδειγμα εξαγόμενο από τις παρατηρήσεις του εγχειρήματος είναι η τάση «#σεισμος», πρωτοεμφανίστηκε πρώτο στις κορυφαίες τάσεις την ημερομηνία 25/10/21 16:20 μέχρι 25/10/21 20:50 όπου και εξαφανίστηκε. Δηλαδή για ένα διάστημα 5 ωρών υπήρχε στον πίνακα με τις κορυφαίες τάσεις. Οι χρήστες γράφουν για τον σεισμό που συνέβη στην Αθήνα μικρής εντάσεως 2.8 ρίχτερ. Ο τίτλος της τάσης είναι γενικός και μπορεί να ξαναεμφανιστεί με άλλη ημερομηνία και να μιλάει πιθανώς για κάποιον άλλο σεισμό. Όπως συνέβη με την τάση «#κρουσματα», στις 26/10/21 οι χρήστες μιλάνε για τα κρούσματα κορονοϊού στην Ελλάδα που έχουν ξεπεράσει τα 4000. Ενώ δύο εβδομάδες αργότερα ξαναεμφανίζεται η ίδια τάση μόνο που τώρα μιλάνε για 8000 κρούσματα. Το περιεχόμενο των χρηστών έχει αλλάξει από την μία ημερομηνία στην άλλη. Άλλο τρανταχτό παράδειγμα είναι η τάση «#γυναικοκτονια», η ίδια τάση φαίνεται να εμφανίζεται στις 29/10/21 και αφορά γυναικοκτονία της εν διαστάσει συζύγου από τον άνδρα της σε περιοχή της Κρήτης. (βλ. https://www.efsyn.gr/efkriti/koinonia/317072_i-48hroni-nektaria-dekato-trito-fetino-thyma-gynaikoktonias). Ενώ ένα μήνα μετά η τάση εμφανίζεται ακριβώς με τον ίδιο τίτλο μόνο που αυτή την φορά υπάρχει άλλο θύμα γυναικοκτονίας σε περιοχή της Θεσσαλονίκης (βλ. <https://www.iefimerida.gr/ellada/nea-gynaikoktonia-i-14i-mesa-sto-2021-thessaloniki>). Είναι αναμενόμενο τα σχόλια και οι παρατηρήσεις των χρηστών πάνω σε ένα θέμα να ποικίλουν. Πολλές φορές δημιουργούνται αντικρουόμενες ομάδες μέσα στην ίδια θεματολογία (tweets που μοιράζονται την ίδια τάση-hashtag). Ενώ υπάρχουν και φορές που η τάση σαν τίτλος δεν είναι σαφής και μέσα σε αυτή εμπεριέχονται tweets ασυσχέτιστα μεταξύ τους.

Οι τάσεις κανονικά διαμορφώνονται με βάση τις προτιμήσεις του χρήστη. Δηλαδή βασίζονται πάνω στα άτομα που ακολουθεί, τα ενδιαφέροντα του και την τοποθεσία του (βλ. <https://help.twitter.com/en/using-twitter/twitter-trending-faqs>). Φυσικά το εγχείρημα θα πρέπει να λειτουργεί αμερόληπτα και να παρέχει όσο το δυνατόν αφιltrάριστες τάσεις. Αυτές είναι οι τάσεις συνολικά της ελληνικής κοινότητας του Twitter, ευτυχώς το Twitter API παρέχει τέτοια λειτουργία, όπου και χρησιμοποιήθηκε σε αυτήν την εργασία.

1.3 Twitter API

Για την εξαγωγή αναγκαίων δεδομένων από το Twitter χρησιμοποιήθηκε η επίσημη προγραμματιστική διεπαφή της εφαρμογής που παρέχει η ίδια εταιρεία. Αυτή ορίζεται σύμφωνα με τα λόγια τους:

Το Twitter API επιτρέπει την προγραμματιστική πρόσβαση στο Twitter με ξεχωριστούς και προηγμένους τρόπους. Χρησιμοποιείτε μοναδικά στοιχεία του Twitter όπως: Tweets, Απευθείας μηνύματα, Spaces, Λίστες, χρήστες και άλλα.⁶

Δηλαδή σχεδόν ότι μπορεί να γίνει στην πλατφόρμα μέσω φυλλομετρητή ή εφαρμογής στο smartphone, γίνεται και μέσω του Twitter API. Αυτό δίνει τη δυνατότητα στους προγραμματιστές να αναλύσουν τα δεδομένα, να αυτοματοποιήσουν κάποιες διαδικασίες, να βελτιώσουν την εμπειρία των χρηστών της εφαρμογής καθώς και άλλα πολλά.

Για τους σκοπούς αυτής της εργασίας χρησιμοποιήθηκε η βιβλιοθήκη Tweepy. Αυτή είναι ένα σύνολο εργαλείων, εύκολα διαχειρίσιμων για την πρόσβαση του Twitter API μέσω της γλώσσας προγραμματισμού Python. Η επίσημη έκδοση API του Twitter χρησιμοποιεί REST endpoint (Representational state transfer) για την λήψη και αποστολή δεδομένων, αυτό είναι ουσιαστικά η αποστολή HTTP requests. Εφόσον η γλώσσα που επιλέχθηκε είναι η Python για αυτήν την υλοποίηση, φυσικό ακόλουθο ήταν να χρησιμοποιηθεί η βιβλιοθήκη Tweepy, αξιοποιώντας πλήρως όλα τα πλεονεκτήματα. Όπως η ελεγχόμενη διατήρηση σύνδεσης με το API, η αναμονή σε περίπτωση που ξεπεραστεί το όριο χρήσης, ο πιο ευανάγνωστος κώδικας, πιο γρήγορα αποτελέσματα με τη χρήση Threading κ.α. .

Για την πρόσβαση στο API του Twitter, είναι υποχρεωτικός ένας λογαριασμός Developer. Αυτός αποκτιέται μετά από αίτηση και έγκριση από την πλατφόρμα. Υπάρχουν διάφορα πλάνα παροχών καθώς και πλεονεκτήματα. Κάποια είναι δωρεάν για ερασιτεχνικούς ή ακαδημαϊκούς σκοπούς και άλλα είναι επί πληρωμή. Τα πλάνα είναι κλιμακωτά και όσο ανεβαίνουν τόσο περισσότερες ενέργειες μπορούν να πραγματοποιηθούν. Όπως για παράδειγμα το πλήθος εξαγωγής tweet, τα φίλτρα στα requests, αριθμός ενεργών έργων, καθώς και πρόσβαση σε κλειδωμένα δεδομένα. Σε αυτό εδώ το εγχείρημα χρησιμοποιήθηκε το δεύτερο πιο απλό πλάνο Developer λογαριασμού, το Elevated. Δεν θα αναλυθεί τι προσφέρει λεπτομερώς, αλλά αξίζει να σημειωθεί ότι παρέχει 180 requests ανά δεκαπέντε λεπτά. Πράγμα που αντιστοιχεί πρόσβαση σε περίπου 10.000 tweets την ώρα.

Το Twitter API επιτρέπει στους ερευνητές να συγκεντρώσουν πληροφορίες χρησιμοποιώντας δομημένα ερωτήματα (structured queries βλ. <https://www.tweepy.org>). Σαν μέσο για την πρόσβαση πληροφοριών που αφορούν συγκεκριμένα θέματα ή χαρακτηριστικά χρηστών του Twitter. Σε αυτήν την εργασία τα ερωτήματα κατασκευάστηκαν για να συλλέγουν tweets που εμπεριέχουν κορυφαίες τάσεις κάθε εκάστοτε χρονική στιγμή.

1.4 Scraping Ειδησεογραφικών

Ο όρος scraping (web scraping) αναφέρεται στην μέθοδο απόκτησης και συλλογής δεδομένων από το διαδίκτυο. Η συλλογή δεδομένων μπορεί να γίνει από κάποια ομάδα ατόμων χειροκίνητα. Στην περίπτωση μας βέβαια είναι υποχρεωτικό να κατεβαίνουν τα άρθρα από ειδησεογραφικά στην Ελλάδα μαζικά, μηχανικά και γρήγορα. Αυτό μας οδηγεί από μόνο του, στην αυτοματοποίηση αυτής της διαδικασίας. Δηλαδή στην απόκτηση άρθρων από ειδησεογραφικά μαζί με όλα τα μεταδεδομένα τους. Τα ειδησεογραφικά που επιλέχθηκαν για web scraping είναι το Σκάϊ, Πρώτο Θέμα, Καθημερινή, In και Enikos. Η επιλογή έγινε τυχαία μεταξύ των διασημότερων ειδησεογραφικών στην Ελλάδα. Από αυτά τα ειδησεογραφικά λοιπόν είναι χρήσιμο να αποθηκεύεται μαζί με το κείμενο του άρθρου και η ημερομηνία δημοσίευσης, ο τίτλος καθώς και η κατηγορία. Το κείμενο του άρθρου χρειάζεται ώστε να υπάρχει η πληροφορία για να πραγματοποιηθεί η μαθηματική συσχέτιση με τα tweets, όπως θα περιγράψω παρακάτω. Η ημερομηνία είναι πολύ χρήσιμη για να συγκριθεί με την ημερομηνία της πρωτοεμφάνισης της αντιστοιχούμενης τάσης. Ούτως ώστε να βρεθεί αν το άρθρο προηγήθηκε της τάσης ή το αντίστροφο.

Για την επίτευξη αυτού του στόχου χρησιμοποιήθηκε η προγραμματιστική βιβλιοθήκη σε Python «Beautiful Soup». Το συγκεκριμένο εργαλείο έχει κατασκευαστεί για να εξαγάγει δεδομένα μέσα από HTML και XML δομές. Η

⁶ ΜτΣ από την σελίδα του Twitter Developer Platform

HTML (HyperText Markup Language) είναι η ακρογωνιαίος λίθος του διαδικτύου. Είναι markup γλώσσα προγραμματισμού, πράγμα που σημαίνει ότι χρησιμοποιεί προκαθορισμένες ετικέτες (tags) για να χτίσει δομή στα στοιχεία της σελίδας. Από την άλλη η XML είναι παρόμοια με την HTML, με την διαφορά ότι δεν χρησιμοποιεί προκαθορισμένες ετικέτες. Παρέχει την ελευθερία στον προγραμματιστή να ορίσει τις δικές του ετικέτες ανάλογα τις ανάγκες του. Χρησιμοποιείτε ευρέως στο διαδίκτυο και όχι μόνο, για να αποθηκεύσει δεδομένα με συγκεκριμένη δομή, να αναζητήσει δεδομένα και γενικότερα για τον διαμοιρασμό των δεδομένων λόγω της τυποποιημένης και ευρέως αποδεκτής σύνταξης.

Όλες οι σελίδες των ειδησεογραφικών είναι γραμμένες στη βάση τους με HTML. Υπάρχουν περιπτώσεις που η ιστοσελίδα μπορεί να έχει κάποιο λάθος στην σύνταξη της (broken HTML). Λόγω όμως της επιείκειας που δείχνουν οι φυλλομετρητές σε συντακτικά λανθασμένο κώδικα, ο αναγνώστης μπορεί να μην αντιληφθεί κάποιο σφάλμα στην σελίδα. Ωστόσο όταν γίνει προγραμματιστική ανάλυση της markup δομής για να εξαχθεί κάποια πληροφορία, μπορεί να προκύψει κάποιο σφάλμα και να αποτύχει η προσπάθεια. Εκεί ακριβώς μπαίνουν στο παιχνίδι οι parsers για να διορθώσουν τα λάθη. Στη προκειμένη περίπτωση χρησιμοποιήθηκε η lxml, παρακάτω φαίνεται ένα σκοπίμως συντακτικό λάθος στην σύνταξη της HTML. Το οποίο οι περισσότεροι σύγχρονοι φυλλομετρητές θα διόρθωναν αυτόματα και θα εμφάνιζαν σωστά το περιεχόμενο. Η lxml parser διορθώνει συντακτικά λάθη στη σύνταξη της HTML αλλά και της XML. Πέρα από αυτό αναδιαμορφώνει τον κώδικα, έτσι ώστε κάθε φορά που εισάγεται από διαφορετική πηγή HTML αρχείο να εξάγεται όσο είναι δυνατόν η ίδια δομή.

```
<html>
<!--Συντακτικά λάθος HTML-->
<!--Το σωστό θα ήταν να υπήρχε-->
<!--closing tag </p>-->

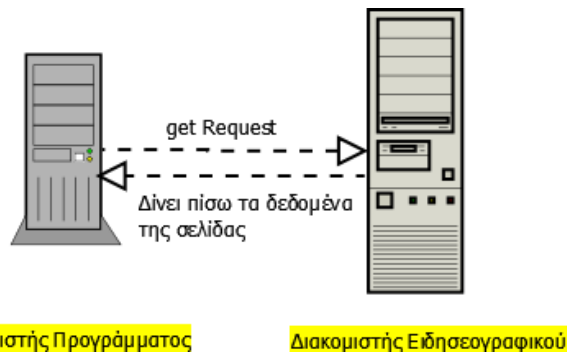
<p>Hello World!

</html>
```

```
<html>
<body>
<!--Πολύ σωστά συμπληρώθηκαν-->
<!--το closing tag καθώς-->
<!--και άλλα στοιχεία-->
<p>Hello World!</p>
</body>
</html>
```

Εικόνα 1 Αριστερά, λανθασμένη HTML, δεξιά διορθωμένη αυτόματα χρησιμοποιώντας lxml parser

Ο συνδυασμός του lxml parser με τη διόρθωση του πηγαίου κώδικα και της βιβλιοθήκης BeautifulSoup με την αξιοποίηση των μεθόδων εξαγωγής περιεχομένου επιτρέπουν την συλλογή των πληροφοριών από τα ειδησεογραφικά. Αρχικά το πρόγραμμα κατεβάζει το newsfeed, κάθε ειδησεογραφικό έχει αρχική σελίδα όπου ό,τι άρθρο και αν δημοσιευθεί ανεξαρτήτως κατηγορίας, φαίνεται εκεί πέρα. Η αρχική σελίδα είναι γραμμένη σε HTML και άρα είναι δυνατόν να γίνει scraping. Αν το ειδησεογραφικό έχει ενεργοποιημένη την υπηρεσία RSS, επιλέγεται αυτή. Το RSS Feed είναι λειτουργία μιας σελίδας που επιτρέπει στους χρήστες και τις εφαρμογές να έχουν πρόσβαση στα δεδομένα μια σελίδας με τυποποιημένη φιλική για τον υπολογιστή γλώσσα, συνήθως είναι XML. Πρώτα από όλα γίνεται ένα απλό GET request στην αρχική σελίδα του ειδησεογραφικού. Το GET request είναι μέθοδος που προέρχεται από τον HTTP, να σημειωθεί ότι το HTTP (Hypertext Transfer Protocol) είναι ο τρόπος που επικοινωνεί επί το πλείστον όλο το διαδίκτυο σήμερα.



Εικόνα 2 Επικοινωνία μέσω GET αιτήματος από τον διακομιστή του προγράμματος στον διακομιστή του ειδησεογραφικού.

Στη συνέχεια επιστρέφεται στον διακομιστή του προγράμματος η πληροφορία της σελίδας, αυτή η επιστρεφόμενη πληροφορία είναι κυριολεκτικά όλη η σελίδα σε μορφή HTML ή XML αν έχουν ενεργοποιημένη την υπηρεσία RSS Feed. Για κάθε άρθρο πραγματοποιείτε επιπλέον GET request στο link του, ούτως ώστε να συγκεντρωθεί το κείμενο. Έπειτα μετατρέπεται αυτό το αρχείο με lxml μέσω της BeautifulSoup βιβλιοθήκης και από εκεί και πέρα γίνεται η αναζήτηση στις επιθυμητές πληροφορίες που πρέπει να εξαχθούν.

Αυτή η διαδικασία ενώ στο συγκεκριμένο εγχείρημα περιορίζεται σε πέντε ειδησεογραφικά, μπορεί να επεκταθεί σε απεριόριστα. Ο λόγος που περιορίστηκε σε πέντε είναι λόγω οικονομίας στους υπολογιστικούς πόρους. Όσο περισσότερα τα ειδησεογραφικά τόσο περισσότερα άρθρα κατεβαίνουν, επεξεργάζονται και αποθηκεύονται. Για παράδειγμα με αυτά τα πέντε ειδησεογραφικά που επιλέχθηκαν, κατεβαίνουν περίπου 260 άρθρα ανά ειδησεογραφικό μέσα σε διάστημα 24 ωρών.

Να σημειωθεί ότι υπάρχουν ιστοσελίδες που εξαρτώνται από την JavaScript για να παράγουν την πληροφορία τους. Η JavaScript είναι επίσης βασικό δομικό στοιχείο του διαδικτύου, είναι η γλώσσα προγραμματισμού κυρίως της HTML, παρέχει επιπλέον λειτουργίες μιας ιστοσελίδας. Στην περίπτωση που αποστέλλοταν με τον παραπάνω τρόπο ένα GET request, τότε η πληροφορία που θα δεχόταν ο διακομιστής προγράμματος θα ήταν δυσνόητος JavaScript κώδικας και όχι οι πληροφορίες που θα θέλαμε. Αυτό συμβαίνει επειδή στον φυλλομετρητή εκτελείται ο JavaScript κώδικας και παράγεται η HTML όπου έχει όλη την πληροφορία. Δεν συναντήθηκε αυτό το πρόβλημα με κανένα ειδησεογραφικό, ωστόσο αν υπήρχε τέτοιο ειδησεογραφικό η λύση θα ήταν να χρησιμοποιηθεί η βιβλιοθήκη Selenium⁷. Η τελευταία είναι μια ομπρέλα από διάφορα εργαλεία και βιβλιοθήκες, προσανατολισμένα να ενεργοποιούν και να υποστηρίζουν την αυτοματοποίηση των φυλλομετρητών. Ουσιαστικά δημιουργεί μια προσομοίωση φυλλομετρητή, όπου εκτελούνται όλα τα επιμέρους κομμάτια της σελίδας με αποτέλεσμα να είναι δυνατή η εξαγωγή της επιθυμητής HTML πληροφορίας.

⁷ Περισσότερες πληροφορίες για την βιβλιοθήκη Selenium εδώ: <https://www.selenium.dev/>

2. Βάση Δεδομένων

Οι σχεσιακές βάσεις δεδομένων χρησιμοποιούνται εδώ και πολλά χρόνια ευρέως στις περισσότερες από τις εφαρμογές παγκοσμίως και έχουν καλή απόδοση όταν **χειρίζονται** περιορισμένο αριθμό δεδομένων. Στο παρόν εγχείρημα με την πάροδο του χρόνου συνεχώς αποθηκεύονται και **διαβάζονται** πολλά δεδομένα, πράγμα που το κάνει να γίνεται όλο και πιο απαιτητικό σε πόρους όσο μεγαλώνει. Για να χειριστούν έναν τεράστιο όγκο δεδομένων όπως εκατοντάδες χιλιάδες άρθρα ειδησεογραφικών και tweets, οι παραδοσιακές σχεσιακές βάσεις δεδομένων είναι αναποτελεσματικές. Για να ξεπεραστεί αυτό το πρόβλημα χρησιμοποιήθηκε NO-SQL.

Ένα επιστημονικό άρθρο έχει δείξει ότι οι NO-SQL υλοποιήσεις είναι έως και 10 φορές πιο γρήγορες στο να διαβάζουν τα δεδομένα από ότι στις παραδοσιακές SQL (TABLE II Yishan & Sathiamoorthy, 2013). Για την ακρίβεια σε αυτό το άρθρο βρέθηκε ότι η καλύτερη βάση δεδομένων για εγγραφή, ανάγνωση, διαγραφή είναι η MongoDB και η Couchbase. Με μια επιφύλαξη, διότι το ίδιο αναφέρει ότι δεν δοκιμάστηκαν για πολύπλοκα queries -αναζητήσεις στα έγγραφα μέσα από την βάση δεδομένων. Ωστόσο στην παρούσα εργασία χρησιμοποιούνται άλλες μέθοδοι για πολύπλοκα queries, πέρα από την φυσική γλώσσα της ΒΔ όπως θα δούμε και στην συνέχεια.

Η βάση δεδομένων σε αυτό το εγχείρημα έχει υλοποιηθεί σε *MongoDB v5.0.3*. Η MongoDB είναι *NoSQL* ή αλλιώς *no-relational* βάση δεδομένων. Σε αντίθεση με την κλασσική πινακοειδή SQL, η MongoDB χρησιμοποιεί BSON τύπου δεδομένα (εμπνευσμένο από την JSON) με συμπληρωματική τη χρήση schemas για τη δήλωση τύπου δεδομένων ή ακόμα και περιορισμών. Σε αντιστοιχία SQL και MongoDB, ο πίνακας ονομάζεται *collection* και μια εγγραφή ονομάζεται *document*.

id	name	age
1	Joe	31
2	Mary	27

```
[  
  {  
    "id": 1,  
    "name": "Joe",  
    "age": 31  
  },  
  {  
    "id": 2,  
    "name": "Mary",  
    "age": 27  
  }  
]
```

1. Μορφή αποθήκευσης δεδομένων σε SQL

2. Μορφή αποθήκευσης δεδομένων σε NO-SQL, mongoDB

Η MongoDB παρέχει ευελιξία κατά τη διάρκεια ανάπτυξης της εφαρμογής. Έχει ενσωματωμένη υποστήριξη για οριζόντια επεκτασιμότητα. Είναι εύκολο να αντιγραφεί η ΒΔ από τον ένα διακομιστή στον άλλο χρησιμοποιώντας επίσημα εργαλεία εξαγωγής - εισαγωγής. Είναι δυνατόν να αποθηκευτούν σύνθετα δεδομένα σε ένα πεδίο όπως για παράδειγμα έναν πίνακα, ένα αντικείμενο (object). Υποστηρίζει διάφορους native τύπους δεδομένων από ποικίλες γλώσσες προγραμματισμού και δεν χρειάζεται κανένα είδος μετατροπής. Παράδειγμα είναι η JavaScript με object τύπου δεδομένων και η python με dictionary τύπου δεδομένων.

2.1 Schema των μοντέλων

Το schema εδώ μεταφράζεται σαν προσχέδιο και το μοντέλο είναι το πρότυπο ενός collection. Δηλαδή το προσχέδιο των μοντέλων περιέχει όλους τους κανόνες και τους τύπους των δεδομένων που επιτρέπεται να εισαχθούν. Όλες οι ιδιότητες που μπορούν να εισαχθούν είναι:

- αντικείμενο (object)
- πίνακας
- αλφαριθμητικό (string)
- αριθμός (μπορεί να είναι όλο το σύνολο των πραγματικών αριθμών)
- Boolean
- UUID (Universal Unique Identifier, Παγκόσμιο Μοναδικό Αναγνωριστικό)
- ObjectId (μοναδικό αναγνωριστικό που δημιουργείται αυτόματα μέσα στην βάση δεδομένων)
- Set (το σετ είναι το ίδιο με ένα πίνακα μόνο που δεν υπάρχουν διπλότυπα)
- Dictionary (είναι μια λίστα με ζεύγη που χωρίζονται στο κλειδί και την τιμή, το κλειδί είναι μοναδικό μέσα στο dictionary)

Αυτές οι προδιαγραφές ορίζονται ταυτόχρονα με την δημιουργία του collection. Σε αντίθεση με την SQL λογική, εδώ μπορούν να αλλάξουν τα schemas ανά πάσα στιγμή, να αφαιρεθούν στήλες ακόμα και να προστεθούν καινούργιες. Τα schemas στην mongoDB είναι ευέλικτα και δεν είναι υποχρεωτικά. Δεν υπάρχουν συσχετίσεις μεταξύ άλλων collection, όπως υπάρχουν στα παραδοσιακά SQL. Ωστόσο μπορεί να φτιαχτούν από τον προγραμματιστή αν το επιθυμεί. Ένας τρόπος είναι με την χρήση κλειδιών ανάμεσα σε δύο collections, και έπειτα με την χρήση των σωστών ερωτημάτων (queries). Άλλος τρόπος είναι με embedded δεδομένα, δηλαδή με την ενθυλάκωση δεδομένων μέσα σε ένα πεδίο του collection. Παράδειγμα το έγγραφο:

```
{
  _id: <id>,
  name: "Yamaha Moto",
  engine: {
    type: 'liquid cooled 4-stroke',
    horsepower: 84.6,
    max_rpm: 10000
  },
  fuel: {
    capacity: 18,
    supply: "Fuel Injection"
  }
}
```

Εδώ τα πεδία του «engine» και του «fuel», αν δεν έχουν οριστεί, αυτόματα θεωρούνται τύπου Object (αντικείμενο).

Μέσα σε ένα collection κάθε εγγραφή μπορεί να έχει τελείως διαφορετική δομή σε σχέση με τις άλλες. Για παράδειγμα το collection που έχει δύο έγγραφα:

```
{
  _id: <id>,
  brand: "Kawasaki",
  number_of_wheels: 4
```

```

},
{
  _id: <id>,
  name: "Honda 1000cc",
  type: "Street"
}

```

Εδώ ενώ το πρώτο έγγραφο έχει τα πεδία `_id`, `brand`, `number_of_wheels` το δεύτερο έγγραφο έχει το `_id`, `name`, `type`. Αυτή η ιδιαιτερότητα επιτρέπει στον σχεδιαστή της βάσης δεδομένων `mongodb` να ορίσει αυτός όπως θέλει το περιεχόμενο των `collections`. Ακολουθεί το `schema` των `collections` όπως ορίστηκαν σε αυτό το έργο καθώς και η επεξήγηση τους.

articleDownloadDate

Όνομα Πεδίου	Τύπος	Επεξήγηση
<code>_id</code>	<code>ObjectId()</code>	Μοναδικό ID.
<code>lastTimeArticle</code>	<code>Date</code>	Η τελευταία στιγμή που κατέβηκαν και αποθηκεύτηκαν άρθρα.

Κάθε φορά που το πρόγραμμα πάει να κατεβάσει άρθρα πάει και κατεβάζει από την τελευταία ημερομηνία που είχε ξανακατεβάσει (`lastTimeArticle`). Στο τέλος όταν κατέβουν τα νέα άρθρα, σημειώνεται νέα ημερομηνία `lastTimeArticle`.

articles

Όνομα Πεδίου	Τύπος	Επεξήγηση
<code>_id</code>	<code>ObjectId()</code>	Μοναδικό ID.
<code>title</code>	<code>String</code>	Τίτλος του άρθρου.
<code>summary</code>	<code>String</code>	Περίληψη του άρθρου.
<code>date</code>	<code>Date</code>	Ημερομηνία που δημοσιεύτηκε το άρθρο.
<code>category</code>	<code>String</code>	Κατηγορία του άρθρου.
<code>link</code>	<code>String</code>	Το url του άρθρου.
<code>article</code>	<code>String</code>	Το κείμενο του άρθρου (όλο το άρθρο).
<code>source</code>	<code>String</code>	Από ποιο ειδησεογραφικό είναι.

Εδώ αποθηκεύονται τα άρθρα, όπως γίνονται parsing από τα ειδησεογραφικά.

trends

Όνομα Πεδίου	Τύπος	Επεξήγηση
_id	ObjectId()	Μοναδικό ID.
dateDownloaded	Date	Ημερομηνία που αποθηκεύτηκε το trend στη ΒΔ.
hashtag	String	Το όνομα της τάσης.
rank	Number	Κατάταξη στο πίνακα των τάσεων του twitter.
volume	Number	Πλήθος των tweets που εμπεριέχουν την τάση.

Εδώ αποθηκεύονται οι τάσεις, όπως μας τις δίνει το Twitter API. Σε κάθε χρονική στιγμή παίρνονται οι πρώτες πέντε (5).

tweets

Όνομα Πεδίου	Τύπος	Επεξήγηση
_id	ObjectId()	Μοναδικό ID.
RT	Date	Ημερομηνία που αποθηκεύτηκε το trend στη ΒΔ.
RTCount	String	Το όνομα της τάσης.
date	Number	Κατάταξη στο πίνακα των τάσεων του twitter.
dateSearched	Number	Πλήθος των tweets που εμπεριέχουν την τάση.
fullText	String	Το κείμενο του tweet
hashtags	Array	Λίστα με όλα τα hashtags (#) που συνοδεύουν το tweet
searchQuery	String	Η λέξη κλειδί αναζήτησης ή αλλιώς η <u>τάση</u> .
tweetID	String	Το μοναδικό ID του tweet
userID	String	Το μοναδικό ID του χρήστη.
userScreenName	String	Το όνομα του χρήστη.

Εδώ αποθηκεύονται τα tweets. Τα tweets που αποθηκεύονται είναι πάντα tweets που εμπεριέχουν τουλάχιστον μία τάση.

resultsLastAll & topSimilTrends

Όνομα Πεδίου	Τύπος	Επεξήγηση
_id	ObjectId()	Μοναδικό ID.
articleCategory	String	Κατηγορία του άρθρου.
articleDate	Date	Ημερομηνία που δημοσιεύτηκε το άρθρο.
articleLink	String	Το url του άρθρου.

articleOriginal	String	Το κείμενο του άρθρου (όλο το άρθρο).
articleSource	String	Από ποιο ειδησεογραφικό είναι.
articleTitle	String	Τίτλος του άρθρου.
cosSimil	Number	Το αποτέλεσμα της συνημιτονοειδής απόστασης.
searchQuery	String	Η λέξη κλειδί αναζήτησης ή αλλιώς η <u>τάση</u> .
resultCompletedDate	Date	Ημερομηνία που ολοκληρώθηκε η επεξεργασία αυτής της εγγραφής.
dateDiscovered_trend	Date	Ημερομηνία που ανακαλύφθηκε η τάση.
freshness_trend	String	Κατηγοριοποίηση της τάσης σε new, veryRecent, recent, old
rank	Number	Κατάταξη στο πίνακα των τάσεων του Twitter.
volume	Number	Πλήθος των tweets που εμπεριέχουν την τάση.
diffDate	String	Διαφορά ημερομηνίας δημοσίευσης του άρθρου με ημερομηνία που ανακαλύφθηκε η τάση. (articleDate – dateDiscovered_trend)
bestResultsDate	Date	Ημερομηνία που αποφασίστηκε να εμφανιστεί στο αποτέλεσμα.

Εδώ αποθηκεύονται τα αποτελέσματα από την επεξεργασία των δεδομένων. Στο collection resultsLastAll, κάθε φορά που το πρόγραμμα ολοκληρώνει το κύκλο του (κατέβασμα άρθρων, tweets και εύρεση συσχετισμών μεταξύ τους) αποθηκεύονται εδώ. Το topSimilTrends έχει τα 50 καλύτερα πιο πρόσφατα αποτελέσματα τα οποία τα παίρνει από το resultsLastAll.

topTweets

Όνομα Πεδίου	Τύπος	Επεξήγηση
_id	ObjectId()	Μοναδικό ID.
RT	Date	Ημερομηνία που αποθηκεύτηκε το trend στη ΒΔ.
RTCount	String	Το όνομα της τάσης.
date	Number	Κατάταξη στο πίνακα των τάσεων του twitter.
dateSearched	Number	Πλήθος των tweets που εμπεριέχουν την τάση.
fullText	String	Το κείμενο του tweet
hashtags	Array	Λίστα με όλα τα hashtags (#) που συνοδεύουν το tweet
searchQuery	String	Η λέξη κλειδί αναζήτησης ή αλλιώς η <u>τάση</u> .
tweetID	String	Το μοναδικό ID του tweet
userID	String	Το μοναδικό ID του χρήστη.

userScreenName	String	Το όνομα του χρήστη.
bestResultDate	Date	Ημερομηνία που αποφασίστηκε να εμφανιστεί στο αποτέλεσμα.

Κάθε φορά που βρίσκεται κάποιος συσχετισμός άρθρων-τάσεων, αποθηκεύονται τα πιο δημοφιλή τέσσερα (4) (με βάση τα re-tweets) tweets. Αυτό για να υπάρχει η δυνατότητα να προβάλλονται περιληπτικά σε τι αναφέρονται οι τάσεις.

sourcesScore

Όνομα Πεδίου	Τύπος	Επεξήγηση
_id	ObjectId()	Μοναδικό ID.
cosBigger35	Number	Πλήθος άρθρων που βρέθηκαν με συνημιτονοειδή απόσταση μεγαλύτερο του 35.
countArticleSource	Number	Πλήθος των άρθρων.
perc	Number	Ποσοστό $\text{cosBigger35} / \text{countArticleSource}$
source	String	Από ποιο ειδησεογραφικό είναι.

Κάθε φορά που ολοκληρώνεται ο κύκλος του προγράμματος, ανανεώνονται τα σκορ των ειδησεογραφικών. Το σκορ ορίζεται βάσει όλων των αποτελεσμάτων που βρέθηκαν με συνημιτονοειδή απόσταση μεγαλύτερη του 35. Όσα άρθρα έχουν κάποια συσχέτιση με τάση που το cosSimil είναι μεγαλύτερο του 35 τότε αυτό σημαίνει ότι υπάρχει κάποια πραγματική συσχέτιση. Αυτός ο αριθμός ορίστηκε με βάση την εμπειρική παρατήρηση, ότι είναι σωστός.

countStats

Όνομα Πεδίου	Τύπος	Επεξήγηση
_id	ObjectId()	Μοναδικό ID.
countArticles	Number	Το πλήθος των μοναδικών άρθρων
countResults	Number	Πόσες φορές έτρεξε ο αλγόριθμος επεξεργασίας δεδομένων.
countTrends	Number	Το πλήθος των μοναδικών τάσεων
countTweets	Number	Το πλήθος των μοναδικών tweet.

Εδώ αποθηκεύεται το πλήθος των documents των παραπάνω αναφερόμενων collection.

scriptStopped

Όνομα Πεδίου	Τύπος	Επεξήγηση
_id	ObjectId()	Μοναδικό ID.
lastTimeStopped	Date	Η ημερομηνία που το πρόγραμμα εμφάνισε κάποιο πρόβλημα και προσπάθησε να κάνει επαναφορά.

Κάθε φορά που προκύπτει κρίσιμο σφάλμα στην εφαρμογή και διακόπτεται η ομαλή λειτουργία του, αποθηκεύεται η ημερομηνία που προέκυψε το σφάλμα.

2.2 Οι CRUD διαδικασίες της Βάσης Δεδομένων

Ο CRUD όρος είναι ακρωνύμιο των λέξεων CREATE, READ, UPDATE, DELETE, στα ελληνικά αυτές αντίστοιχα είναι, ΔΗΜΙΟΥΡΓΙΑ, ΠΡΟΣΠΕΛΑΣΗ, ΕΝΗΜΕΡΩΣΗ, ΔΙΑΓΡΑΦΗ. Αφορούν γενικά όλες τις βάσεις δεδομένων καθώς είναι οι πιο βασικές λειτουργίες τους ανεξαρτήτως υλοποίησης. Η ΔΗΜΙΟΥΡΓΙΑ αναφέρεται στις μεθόδους εισαγωγής δεδομένων στην ΒΔ. Η ΠΡΟΣΠΕΛΑΣΗ αναφέρεται στην ανάκτηση, αναζήτηση, προβολή εγγραφών ή οποιαδήποτε άλλη πληροφορία μπορεί να προκύψει μέσα από την ΒΔ. Η ΕΝΗΜΕΡΩΣΗ είναι η αλλαγή τιμών σε προϋπάρχοντα έγγραφα μέσα στα collections και τέλος η ΔΙΑΓΡΑΦΗ αφορά είτε την διαγραφή μεμονωμένων εγγράφων ή ολόκληρων collections.

Η ΒΔ επικοινωνεί με δύο τρόπους, αρχικά μέσω python με την βιβλιοθήκη «pymongo» και μέσω του NodeJS server με την βιβλιοθήκη «mongoose»⁸. Οι κύριες λειτουργίες CRUD συμβαίνουν μέσω της python ενώ στον NodeJS server χρησιμοποιείται μόνο η λειτουργία ΠΡΟΣΠΕΛΑΣΗ. Επειδή όλη η δημιουργία των δεδομένων καθώς και η επεξεργασία τους πραγματοποιείται στην python, τα αποτελέσματα του προγράμματος παράγονται από εκεί. Ενώ η διεπαφή της εφαρμογής το μόνο που χρειάζεται είναι να προσπελάσει τα δεδομένα ώστε να τα προβάλλει με κατάλληλο τρόπο στον χρήστη. Εδώ θα συζητηθεί το πιο σημαντικό κομμάτι που αφορά το CRUD μέσω python.

Η βιβλιοθήκη «pymongo»⁹ είναι περικάλυμμα της native γλώσσας mongoDB γραμμένη σε python. Παρέχει τις ίδιες λειτουργίες, με διαφορετικό συντακτικό, φτιαγμένο για την python. Εξού και το σύνθετο όνομα py – mongo, όπου το πρώτο συνθετικό, βγαίνει από τη λέξη python και το δεύτερο συνθετικό βγαίνει από την mongoDB. Αρχικά για να γίνει σύνδεση με τη ΒΔ είναι υποχρεωτικό να λειτουργεί στο σύστημα η mongoDB service. Η τελευταία είναι η ίδια η ΒΔ mongoDB που πρέπει να είναι εγκατεστημένη και να ακούει στην σωστή θύρα (port). Στην συνέχεια πραγματοποιείται η σύνδεση pymongo και mongoDB service. Αυτό πραγματοποιείται με την εντολή MongoClient(), όπου εισάγονται δύο παράμετροι. Η πρώτη παράμετρος αφορά την διαδρομή που είναι ενεργή στο σύστημα η mongoDB και η δεύτερη αφορά την θύρα που ακούει. Αυτόματα, αυτές είναι η διαδρομή (αν είναι σε τοπικό περιβάλλον) «mongodb:localhost» και η θύρα με το νούμερο 27017. Απευθείας, από την pymongo δημιουργείται η βάση δεδομένων με το όνομα «database», αν υπάρχει ήδη, τότε απλά συνδέεται. Από εκεί και πέρα δημιουργούνται τα collections ή αν υπάρχουν ήδη πραγματοποιείται η σύνδεση σε αυτά. Γενικά, όλος ο κώδικας στην python είναι γραμμένος με συναρτησιακή λογική, οπότε έχουν φτιαχτεί όλες οι συναρτήσεις (functions) για την εγγραφή δεδομένων, προβολή, διαγραφή και οποιαδήποτε άλλη διεργασία χρειαζόταν.

Η βιβλιοθήκη σε python «pandas»¹⁰ παρείχε όλες τις απαραίτητες λειτουργίες και ακόμα παραπάνω όσον αφορά τον ενδιάμεσο φορέα για την διαχείριση των δεδομένων. Η «pandas» είναι μια υψηλού επιπέδου βιβλιοθήκη, ευέλικτη και γρήγορη, κατάλληλη για τον χειρισμό δεδομένων. Κύρια συστατικά της είναι τα series και τα dataframes.

⁸ Περισσότερες πληροφορίες για την βιβλιοθήκη pymongo εδώ: <https://mongoosejs.com/>

⁹ Περισσότερες πληροφορίες για την βιβλιοθήκη pymongo εδώ: <https://pymongo.readthedocs.io/en/stable/>

¹⁰ Περισσότερες πληροφορίες για την βιβλιοθήκη pandas εδώ: <https://pandas.pydata.org/>

Τα series είναι μια στήλη ή αλλιώς ένας μονοδιάστατος πίνακας. Το dataframe είναι συνολικά ο πίνακας που κρατάει όλα τα δεδομένα και αποτελείται από πολλά series. Όλα τα δεδομένα που εξάγονται από τα ειδησεογραφικά ή από το Twitter αρχικά αποθηκεύονται προσωρινά στην μνήμη με την χρήση dataframes. Στην συνέχεια τα δεδομένα πρέπει να αποθηκευτούν στην ΒΔ. Για την εισαγωγή των δεδομένων στην ΒΔ χρησιμοποιήθηκε η μέθοδος της pymongo, « **insert_many()** ». Είναι η αντίστοιχη εντολή της native mongo shell γλώσσας « **insertMany()** », σε αυτήν εισάγεται σαν παράμετρος ένας πίνακας με όλα τα έγγραφα προς εισαγωγή στην ΒΔ. Ωστόσο τα δεδομένα είναι αρχικά στην μορφή dataframes, αυτή η μορφή είναι ξεχωριστή οντότητα και πρέπει να μετατραπεί σε πίνακα που αναγνωρίζει η μέθοδος « **insert_many()** ». Υπάρχει η μέθοδος της pandas για αυτήν την δουλειά «**to_dict()**», αυτή μετατρέπει το dataframe σε dictionary. Ωστόσο παραμετροποιείται το dataframe και μπορεί να εξαχθεί πίνακας, με την παράμετρο «**orient='records'**». Συνολικά δηλαδή έχουμε την εντολή:

```
<collection>.insert_many(<dataframe>.to_dict('records'))
```

Όπου το collection είναι το επιλεγμένο collection που θα εισαχθούν τα δεδομένα, και όπου το dataframe είναι τα δεδομένα σε μορφή dataframe που έχουν όλη την πληροφορία για εισαγωγή στην ΒΔ. Για την αποφυγή διπλότυπων ή μη αποθήκευση ανεπιθύμητης πληροφορίας τα δεδομένα ελέγχονται με την pandas.

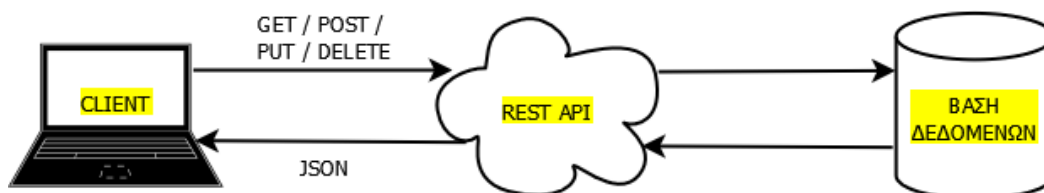
Για την προσπέλαση των δεδομένων χρησιμοποιήθηκε ελάχιστα η γλώσσα της pymongo για απευθείας queries στην ΒΔ ώστε να εξαχθεί η επιθυμητή πληροφορία. Αντίθετα χρησιμοποιήθηκε η βιβλιοθήκη pandas. Αρχικά χρησιμοποιήθηκε η pymongo για την προσωρινή αποθήκευση των δεδομένων στην μνήμη και μετέπειτα το πέρασμα τους στην pandas. Δηλαδή τα collection εξάγονται συνολικά και έπειτα κρατιέται η πληροφορία που είναι αναγκαία μέσα σε ένα dataframe. Έχουν δημιουργηθεί οι κατάλληλες μέθοδοι για την λειτουργία που χρειάζεται να γίνει από το πρόγραμμα ανά πάσα στιγμή. Η δομή για κάθε collection είναι να υπάρχει ένα module της python όπου εκεί μέσα υπάρχει όλος ο σχετικός κώδικας που αφορά την επικοινωνία μεταξύ της βάσης δεδομένων και της python. Για όλα τα collections υπάρχει η συνάρτηση get<collection>DB, όπου collection είναι μία από τις υπάρχουσες collections της ΒΔ. Για παράδειγμα η συνάρτηση getTrendsDB(), όπως το όνομα φανερώνει, η δουλειά του είναι να επιστρέφει τις τάσεις που είναι αποθηκευμένες στην ΒΔ. Όλα τα collections έχουν αυτήν την λειτουργία οπότε έχουν υλοποιηθεί αντιστοίχως οι ίδιες περίπου συναρτήσεις. Επιπρόσθετα υπάρχει η επιλογή σε αυτές του τύπου τις συναρτήσεις, να επιστραφούν τα δεδομένα ανάλογα με την ημερομηνία που θα επιλεγεί. Γενικά όσον αφορά τις συναρτήσεις στον χώρο του προγραμματισμού, υπάρχει η έννοια overloading. Το overloading ή στα ελληνικά η υπερφόρτωση επιτρέπει διαφορετικές συναρτήσεις να έχουν διαφορετικό όνομα αλλά άλλο πλήθος παραμέτρων. Στην python μιμείται αυτή η συμπεριφορά με τη χρήση του ειδικού χαρακτήρα (*) μπροστά από την παράμετρο. Είναι κοινή σύμβαση να χρησιμοποιείται η λέξη args, συντομογραφία του arguments, δηλαδή παραμέτρους. Το « ***args** » εισάγεται στην συνάρτηση σαν μια λίστα από παραμέτρους. Είναι στο χέρι του προγραμματιστή να ορίσει μετά την λειτουργικότητα και την διαφοροποίηση της εξόδου της συνάρτησης με βάση τον αριθμό των παραμέτρων. Αυτό γίνεται με την μη υποχρεωτική παράμετρο args που η πρώτη παράμετρος είναι η ελάχιστη ημερομηνία και η δεύτερη η μέγιστη ημερομηνία. Στην συνέχεια αποθηκεύονται προσωρινά όλα τα δεδομένα από το collection και μετέπειτα με την χρήση φίλτρου επιστρέφεται η επιθυμητή πληροφορία. Ανάλογα το περιεχόμενο των δεδομένων τα args μπορεί να είναι διαφορετικά ώστε να έχουν νόημα στην εξαγωγή πληροφορίας. Στην getResultsDB() όπου επιστρέφει τα αποτελέσματα του προγράμματος, στην εργασία είχε νόημα να είναι δυνατόν να επιστρέφονται τα αποτελέσματα των τελευταίων πέντε τάσεων του twitter. Οπότε υλοποιήθηκε αυτή η λειτουργία, στην περίπτωση που δεν εισάγονται παράμετροι, η συνάρτηση επιστρέφει όλα τα δεδομένα του collection με τα αποτελέσματα. Σε όλα τα modules που αφορούν την επικοινωνία με την ΒΔ, έχουν οριστεί διάφορες τέτοιου τύπου overloading συναρτήσεις.

Κάθε ένα συγκεκριμένο διάστημα διαγράφονται διάφορα δεδομένα της ΒΔ για εξοικονόμηση χώρου στον σκληρό δίσκο. Επίσης η ΒΔ χρησιμοποιείται ταυτόχρονα και από τον client οπότε τα collections πρέπει να έχουν έτοιμα κάποια δεδομένα προς προβολή για τον χρήστη. Έτσι σε κάθε κύκλο που ολοκληρώνεται από το πρόγραμμα κάποια collections αδειάζουν τελείως με σκοπό να αντικατασταθούν από τα καινούργια δεδομένα. Αν εμφανιστεί ξανά τάση που είχε εμφανιστεί στο παρελθόν, αυτό σημαίνει ότι τα αποθηκευμένα tweets από την προηγούμενη φορά που ήταν η ίδια τάση είναι ανενημέρωτα και θα ήταν σφάλμα να χρησιμοποιηθούν τα ίδια με τα παλιά. Οπότε τα

παλιά διαγράφονται μιας και δεν χρησιμεύουν κάπου. Κατά την διάρκεια ανάπτυξης της εφαρμογής ήταν χρήσιμο να μπορεί να γίνει διαγραφή ολόκληρων collections, ωστόσο στην τελική έκδοση δεν χρησιμοποιούνται όλες αυτές οι συναρτήσεις. Για αυτό υπάρχει σε κάθε module η συνάρτηση `clear<collection>DB()`, όπου το collection είναι το εκάστοτε collection της ΒΔ. Και αυτό που κάνει είναι να διαγράφει όλα τα δεδομένα από το collection που έχει επιλεγεί. Για αυτήν την λειτουργία χρησιμοποιήθηκε η εντολή `<collection>.delete_many({})`, όπου η μέθοδος της `rymongo` διαγράφει πολλά έγγραφα, και συγκεκριμένα αυτά που αντιστοιχούν στο query που εισάγεται σαν παράμετρος. Η παράμετρος που εισάγεται εδώ είναι για την διαγραφή όλων των στοιχείων οπότε εισάγεται ένα άδειο object που σηματοδοτεί την αντιστοίχιση όλων των εγγράφων. Για την διαγραφή των tweets, εφόσον έχουμε στην κατοχή μας τα καινούργια tweets μπορούν να διαγραφούν όλα τα παλιά. Το query εδώ αντιστοιχεί σε όλα τα tweets που έχουν αυτή την συγκεκριμένη τάση. Και μετέπειτα εισάγονται τα καινούργια tweets με αυτήν την τάση.

2.3 Ο ρόλος της βάσης δεδομένων στον client

Η διεπαφή (ιστοσελίδα) που κατασκευάστηκε πρέπει να παίρνει από κάπου τα δεδομένα για να εμφανίζονται στον χρήστη. Ο client φυσικά, δεν μπορεί να συνδεθεί απευθείας στην βάση δεδομένων λόγω ασφάλειας και ακεραιότητας των δεδομένων, για αυτόν το σκοπό ο server είναι ο διαμεσολαβητής των δεδομένων. Ο server με βάση του τι χρειάζεται ο client, επικοινωνεί με την ΒΔ και επιστρέφει την πληροφορία πίσω στον client. Ο client αυτό που επιτρέπεται να κάνει στην ΒΔ είναι να έχει πρόσβαση σε κάποια δεδομένα μόνο για προβολή. Για αυτό τον σκοπό φτιάχτηκε ένα restful API για τον client ώστε να επιτραπεί η επικοινωνία μεταξύ της ΒΔ. Η επικοινωνία είναι έτσι όπως παρουσιάζεται στο παρακάτω διάγραμμα. Το client μπορεί να στείλει κάποιο αίτημα (request GET, POST, PUT, DELETE) στον server μέσω του REST API το οποίο με την σειρά του στέλνει κατάλληλο αίτημα στην ΒΔ και επιστρέφεται το αποτέλεσμα πίσω στον client με την μορφή JSON. Τα requests PUT και DELETE δεν υλοποιήθηκαν γιατί δεν χρειάστηκαν, απεναντίας υλοποιήθηκαν τα GET και POST requests.



Εικόνα 3 Η επικοινωνία του client με την βάση δεδομένων μέσω του rest API.

Ο server κάνει απευθείας queries στην ΒΔ με την γλώσσα που έχει ορίσει η βιβλιοθήκη mongoose. Η τελευταία είναι παρόμοια με την `rymongo` μόνο που είναι σχεδιασμένη για την γλώσσα `Node.js`¹¹. Οι mongoose και `rymongo` επειδή και οι δύο βιβλιοθήκες βασίστηκαν με την native γλώσσα `mongodb`, έχουν παρόμοια σύνταξη. Εδώ επίσης ορίζεται το schema των collections, αν και σε αυτήν την φάση έχουν οριστεί ήδη από την `rymongo`, είναι υποχρεωτικό να δηλώνονται και στην mongoose. Στην `rymongo` όπως έχει ήδη ειπωθεί τα collections ολόκληρα αποθηκεύονται προσωρινά στην μνήμη του server με σκοπό να γίνει η επεξεργασία των δεδομένων και να κρατηθούν μόνο αυτά που χρειάζονται. Επειδή στον server δεν χρειάζεται να γίνει κάποια ειδική προετοιμασία των δεδομένων αυτή η διαδικασία είναι αχρείαστη, οπότε και τα δεδομένα εξάγονται από την ΒΔ με την απευθείας χρήση των queries που προσφέρει η βιβλιοθήκη mongoose. Κάποια queries είναι απλά και άλλα πιο πολύπλοκα, κυρίως

¹¹ Η `Node.js` είναι server-side γλώσσα προγραμματισμού και έχει την ίδια σύνταξη με την γλώσσα `JavaScript`. Η `Node.js` έκανε την εμφάνισή της στον κόσμο των διαδικτυακών εφαρμογών τα τελευταία χρόνια με τις τεχνολογίες PWA (Progressive Web Apps) και άλλες προεκτάσεις. Για περισσότερες πληροφορίες εδώ: <https://nodejs.org/en/>

χρησιμοποιήθηκαν δύο μορφές ερωτημάτων. Στα πιο απλά ερωτήματα χρησιμοποιήθηκε η συνάρτηση `find()` που επιστρέφει τα δεδομένα του collection. Για παράδειγμα ο κώδικας:

```
searchQuery.find(
  { hashtag: hashtag },
  '-_id dateDownloaded hashtag rank',
  {
    limit: 5000,
    sort: {
      dateDownloaded: -1 //Sort by dateDownloaded ASCENDING
    }
  }
);
```

Αυτό το ερώτημα αφορά το collection `searchQuery`, σε αυτό το collection αποθηκεύονται όλες οι τάσεις που έχει βρει το πρόγραμμα. Έχει μόνο τρία πεδία, την ημερομηνία που εμφανίστηκε η τάση, το όνομα της τάσης και την κατάταξη της στον πίνακα των τάσεων του Twitter. Με την συνάρτηση `find()`, επιλέγεται το κλειδί αναζήτησης (`hashtag`). Το συγκεκριμένο query είναι πολύ απλό, διότι ζητείται απλώς να βρεθούν όλα τα έγγραφα που έχουν στο πεδίο `hashtag`, που ισούνται με την μεταβλητή `hashtag`. Στην δεύτερη παράμετρο διευκρινίζεται η προβολή (παρόμοιο με το `SELECT` της `SQL`) και τέλος στην τρίτη παράμετρο ορίζονται κάποιες ρυθμίσεις του query, όπως να περιοριστούν τα έγγραφα σε 5000 και να επιστραφούν σε φθίνουσα σειρά με βάση την ημερομηνία τους. Το αποτέλεσμα θα είναι να επιστραφούν τα τελευταία έγγραφα, βάσει της ημερομηνίας, που είναι αποθηκευμένα στην ΒΔ. Αυτή η δομή επικράτησε στα πιο απλά ερωτήματα, ωστόσο στα πιο πολύπλοκα χρησιμοποιήθηκε τελείως διαφορετική σύνταξη με την χρήση του aggregation.

Το Aggregation framework, έρχεται ενσωματωμένο με την `mongoose` και είναι ένας άλλος πιο ανώτερος τρόπος για να γράφονται τα ερωτήματα (queries). Γενικά τα ερωτήματα του είναι πιο ευδιάκριτα, εξελιγμένα και προσανατολισμένα να είναι αποδοτικά, διότι τα ερωτήματα επεξεργάζονται απευθείας από την `mongoDB` και όχι από το driver (στην περίπτωση μας, τις βιβλιοθήκες `mongoose` και `rymongo`).

```
results
.aggregate([
  {
    $match: {
      searchQuery: {
        $regex: hashtagKeyword,
        $options: 'i'
      }
    },
    {
      $sort: { cosSimil: -1 }
    },
    {
      $group: {
```

Ένα παράδειγμα της aggregation λογικής βρίσκεται στα αριστερά, το ερώτημα αφορά το collection **results**, όπου είναι αποθηκευμένα όλα τα αποτελέσματα που έχουν παραχθεί από το πρόγραμμα. Σε αυτό το collection υπάρχουν μαζεμένα όλα τα αποτελέσματα μικτά, επειδή κάποιες τάσεις παραμένουν για αρκετή ώρα στον πίνακα των τάσεων του Twitter, υπάρχουν όλα τα αποτελέσματα ανεξάρτητα αν βρέθηκε κάποια καλύτερη συσχέτιση ή χειρότερη. Αρχικά χρησιμοποιείται η δεσμευμένη λέξη `$match`, όπου ταιριάζει τα πεδία, εδώ ταιριάζετε το πεδίο `searchQuery` με την χρήση regular expressions¹². Στην συνέχεια ταξινομείται με φθίνουσα σειρά με βάση το πεδίο `cosSimil` που αφορά το score της ομοιότητας που έχει με κάποιο άρθρο. Ο συνδυασμός της παραμέτρου `$group` και `$replaceRoot`, ομαδοποιεί τα έγγραφα με βάση το `hashtag` τους και κρατάει μόνο το πρώτο. Με αποτέλεσμα να υπάρχουν όλα τα καλύτερα

¹² Τα regular expressions είναι μια ακολουθία από χαρακτήρες που ορίζει ένα μοτίβο αναζήτησης κειμένου. Περισσότερες πληροφορίες εδώ: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions

```
    _id: '$searchQuery',  
    doc: { $first: '$$ROOT' }  
  }  
},  
{  
  $replaceRoot: {  
    newRoot: '$doc'  
  }  
}  
}  
]);
```

αποτελέσματα (μεγαλύτερο `cosSimil`), δηλαδή συσχετίσεις μεταξύ άρθρων και τάσεων για κάθε ξεχωριστή τάση.

3. Τα Δεδομένα

Ένα σημαντικό μέρος του χρόνου και της προσπάθειας του ερευνητή καταναλώθηκε στην προετοιμασία των δεδομένων. Τις περισσότερες φορές τα δεδομένα που είναι αποθηκευμένα στις βάσεις δεδομένων ή σε αρχεία δεν είναι στην κατάλληλη μορφή και κατάσταση για να αναλυθούν. Οπότε είναι αναγκαίο να ερευνηθούν μοτίβα στα δεδομένα που δεν είναι κατάλληλα για την επεξεργασία και να αφαιρεθούν ή να τροποποιηθούν.

Τα δεδομένα δημιουργούνται συνεχώς και συλλέγονται από το πρόγραμμα σε σχεδόν πραγματικό χρόνο. Τα δεδομένα κυρίως είναι κείμενα από δημοσιεύματα στο Twitter και από τα ειδησεογραφικά. Τα υπόλοιπα δεδομένα που συλλέγονται είναι τα μεταδεδομένα συνδεδεμένα με τα κείμενα. Μέσα σε όλα αυτά μπορεί να υπάρχουν δεδομένα που λείπουν, ανώμαλα δεδομένα, διπλότυπα και λανθασμένα. Μέσα από την παρατήρηση και επανάληψη, δημιουργήθηκαν μοτίβα που ανιχνεύουν και διορθώνουν τα λάθη. Δεν ήταν δυνατόν να δημιουργηθεί ένα μοντέλο εξαρχής που θα επιδιορθώνει τα δεδομένα, λόγω του ότι το περιεχόμενο των αρθρογράφων καθώς και των χρηστών του Twitter, συνεχώς παράγεται με διαφορετικούς τρόπους. Επιπλέον μπορεί να υπάρχουν τεχνικά προβλήματα στο scraping των ειδησεογραφικών, όπως για παράδειγμα κατά την διάρκεια της λειτουργίας του προγράμματος, ένα ειδησεογραφικό άλλαξε σχεδιαστικά την σελίδα με αποτέλεσμα να μετατραπεί η δομή του και το πρόγραμμα να αναζητά περιεχόμενο εκεί όπου δεν υπάρχει. Επίσης τα ειδησεογραφικά, ανά καιρούς δημιουργούν καινούργιες κατηγορίες που η HTML μορφή τους είναι διαφορετική από αυτά που μπορεί το πρόγραμμα να αναγνωρίσει. Άρα αργά ή γρήγορα το πρόγραμμα θα είναι ευάλωτο σε τέτοιες αλλαγές, για αυτό είναι αναγκαίο να προβλέπονται τέτοια λάθη καθώς και να επιδιορθώνονται με την πάροδο του χρόνου.

3.1 Διπλότυπα

Πολλές φορές μπορεί να υπάρξουν διπλότυπα έγγραφα, αυτό μπορεί να συμβεί με ποικίλους τρόπους. Αρχικά, μπορεί να πραγματοποιηθεί κάποιο σφάλμα στην εκτέλεση του προγράμματος και να κατεβούν άρθρα τα οποία είχαν ξανακατεβεί. Έχει παρατηρηθεί ότι τα άρθρα κάποιες φορές υπάρχουν δύο φορές ανάμεσα από δύο σελίδες (συνήθως είναι τελευταία στην πρώτη σελίδα και πρώτη στην δεύτερη). Τα άρθρα είναι ταξινομημένα βάσει ημερομηνίας στα ειδησεογραφικά. Κάποια άρθρα είναι επισημασμένα (pinned), οπότε θα υπάρχουν πρώτα στην κατάταξη των άρθρων της σελίδας ανεξάρτητα από την ημερομηνία τους και μερικές σελίδες πιο πίσω θα υπάρχει ξανά με την κατάταξη της ημερομηνίας τους. Τα tweets δεν πάνε πίσω, καθώς σε αυτά είναι σίγουρο ότι θα υπάρχουν διπλότυπα. Αυτό διότι κάθε φορά που ζητούνται tweets από το Twitter API, ζητούνται μαζί τα πιο **πρόσφατα**¹³ και δεν υπάρχει τρόπος που να βρέθηκε να ζητηθούν tweets τα οποία έχουν ληφθεί ήδη. Αυτό που γίνεται και στις δύο περιπτώσεις, δηλαδή για τα tweets και τα άρθρα είναι να ελέγχεται κάθε φορά αν υπάρχουν ήδη στη βάση δεδομένων. Έτσι αντιστοίχως, κείμενα που υπάρχουν ήδη στην βάση δεδομένων δεν αποθηκεύονται ενώ αντιθέτως αποθηκεύονται. Η βάση δεδομένων στη mongoDB τρέχει κυρίως στην μνήμη του υπολογιστή οπότε αυτή η διαδικασία γίνεται εξαιρετικά γρήγορα.

¹³ Το Twitter API παρέχει τρεις τρόπους εύρεσης tweets, τα πιο πρόσφατα, τα πιο δημοφιλή και μικτά. Παρατηρήθηκε ότι ο πιο κατάλληλος τρόπος για την αποφυγή διπλότυπων ήταν να επιλεγεί η αναζήτηση στα πιο πρόσφατα tweets.

3.2 Missing δεδομένα

Στην διαδικασία του scraping όπως αναλύθηκε παραπάνω πολλά σφάλματα μπορούν να προκύψουν. Το πιο σύνηθες που παρατηρήθηκε ήταν να συλλέγονται τα μεταδεδομένα του άρθρου και όχι το κείμενο του. Αυτό εξηγείται, διότι η HTML ή XML της αρχικής σελίδας έχει ίδια δομή για όλα τα άρθρα. Ωστόσο κάθε άρθρο ξεχωριστά μπορεί να έχει ιδιόμορφη παρουσίαση με αποτέλεσμα και διαφορετική HTML. Οπότε στην ουσία να αδυνατεί ο scraper να αναγνωρίσει το κείμενο. Για αυτήν την περίπτωση, δημιουργήθηκαν λίστες με όλες αυτές τις ιδιαίτερες υποσελίδες των ειδησεογραφικών με σκοπό να αγνοούνται. Πολλά άρθρα μπορεί να μην έχουν καθόλου γραπτό λόγο, παρά μόνο μια εικόνα ή ένα βίντεο. Αυτά τα άρθρα επίσης αγνοούνται καθώς δεν έχουν να προσφέρουν κάτι στις αναλυτικές μεθόδους που επιλέχθηκαν στο εγχείρημα. Άλλες φορές τα ειδησεογραφικά που επιλέχθηκαν μπορεί να συνεργάζονται με τρίτα ειδησεογραφικά και να εμπεριέχουν στην αρχική τους σελίδα άρθρα που δεν αντιστοιχούν σε αυτά. Χειροκίνητα δημιουργήθηκε μια λίστα με όλα τα τρίτα ειδησεογραφικά με σκοπό να αγνοούνται τέτοιου είδους άρθρα. Και αυτό έγινε διότι είναι σημαντικό τα άρθρα των ειδησεογραφικών να αντιστοιχούν στα ίδια και να μην αναγνωρίζονται από τον αλγόριθμο σαν ιδιοκτησία τους.

3.3 Καθαρισμός δεδομένων

Οι αρθρογράφοι στα ειδησεογραφικά παράγουν επίσημο λόγο με ορθή σύνταξη, ορθογραφία και γραμματική. Στην περίπτωση των tweets όμως αυτό δεν ισχύει, καθώς οι χρήστες έχουν την πλήρη ελευθερία να γράψουν ότι θέλουν και όπως θέλουν. Έχουν παρατηρηθεί slang λέξεις, ποικίλες συντομογραφίες καθώς και άλλες μορφές κειμένου όπως για παράδειγμα emoticons, mentions, links, hashtags και greeklish. Ωστόσο τα άρθρα και τα tweets, περιέχουν stopwords. Τα stopwords είναι οι ενδιαμέσες λέξεις που υπάρχουν στα ελληνικά, όπως για παράδειγμα τα άρθρα, αντωνυμίες και επιρρήματα. Αφαιρώντας αυτές τις λέξεις, αφαιρείται η χαμηλού επιπέδου πληροφορία από τα κείμενα ούτως ώστε να δοθεί περισσότερη αξία στην σημαντική πληροφορία. Παρατίθεται η παρακάτω σύγκριση wordclouds, όπου τα wordclouds βοήθησαν αρκετά αρχικά τον ερευνητή στην κατανόηση των δεδομένων.



Table 2 Σύγκριση wordcloud, μεταξύ ανεπεξέργαστου κειμένου (αριστερά), με επεξεργασμένο καθαρισμένο κείμενο (δεξιά)

Το wordcloud είναι μια εικόνα με ένα σύνολο από λέξεις. Για να παραχθεί χρησιμοποιήθηκε η βιβλιοθήκη «word_cloud» σε python. Χάριν του παραδείγματος συλλέχθηκαν από την βάση δεδομένων όλα τα tweets για την τάση «#σεισμός», μετρήθηκε η συχνότητα όλων των λέξεων που υπάρχουν μέσα σε αυτά. Η τάση «#σεισμός» αναφέρεται στον σεισμό που συνέβη στην Αθήνα μικρής εντάσεως 2.8 ρίχτερ. Το αποτέλεσμα είναι μια εικόνα που

δείχνει τις λέξεις με το μεγαλύτερο βαθμό συχνότητας. Όσο πιο μεγάλη εμφανίζεται η λέξη στην εικόνα τόσες φορές περισσότερο εμφανίζεται στα tweets. Η αριστερή εικόνα είναι οι λέξεις των tweets πριν την επεξεργασία και δεξιά είναι μετά την επεξεργασία. Πριν την επεξεργασία οι πρώτες λέξεις που είναι πιο ευδιάκριτες είναι οι: σεισμος, σεισμός, https, στην, και, δεν, να κτλ. Παρατηρούμε ότι υπάρχουν λέξεις με τόνους και άλλες χωρίς, αυτές ερμηνεύονται σαν διαφορετικές λέξεις από το πρόγραμμα. Επίσης υπάρχουν με μεγάλη συχνότητα λέξεις stopwords καθώς και χαρακτηριστές χωρίς αξία όπως το https που προφανώς υπάρχει, επειδή μέσα στα tweets εμπεριέχονται links. Στην δεύτερη εικόνα δεξιά φαίνονται αρκετά ξεκάθαρα κάποιες λέξεις να ξεχωρίζουν όπως: επικεντρο, σεισμος, αθηνα, μαρουσι, ριχτερ, αισθητος. Εύκολα μπορεί κάποιος να κάνει την υπόθεση πλέον ότι: Στην Αθήνα έγινε αισθητός σεισμός με επίκεντρο το Μαρούσι (πιθανός το Μαρούσι). Τώρα είναι ξεκάθαρο και σε οπτική μορφή πόσο σημαντικό ρόλο παίζει η εκκαθάριση του κειμένου καθώς και πόση μεγαλύτερη αξία είναι δυνατόν να εξαχθεί.

Οι λέξεις όπως θα έχει γίνει αντιληπτό μέχρι στιγμής ερμηνεύονται από τον υπολογιστή σαν συμβολοσειρές. Στόχος είναι όλες οι ίδιες λέξεις, ανεξάρτητα αν έχουν τονισμό ή όχι, αν είναι με κεφαλαία ή με πεζά. Αρχικά για την αφαίρεση των stopwords χρησιμοποιήθηκε η βιβλιοθήκη NLP spaCy. Για την ακρίβεια δεν χρησιμοποιήθηκε ακριβώς η βιβλιοθήκη αλλά μια λίστα¹⁴ που εμπεριέχετε σε αυτή από stopwords στα ελληνικά. Κάθε φορά που γίνεται ο καθαρισμός των δεδομένων αφαιρούνται αυτές οι λέξεις. Όπως επώθηκε πιο πάνω, οι λέξεις με τόνο ερμηνεύονται διαφορετικά από τις ίδιες λέξεις δίχως τόνο. Επιλέχθηκε από όλες τις λέξεις τα γράμματα να μετατραπούν από κεφαλαία σε πεζά και να αφαιρεθεί ο τονισμός από όλα τα φωνήεντα. Επιπλέον links, hashtags, mentions αφαιρέθηκαν επίσης. Στην πραγματική τους μορφή τα κείμενα είναι κωδικοποιημένα και περιέχουν κάποιους ιδιαίτερους ASCII χαρακτήρες, όπως για παράδειγμα το /n (νέα γραμμή), /t (στηλοθέτης), /r (return) κ.α. τα οποία αφαιρέθηκαν. Οι αριθμοί δεν προσθέτουν κάποια ιδιαίτερη αξία οπότε και αυτοί αφαιρέθηκαν.

3.4 Stemming

Η διαδικασία stemming χρησιμοποιείται για να μειώσει μια λέξη στη ρίζα της, για να αυξήσουν το recall στα έγγραφα, δίνοντας μας με αυτό το τρόπο πιο σχετικά αποτελέσματα. Χρησιμοποιείται ευρέως στους κλάδους της ανάκτησης πληροφορίας και στην λεξικολογική ανάλυση. Υπάρχουν διάφοροι τρόποι για να πραγματοποιηθεί η διαδικασία του stemming, από χειροκίνητες μέχρι αυτόματες μηχανικές μεθόδους. Με το πλήθος των δεδομένων που συλλέγονται σε αυτήν την εργασία είναι πολύ σημαντικό η πληροφορία να αναπαρασταθεί με ακρίβεια. Ο παραδοσιακός τρόπος σε μια αναζήτηση για παράδειγμα, θα ήταν να ψαχθεί το ερώτημα λέξη προς λέξη. Αυτή η προσέγγιση είναι χρονοβόρα και μπορεί να χάσει έγγραφα ίδιας σημαντικότητας με αυτά που μπορεί να βρει. Για να αποφευχθεί αυτή η κατάσταση, η διαδικασία stemming έχει χρησιμοποιηθεί πολλάκις σε διάφορα συστήματα ανάκτησης πληροφορίας για να βελτιώσουν την ακρίβεια (precision)¹⁵ τους.

Ακόμα και αν δύο λέξεις έχουν ίδια ρίζα αλλά διαφορετικές καταλήξεις θα πρέπει να αντιμετωπίζονται σαν ίδιες (stemming). Για παράδειγμα ας πάρουμε το ρήμα «τρέχω», λέξεις που ανήκουν στην ίδια ρίζα είναι στο γ' ενικό πρόσωπο η λέξη «τρέχουν» και στο γ' πληθυντικό πρόσωπο η λέξη «τρέχει». Οι δύο τελευταίες είναι δόκιμο να μετατραπούν στη ρίζα τους και να αντιμετωπίζονται σαν ίδιες, καθώς προσφέρουν μεγαλύτερες πιθανότητες να βρεθούν ανάμεσα σε δύο διαφορετικά κείμενα. Ο εκάστοτε συγγραφέας κάποιου άρθρου ή tweet μπορεί να χρησιμοποιήσει διαφορετικούς χρόνους στη συγγραφή του κειμένου τους. Βέβαια υπάρχουν περιπτώσεις όπου η αλλαγή της γραμματικής στις λέξεις να προσφέρουν εντελώς διαφορετικό νόημα. Εδώ αγνοούνται τέτοιες περιπτώσεις καθώς με βάση τις αρχικές δοκιμές στην απόδοση της μαθηματικής συσχέτισης το αποτέλεσμα με stemming είχε σημαντική βελτίωση σε σχέση με το αποτέλεσμα δίχως stemming. Η αρχική ιδέα προέρχεται από τον

¹⁴ Η λίστα με διαθέσιμα stopwords στα ελληνικά μπορεί να βρεθεί εδώ:

https://github.com/explosion/spaCy/blob/master/spacy/lang/el/stop_words.py

¹⁵ Η ακρίβεια ή στα αγγλικά precision είναι ο αριθμός ανακτημένων σχετικών εγγράφων σε σχέση με τα ανακτηθέντα έγγραφα. Το recall από την άλλη είναι ο αριθμός ανακτημένων σχετικών εγγράφων σε σχέση με όλα τα σχετικά έγγραφα. Και στις δύο περιπτώσεις η ικανότητα να είναι υψηλά αυτά τα μέτρα (100% ή κοντά σε αυτό) είναι θεμιτό (M. Buckland, 1994).

Porter (1980)¹⁶ και συνοψίζεται, αρχικά σε μια λίστα με όλες τις καταλήξεις μιας γλώσσας που θα λειτουργεί ως οδηγό για να εντοπίζονται σε λέξεις και να αφαιρούνται. Στην παρούσα εργασία για την επίτευξη αυτού του σκοπού χρησιμοποιήθηκε η βιβλιοθήκη σε python greek-stemmer¹⁷. Σημασία δεν έχει η λέξη που θα μετατρέπεται σε ρίζα να είναι ορθώς γραμμένη αλλά πολλές εκδοχές της ίδιας λέξης να καταλήγουν στην ίδια ρίζα. Έτσι για παράδειγμα η λέξη «θάλασσεσ» και «θάλασσα» μέσω του «greek-stemmer» καταλήγουν στην ίδια ρίζα «θαλασσ».

¹⁶ Στην αρχή υπήρχε η λίστα με τις καταλήξεις (suffix) μόνο στα αγγλικά, βέβαια ο αλγόριθμος μπορούσε να παρέχει την ίδια λειτουργία και σε άλλες γλώσσες αν υπήρχε το λεξιλόγιο των καταλήξεων. Το 2006 δημιουργήθηκε μια υψηλού επιπέδου γλώσσα προγραμματισμού με το όνομα Snowball που κάλυπτε πολλές γλώσσες (Willett, Peter 2006).

¹⁷ Η βιβλιοθήκη «greek-stemmer», όπως αναφέρουν οι προγραμματιστές της είναι υλοποίηση του «Skroutz's Ruby Greek stemmer» σε Python: https://github.com/alup/python_greek_stemmer

4. Επεξεργασία των δεδομένων

Εφόσον τα δεδομένα είναι προεπεξεργασμένα περνάμε στην επόμενη φάση της επεξεργασίας τους για να εξαχθούν αποτελέσματα. Αυτό που είναι θεμιτό σε αυτήν τη φάση είναι να βρεθούν συσχετίσεις μεταξύ των εγγράφων, δηλαδή μεταξύ των άρθρων από τα ειδησεογραφικά και τα tweets των χρηστών από το Twitter. Οι συσχετίσεις μπορεί να είναι υψηλές ή χαμηλές, υψηλές όταν τα tweets έχουν κάποια σημασιολογική συσχέτιση με κάποιο άρθρο ενώ χαμηλές όταν συμβαίνει το αντίθετο. Το συμπέρασμα που βγαίνει στο τέλος είναι η απάντηση στο ερώτημα ποια ειδησεογραφικά έγραψαν για κάποια τάση του Twitter και ποια όχι. Αν τα ειδησεογραφικά έγραψαν στην ολομέλεια τους για κάποια τάση, αυτό σημαίνει ότι το γεγονός που εξιστορείται μέσα από τα tweets ήταν σημαντικό. Αν πάλι δεν γράφτηκε καθόλου από μεριάς ειδησεογραφικών τότε είτε αγνοήθηκε ή δεν αποτέλεσε πόλο έλξης και αξίας για να δημοσιευθεί. Αυτή η πληροφορία δεν εξάγεται στα εξαγόμενα αποτέλεσμα αυτής της εργασίας παρά μένει σαν ερώτημα προς απάντηση από τους χρήστες που αναλύουν τα δεδομένα με την ίδια τους την κρίση.

4.1 Η αναπαράσταση των εγγράφων με Bag of Words

Το Bag of Word στα ελληνικά μεταφράζεται σαν «Σάκος από Λέξεις», πολύ σωστά και στοχευμένα, διότι το συγκεκριμένο μοντέλο παίρνει όλες τις λέξεις συνολικά από τα έγγραφα και είναι σαν να τις βάζει σε έναν σάκο. Στον κόσμο των υπολογιστών δηλαδή δημιουργείται το λεξικό από όλες τις λέξεις των εγγράφων σε κάποια προγραμματιστική μορφή (λίστα, πίνακας κ.τ.λ.). Το συγκεκριμένο λεξικό δεν έχει κάποιο είδος ταξινόμησης, δεν υπάρχουν διπλότυπα και κάθε λέξη ορίζεται από την μοναδικότητα του σε σύμβολα με την ίδια σειρά. Το λεξικό είναι ένα διάνυσμα στον πολυδιανυσματικό χώρο, έτσι ώστε κάθε λέξη να αντιστοιχεί σε μία ξεχωριστή λέξη. Έτσι το λεξικό ορίζεται μαθηματικά:

$$\vec{d}_j = (w_1, w_2, w_3, \dots, w_j)$$

Όπου το d είναι το λεξικό, το j είναι το πλήθος των λέξεων όλων των εγγράφων και το w οι λέξεις. Προτάσεις ονομάζονται όλα τα κείμενα ξεχωριστά, ανεξαρτήτου μεγέθους που μετατρέπονται με την μέθοδο BoW στην διανυσματική αναπαράσταση. Οι λέξεις στο λεξικό εισέρχονται αρχικά με τυχαία σειρά, ωστόσο από την στιγμή που παραχθεί το λεξικό η σειρά των λέξεων μέσα σε αυτό πρέπει να διατηρηθεί. Η διαδικασία για να δημιουργηθεί μια πρόταση σε διανυσματική μορφή είναι απλή. Έχοντας ένα διάνυσμα μεγέθους ίσου με το μέγεθος του λεξικού (j), αρχικοποιημένο με μηδενικά. Κάθε λέξη της πρότασης αναζητείται στο λεξικό και προστίθεται σε αυτήν την θέση του πίνακα η τιμή 1. Η διαδικασία επαναλαμβάνεται μέχρι την τελευταία λέξη της πρότασης. Στην περίπτωση που κάποια λέξη βρεθεί δύο ή παραπάνω φορές, προστίθεται κάθε φορά η τιμή 1 στην ίδια θέση του πίνακα. Έτσι για παράδειγμα, έχουμε τρία υποθετικά έγγραφα:

1. τα ποντίκια έφαγαν τα ψίχουλα
2. οι γάτες έφαγαν τα ποντίκια
3. τα σκυλιά έφαγαν τις γάτες

Το λεξικό αποτελείται από όλες τις λέξεις των εγγράφων:

(τα, ποντίκια, έφαγαν, ψίχουλα, οι, γάτες, σκυλιά, τις)

Αν για παράδειγμα θέλουμε να αναπαραστήσουμε την πρόταση (1) διανυσματικά τότε έχουμε το διάνυσμα:

τα ποντίκια έφαγαν τα ψίχουλα $\rightarrow (2, 1, 1, 1, 0, 0, 0, 0)$

Το μοντέλο Bag of Words (BoW) έχει αρκετά συχνά υιοθετηθεί σε προβλήματα κατηγοριοποίησης εγγράφων λόγω της απλότητας του, των αξιοσέβαστων αποτελεσμάτων του και την εύκολη κατανόηση του στον τρόπο που λειτουργεί. Ωστόσο υπάρχουν κάποια μειονεκτήματα που πρέπει να συζητηθούν. Αρχικά το μοντέλο BoW κοιτάει στην επιφάνεια των λέξεων αγνοώντας την σημασιολογική συνάφεια και εννοιολογική πληροφορία των λέξεων ή φράσεων. Για παράδειγμα, οι δύο προτάσεις «Η Apple έβγαλε καινούργιο έξυπνο κινητό», «Τα καλύτερα smartphones τα έχει η Samsung» δεν έχουν καμία κοινή λέξη μεταξύ τους και το μοντέλο BoW θα τα αναπαραστήσει σαν δύο τελείως διαφορετικά διανύσματα. Ωστόσο, οι δύο προτάσεις είναι συσχετισμένες μεταξύ τους, λόγω του ότι οι εταιρείες «Apple» και «Samsung» παράγουν κινητά τηλέφωνα καθώς και οι δύο προτάσεις μιλάνε για τα προϊόντα τους. Το μοντέλο δεν κρατάει σειρά στις λέξεις της πρότασης και το σημασιολογικό περιεχόμενο των κειμένων χάνετε. Η σειρά των λέξεων είναι σημαντική στην πρόταση ώστε να αναγνωριστεί η διαφορά σε συνώνυμες λέξεις ή φράσεις με βάση τα συμφραζόμενα. Για παράδειγμα, η πρόταση «Ο τύπος φοράει καπέλο και γυαλιά.» με την πρόταση «Ο κίτρινος τύπος έχει πολλές πωλήσεις.», μοιράζονται την λέξη «τύπος», ωστόσο αναφέρονται σε τελείως διαφορετικά πράγματα. Αν κοιτάξουμε τις συμφραζόμενες λέξεις μπορούμε να κατανοήσουμε ότι στην πρώτη πρόταση η λέξη «τύπος» αναφέρεται σε κάποιο πρόσωπο και στην δεύτερη πρόταση αντιστοιχίζεται η λέξη εφημερίδα. (Deepu S. et al)

Δεδομένων των πλεονεκτημάτων αλλά και των μειονεκτημάτων, αυτή η τακτική χρησιμοποιήθηκε για την αναπαράσταση των κειμένων. Η διανυσματικοποίηση των κειμένων πραγματοποιείται σε σύντομο χρονικό διάστημα και αυτό συμβαίνει κάθε φορά που έρχεται κάποιο tweet ή άρθρο στον αλγόριθμο.

4.2 Κανονικοποίηση των BoW Διανυσμάτων

Τα κείμενα είναι δεδομένο να έχουν δυσανάλογο πλήθος λέξεων. Ένα tweet όπως έχει ήδη αναφερθεί περιορίζεται στους 280 χαρακτήρες ενώ ένα άρθρο είναι πολύ μεγαλύτερο σε έκταση. Σε αυτήν την εργασία βρέθηκε κατά μέσο όρο κάθε tweet να έχει περίπου μόλις 17 λέξεις (αφαιρώντας emoticons, hashtags, mentions κ.α. , οι λέξεις που έχουν αξία είναι πολύ λιγότερες). Ενώ τα άρθρα κατά μέσο όρο βρέθηκαν να έχουν περίπου 381 λέξεις. Όταν θα έρθει η στιγμή να συγκριθούν τα δυσανάλογα BoW διανύσματα θα δοθεί πλεονέκτημα στα μεγαλύτερα, σε πλήθος λέξεων, έγγραφα από ότι αυτά με λιγότερα. Αυτό συμβαίνει όταν μια πρόταση μπορεί να επαναλάβει κάποιες λέξεις παράγοντας έτσι μεγαλύτερο μέγεθος στην διάσταση της λέξης, ενώ μια πιο μικρή πρόταση να έχει μεν περισσότερες λέξεις αλλά το μέγεθος των διαστάσεων να είναι μικρότερο.

Για να αποφευχθεί αυτή η δυσαναλογία στα διανύσματα χρησιμοποιήθηκε η L^2 -Norm κανονικοποίηση. Έστω η πρόταση X μετά από τον BoW αλγόριθμο, x οι τιμές του πίνακα και n το μέγεθος του διανύσματος:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Το l_2 norm ορίζεται από τον αριθμό,

$$l_2norm = \sqrt{\sum_{k=1}^n |x_k|^2}$$

επειδή, τα χ θα είναι πάντα πραγματικοί αριθμοί δεν είναι αναγκαστικό να παίρνουμε κάθε φορά την απόλυτη τιμή τους.

Δηλαδή είναι η τετραγωνική ρίζα του αθροίσματος όλων των τιμών υψωμένων στο τετράγωνο. Για παράδειγμα έστω το διάνυσμα $Y = (1, 0, 3, 2, 0)$ το L²-Norm υπολογίζεται ως εξής:

$$l2norm = \sqrt{1^2 + 0^2 + 3^2 + 2^2 + 0^2} = \sqrt{14} \approx 3.74$$

Έπειτα διαιρείται από κάθε τιμή του διανύσματος η τιμή l2norm.

4.3 Συνημιτονοειδής ομοιότητα μεταξύ διανυσμάτων

Η ομοιότητα μεταξύ διανυσμάτων με χαρακτηριστικά γνωρίσματα (feature set) είναι θεμελιώδες ζήτημα στην κατηγοριοποίηση και ομαδοποίηση των δεδομένων. Η έννοια της ομοιότητας σχετίζεται με την έννοια της απόστασης. Ωστόσο οι έννοιες της ομοιότητας και της απόστασης δεν είναι ακριβώς ίδιες. Για παράδειγμα, η ομοιότητα χρησιμοποιείται για να μετρήσει τα κοινά χαρακτηριστικά μεταξύ δύο στιγμιότυπων, και η απόσταση για να μας δείξει τις διαφορές μεταξύ του. Και πάλι όμως υπάρχει ισχυρή σύνδεση μεταξύ ομοιότητας και απόστασης. Πρώτα μπορούμε να υπολογίσουμε την απόσταση μεταξύ δύο στιγμιότυπων και μετέπειτα να ορίσουμε ένα κατώφλι για να αποφασίζεται αν τα δύο στιγμιότυπα είναι όμοια ή όχι. Η συνέπεια αυτού είναι, τα δύο στιγμιότυπα να είναι πιο όμοια καθώς η απόσταση τους μειώνεται.

Όπως αναφέρθηκε προηγουμένως κάθε κείμενο – πρόταση μετατρέπεται σε διάνυσμα με τόσες διαστάσεις όσες και το πλήθος των λέξεων στο λεξικό. Όσο αυξάνεται ο αριθμός των διαστάσεων στο λεξικό συνεπάγεται περισσότερες καταχωρήσεις στα διανύσματα που δημιουργούνται από το BoW. Με συνέπεια η απόσταση να αυξάνεται ραγδαία όσο αυξάνονται και οι διαστάσεις. Αυτό το πρόβλημα αναφέρεται σαν την κατάρα της διαστατικότητας (curse of dimensionality) το οποίο δεν παρουσιάζεται σε χώρους μικρών διαστάσεων όπως για παράδειγμα ο τρισδιάστατος φυσικός χώρος της καθημερινής εμπειρίας. Τα δεδομένα μεγάλων διαστάσεων παρουσιάζουν τις πληροφορίες αραιά και ανόμοια με πολλούς τρόπους, δηλαδή ο χώρος γίνεται λιγότερο πυκνός και περισσότερο κενός.

Ο τρόπος επιλογής μιας καλά ορισμένης μεθόδους μέτρησης της απόστασης είναι συχνά μεγάλη πρόκληση, λόγω της απουσίας προηγούμενης γνώσης. Η ευκλείδεια απόσταση θεωρείται από τις πιο απλές μεθόδους μέτρησης ομοιότητας στους κλάδους της μηχανικής μάθησης και στην εξόρυξη δεδομένων. Στον Ευκλείδειο χώρο, η απόσταση μεταξύ δύο σημείων είναι το μήκος της γραμμής που τα ενώνει.

Η μέθοδος μέτρησης ομοιότητας «Jaccard» είναι η πιο απλή, συγκρίνει όλες τις λέξεις μεταξύ δύο κειμένων και μετράει τις όμοιες. Όσες περισσότερες υπάρχουν τόσο μεγαλύτερη ομοιότητα υπάρχει μεταξύ των δύο κειμένων. Ωστόσο έχει τους περιορισμούς του, αν για παράδειγμα είχαμε μια ακολουθία λέξεων $A(\alpha, \beta, \gamma, \alpha, \beta, \gamma, \alpha, \beta, \gamma)$ και άλλη μια $B(\alpha, \beta, \gamma)$ η κατά Jaccard ομοιότητα τους είναι ακριβώς ίδια.

Και οι δύο αυτές μέθοδοι απορρίφθηκαν, όταν συγκρίθηκαν εμπειρικά με τα δεδομένα των ειδησεογραφικών και του Twitter. Αντ' αυτού, χρησιμοποιήθηκε η μέθοδος συνημιτονοειδούς ομοιότητας αλλιώς στα αγγλικά «cosine similarity». Αυτή η μέθοδος είναι πιθανώς η πιο δημοφιλής. Η διαδικασία εύρεσης ομοιότητας μεταξύ δύο διανυσμάτων A και B (άρθρο, tweet) είναι η μέτρηση της συνημιτονοειδούς γωνίας μεταξύ αυτών των δύο διανυσμάτων και ορίζεται από τον παρακάτω τύπο:

$$\alpha_{A,B} = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} = \frac{\sum a_i b_i}{\sqrt{\sum a_i^2} \sqrt{\sum b_i^2}}$$

Επειδή έχουμε κανονικοποιημένα διανύσματα, το μόνο που χρειάζεται να υπολογιστεί είναι το εσωτερικό γινόμενο μεταξύ των δύο διανυσμάτων (a,b), το οποίο μας δίνει την ομοιότητα τους. Στους μοντέρνους υπολογιστές αυτή η διαδικασία είναι εξαιρετικά αποδοτική και γρήγορη.

Η μέθοδος συνημιτονοειδούς ομοιότητας γενικά χρησιμοποιείται για την μέτρηση της απόστασης όταν τα μεγέθη των διανυσμάτων δεν έχουν σημασία. Αυτό συμβαίνει όταν έχουμε για παράδειγμα δεδομένα κειμένων παρουσιασμένα με μέτρηση των λέξεων. Μπορούμε να υποθέσουμε όταν μια λέξη (π.χ. ποδήλατο) εμφανίζεται πιο συχνά σε ένα έγγραφο από ένα άλλο έγγραφο, ότι το πρώτο έγγραφο είναι πιο συνυφασμένο με το θέμα της ποδηλασίας. Ωστόσο, μπορεί να είναι η περίπτωση δεδομένων κειμένων που είναι δυσανάλογα σε μήκος λέξεων. Τότε η λέξη «ποδήλατο» πιθανώς να εμφανίστηκε περισσότερο στο πρώτο κείμενο μόνο και μόνο επειδή ήταν μεγαλύτερο σε μήκος από το δεύτερο κείμενο. Όπως φάνηκε στην πράξη, όταν δοκιμάστηκαν και οι τρεις μέθοδοι που προαναφέρθηκαν, η συνημιτονοειδής ομοιότητα διορθώνει τέτοιες ανομοιογένειες, και αυτή η μέθοδος τελικά επιλέχθηκε ως η πιο αποδοτική για το πρόβλημα αυτής της εργασίας.

4.4 Η προσπάθεια να ενσωματωθεί ο αλγόριθμος word2vec

Όπως αναφέρθηκε προηγουμένως το μοντέλο BoW δεν κάνει κάποια συσχέτιση μεταξύ συνώνυμων λέξεων, ούτε παίρνει υπόψιν του τα συμφραζόμενα του κειμένου. Η τεχνική Word2vec ανήκει στην οικογένεια αλγόριθμων word embeddings. Το word embeddings είναι η τεχνική όπου λέξεις εμφυτεύονται σε μια άλλη, με σκοπό να δηλώσουν τον συγγενικό συσχετισμό που έχουν με αυτή την λέξη. Πριν προχωρήσουμε να μιλάμε για το word2vec είναι σημαντικό να γίνει κατανοητό τί είναι αυτά τα εμφυτεύματα.

Το word embeddings είναι μια τεχνική όπου ξεχωριστές λέξεις μετατρέπονται σε αριθμητικές αναπαραστάσεις αυτής της λέξης, δηλαδή σε διανύσματα. Όπου κάθε λέξη χαρτογραφείται σε ένα διάνυσμα, που με την σειρά του εκπαιδεύεται με ένα τρόπο όπως δουλεύει ένα νευρωνικό δίκτυο. Αυτό το διάνυσμα έχει σκοπό να βρει διάφορα χαρακτηριστικά που του ταιριάζουν σε σχέση με τα συνολικά κείμενα. Αυτά τα χαρακτηριστικά μπορούν να συμπεριλαμβάνουν την σημασιολογική συσχέτιση της λέξης, τον ορισμό της, το περιεχόμενο κ.τ.λ. . Με αυτές τις αριθμητικές αναπαραστάσεις είναι δυνατόν να πραγματοποιηθούν συγκρίσεις μεταξύ των λέξεων για να οριστεί η ομοιότητα ή αντίθετα η ανομοιότητα.

Υπάρχουν δύο κεντρικές αρχιτεκτονικές που υλοποιούν την επιτυχία του Word2vec, αυτές είναι οι λεγόμενες skip-gram και CBOW. Η αρχιτεκτονική CBOW (Continuous Bag of Words) για κάθε λέξη των κειμένων (λέξη κέντρο) «βλέπει» και σημειώνει όλες τις διπλανές λέξεις. Το πλήθος των διπλανών λέξεων ονομάζεται παράθυρο. Όσο μεγαλύτερο είναι το παράθυρο τόσες περισσότερες λέξεις θα «βλέπει» και θα σημειώνει ως συγγενικές. Στην τελευταία επανάληψη, όταν θα έχει διεκπεραιωθεί όλο το σύνολο των κειμένων, θα έχουν συγκεντρωθεί όλες οι συσχετίσεις ανάμεσα σε λέξεις. Όσες λέξεις παρατηρήθηκαν περισσότερο ανάμεσα στα παράθυρα, αυτές συσχετίζονται με τις λέξεις «κέντρο». Για παράδειγμα αν έχουμε την πρόταση «Πάνω στο τραπέζι έχει ένα [...] με λουλούδια.», το CBOW μοντέλο θα βρει την λέξη που λείπει στην πρόταση με βάση τα κείμενα που εκπαιδεύτηκε να βρει. Ας πούμε ότι είναι η λέξη «βάζο». Από την άλλη η αρχιτεκτονική skip-gram κάνει ακριβώς το αντίθετο από την CBOW, βρίσκει το περιεχόμενο γύρω από μια λέξη. Δηλαδή αν του δίνουμε την λέξη «βάζο» θα μας έλεγε ότι υπάρχει μεγάλη πιθανότητα το περιεχόμενο αυτής της λέξης να είναι «Πάνω στο τραπέζι έχει ένα βάζο με λουλούδια.». Το

μοντέλο CBOW δουλεύει καλύτερα με λιγότερα δεδομένα και υπολογίζεται πιο γρήγορα. Το skip-gram απαιτεί πολλά δεδομένα και χρειάζεται αρκετούς υπολογιστικούς πόρους και χρόνο για να εκπαιδευτεί (βλ. Tomas Mikolov 2013).

Η αποδοτικότητα του Word2vec αλγορίθμου είναι συνυφασμένη με την ικανότητα του να ομαδοποιεί διανύσματα μεταξύ τους για την εύρεση συσχετισμών στις λέξεις ή στις φράσεις. Έχοντας ένα μεγάλο σύνολο δεδομένων κειμένων, ο αλγόριθμος μπορεί να κάνει ισχυρούς συσχετισμούς σχετικά με το νόημα των λέξεων βασισμένο στον τρόπο που εμφανίζονται οι λέξεις στα κείμενα. Ο αλγόριθμος κατατάσσεται στους αλγορίθμους μηχανικής μάθησης. Με συνέπεια να χρειάζεται ένα αρκετά μεγάλο όγκο δεδομένων για να βγάλει καλά αποτελέσματα. Για άλλες γλώσσες υπάρχουν αρκετά έτοιμα προεκπαιδευμένα σύνολα δεδομένων, ωστόσο στα ελληνικά βρέθηκαν πολύ λίγα. Αποφασίστηκε να φτιαχτεί ένα από την αρχή.

Με την μέθοδο του web scraping από τα ειδησεογραφικά, η οποία είχε υλοποιηθεί προηγουμένως ούτως ή άλλως για τις ανάγκες της εργασίας, συλλέχθηκαν περίπου 2.000 άρθρα από κάθε ειδησεογραφικό, με σύνολο τα περίπου 10.000 άρθρα. Αυτό αντιστοιχεί περίπου σε 400.000 λέξεις. Προεπεξεργάστηκαν με τις μεθόδους που και αυτές είχαν υλοποιηθεί για αυτήν την εργασία. Και με την βοήθεια της βιβλιοθήκης «gensim»¹⁸ εκπαιδεύτηκε ο αλγόριθμος Word2vec. Οι ρυθμίσεις του ήταν σε μέγεθος 100 διαστάσεων, μέγεθος παραθύρου 5, αποκλεισμό λέξεων που εμφανίζονται λιγότερο από 5 φορές στο σύνολο των δεδομένων, Batch Size (ο αριθμός της δειγματοληψίας των λέξεων σε κάθε επανάληψη του αλγορίθμου) ίσο με 10.000 και με την μέθοδο CBOW. Κάθε λέξη αναπαρίσταται σε μία διάσταση του πολυδιάστατου χώρου. Επιλέγοντας μία λέξη ή μία διάσταση μπορούμε να κάνουμε την «ερώτηση» στον εκπαιδευμένο word2vec για το ποιες άλλες διαστάσεις ή λέξεις είναι πιο κοντά σε αυτή. Μερικά από τα αποτελέσματα είναι η λέξη «Ελλάδα» με μεγάλη ομοιότητα βρέθηκαν οι λέξεις Ευρώπη, Τουρκία, Αμερική, Κίνα καθώς και άλλες τοποθεσίες. Με τη λέξη «Μαθηματικά», βρέθηκαν μοντέλα, πειράματα, στατιστικά, επιστημονικά και άλλα συναφή. Δοκιμάστηκε η λέξη «Πρωθυπουργός» και εμφανίστηκαν στα αποτελέσματα ονόματα πρωθυπουργών διάφορων χωρών. Επίσης ελέγχθηκε η λέξη «τρένο» και με μεγάλη επιτυχία έβγαλε διάφορα μέσα μεταφοράς όπως, λεωφορείο, μηχανάκι, αεροπλάνο, αμάξι κ.α. . Το πιο ενδιαφέρον από όλα ήταν η πρόσθεση και αφαίρεση διανυσμάτων στον πολυδιάστατο χώρο. Για παράδειγμα η πρόσθεση των διανυσμάτων με τις λέξεις «βασιλιάς» και «γυναίκα» και η αφαίρεση του διανύσματος της λέξης «άντρας» μας βγάζει το αποτέλεσμα «βασιλίσσα». Ακολουθούν τα παραδείγματα αναλυτικά έτσι όπως εξάχθηκαν από το «gensim» . Σε σχόλιο είναι η εντολή που εκτελέστηκε πάνω στο εκπαιδευμένο σύνολο δεδομένων. Η λέξη positive βάζει θετικό πρόσημο στα διανύσματα, ενώ η λέξη negative βάζει αρνητικό πρόσημο. Στα αποτελέσματα αριστερά είναι η λέξη που βρέθηκε πιο κοντά και δεξιά το σκορ της.

```
# model.wv.most_similar( positive =
['πρωθυπουργός'])
[('υπουργός', 0.7129075527191162),
 ('κοντε', 0.6248044371604919),
 ('κυριακός', 0.6243268847465515),
 ('πρωθυπουργό', 0.6225358843803406),
 ('υφυπουργός', 0.61669921875),
 ('τζόνσον', 0.6069079041481018),
 ('ίδιος', 0.5953857898712158),
 ('καγκελάριος', 0.5928746461868286),
 ('μητσοτακης', 0.5925711393356323),
 ('ακαρ', 0.5912124514579773)]
```

```
# model.wv.most_similar( positive = ['ελλάδα'])
[('ευρωπη', 0.8825979828834534),
 ('τουρκια', 0.7001320719718933),
 ('αμερικη', 0.6934837698936462),
 ('κίνα', 0.6684428453445435),
 ('κυπρο', 0.6665882468223572),
 ('ανθρωποτητα', 0.6554281115531921),
 ('αιγυπτο', 0.6369704008102417),
 ('ιταλια', 0.6368157267570496),
 ('εε', 0.6308513283729553),
 ('αθηνα', 0.6231456995010376)]
```

¹⁸ Η βιβλιοθήκη «gensim» είναι γραμμένη σε python, χρησιμοποιείται από χιλιάδες εταιρείες και είναι λογισμικό ανοικτού κώδικα. Περισσότερες πληροφορίες για την βιβλιοθήκη εδώ: <https://radimrehurek.com/gensim/>

```
#model.wv.most_similar( positive = ['τρενο'])
```

```
[('λεωφορειο', 0.848902702331543),  
( 'μηχανακι', 0.8374761343002319),  
( 'αεροπλανο', 0.8086425065994263),  
( 'φορτηγο', 0.8026442527770996),  
( 'δασος', 0.7763820290565491),  
( 'αμαξι', 0.7678565382957458),  
( 'πεζοδρομιο', 0.7656958699226379),  
( 'ασανσερ', 0.7641466856002808),  
( 'αυτοκινητο', 0.7633139491081238),  
( 'δεντρο', 0.7553384900093079)]
```

```
#model.wv.most_similar( positive =  
['βασίλισσα', 'γυναίκα'], negative = ['άντρας'])
```

```
[('βασίλισσα', 0.6746558547019958),  
( 'αμερικανίδα', 0.6566957235336304),  
( 'φοιτητρια', 0.6164780855178833),  
( 'μαθητρια', 0.6122223138809204),  
( 'σαμάνθα', 0.6080625653266907),  
( 'μαργαριτα', 0.6028112769126892),  
( 'λοπεζ', 0.5984057188034058),  
( 'νοσηλευτρια', 0.59717857837677),  
( 'γιαγια', 0.594476580619812),  
( 'εμιλι', 0.5937211513519287)]
```

```
#model.wv.most_similar( positive =  
['μαθηματικά'])
```

```
[('μοντελα', 0.7302729487419128),  
( 'θεραπευτικά', 0.698531448841095),  
( 'πειράματα', 0.6779818534851074),  
( 'στατιστικά', 0.6627822518348694),  
( 'καρπουζια', 0.6593343019485474),  
( 'ιστορικά', 0.654633641242981),  
( 'προκαταρκτικά', 0.6441912055015564),  
( 'επιστημονικά', 0.6434250473976135),  
( 'λαχανικά', 0.6343380212783813),  
( 'συνοψίζουν', 0.6312335729598999)]
```

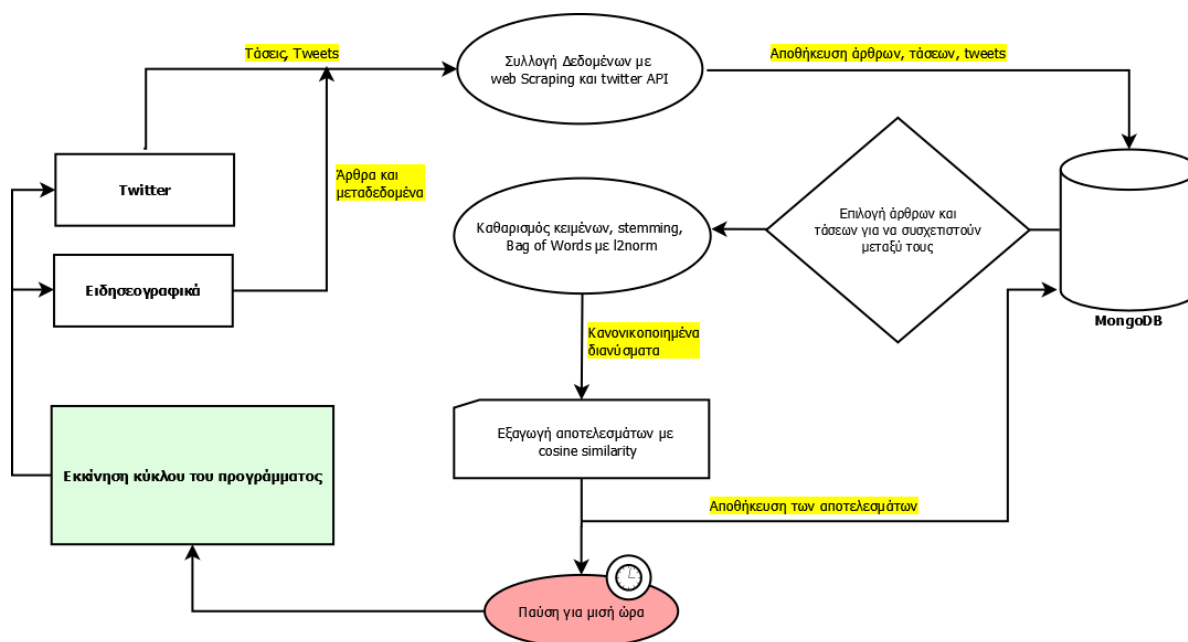
```
#model.wv.most_similar( positive =  
['μητέρα', 'άντρας'], negative = ['γυναίκα'])
```

```
[('πατερας', 0.8180052042007446),  
( 'παππους', 0.7921249866485596),  
( 'αδελφος', 0.7875447273254395),  
( 'μπαμπας', 0.7779601216316223),  
( 'αδερφος', 0.7743542194366455),  
( 'γαμπρος', 0.768739640712738),  
( 'εγγονος', 0.7624061107635498),  
( 'γιος', 0.7613533139228821),  
( 'θειος', 0.7424112558364868),  
( 'ερωτας', 0.7394472360610962)]
```

Συγκρίθηκαν τα αποτελέσματα με τα κείμενα με embedding και με αυτά χωρίς embedding και διαπιστώθηκε ότι δεν υπήρχε σημαντική διαφορά. Για την ακρίβεια μετρήθηκε 2% αύξηση στο βαθμό σχετικότητας μεταξύ δύο κειμένων. Η δοκιμή έγινε σε προϋπάρχων δεδομένα άρθρων και tweets για την ακρίβεια 500 άρθρα συγκρίθηκαν με 2.000 tweets. Αυτό σημαίνει 10.000 επαναλήψεις και πάρα πολύ χρόνο. Με το μηχάνημα που αναπτύχθηκε το πρόγραμμα πραγματοποιήθηκε η όλη αναζήτηση μέσα 8.5 ώρες, ενώ δίχως embedding έκανε μόλις μία ώρα και 40 λεπτά. Δηλαδή η μέθοδος embedding επιβράδυνε το συνολικό χρόνο επεξεργασίας κατά 5 φορές παραπάνω απ'ότι δίχως embedding. Εν κατακλείδι, εγκαταλείφθηκε η ενσωμάτωση του αλγορίθμου word2vec στο κύριο σώμα του προγράμματος, ωστόσο αποκομίστηκαν πολλά από αυτή την εμπειρία.

4.5 Σφαιρική εικόνα όλης της διαδικασίας της επεξεργασίας δεδομένων

Για την κατανόηση της συνολικής λειτουργίας του αλγόριθμου θα περιγραφεί η όλη διαδικασία με την σειρά που εκτελείται. Ο αλγόριθμος δεν σταματάει να εκτελείται από μόνος του ποτέ, εκτελείται αυτόματα σταθερά κάθε μισή ώρα συν τον μεταβλητό χρόνο που παίρνει να διεκπεραιωθεί ένας κύκλος. Ο κύκλος ορίζεται από την στιγμή που ξεκινάει να εκτελείται ο αλγόριθμος, μέχρι που θα τελειώσει.



Εικόνα 4 Διάγραμμα ροής δεδομένων για την όλη διαδικασία που ακολουθεί το πρόγραμμα στην συλλογή των δεδομένων, επεξεργασίας τους και εξαγωγής αποτελεσμάτων.

Αρχικά κατεβαίνουν οι τάσεις από το Twitter και κρατούνται οι πρώτες πέντε στην ΒΔ. Για αυτές τις πρώτες πέντε τάσεις κατεβαίνουν τα αντίστοιχα tweets και αποθηκεύονται στην ΒΔ. Ο αριθμός τους έχει περιοριστεί σε 400 ξεχωριστά tweets, γιατί βρέθηκε ότι εξυπηρετούν τις ανάγκες του εγχειρήματος. Αυτό με σκοπό να βελτιωθεί ο χρόνος ολοκλήρωσης του κύκλου. Επειδή ο αλγόριθμος τρέχει συνεχώς μπορεί να ανιχνεύσει περίπου την στιγμή που μία τάση εμφανίστηκε στον πίνακα των τάσεων του twitter. Όταν πρωτοεμφανίζεται μια τάση είναι πιθανό να έχει πολύ λιγότερα από 400 tweets διαθέσιμα, διότι δεν έχουν γραφτεί ακόμα. Οπότε σε κάθε κύκλο αν υπάρχουν λιγότερα από 400 tweets διαθέσιμα στην ΒΔ, τότε θα γίνει η προσπάθεια μέσω του Twitter API να κατεβούν περισσότερα κάθε φορά αντικαθιστώντας τα παλιά.

Στην συνέχεια κατεβαίνουν τα άρθρα από τα ειδησεογραφικά μέσω του web scraping. Κάθε φορά που συλλέγονται άρθρα αποθηκεύεται η ημερομηνία που κατέβηκαν από τα ειδησεογραφικά. Αυτό με σκοπό σε κάθε επόμενο κύκλο να κατεβαίνουν τα άρθρα τόσο πίσω ημερομηνιακά σε σχέση με την προηγούμενη φορά που κατέβηκαν. Κάθε φορά που αποθηκεύεται άρθρο ή tweet ελέγχεται αν υπάρχει ήδη στην ΒΔ για την μη αποθήκευσης του ούτως ώστε να μην υπάρχουν διπλότυπα.

Αφού έχουν αποθηκευτεί τα άρθρα και τα tweets, αυτό που γίνεται είναι να συλλεχθούν από την βάση δεδομένων αν υπάρχουν παρελθοντικά διαθέσιμα αποτελέσματα για κάποιες από τις πρώτες τάσεις. Αν υπάρχουν τότε ελέγχονται μόνο τα καινούργια άρθρα που συλλέχθηκαν, μήπως υπάρχει κάποια καλύτερη συσχέτιση με αυτά. Αν δεν υπάρχει διαθέσιμο κάποιο αποτέλεσμα, τότε σημαίνει ότι η τάση είναι καινούργια για το πρόγραμμα και δεν έχει ελεγχθεί καμία φορά κάποια συσχέτιση με τα άρθρα. Από τον αρχικό πειραματισμό, βρέθηκε ότι το πλήθος των

άρθρων που αξίζει να συσχετιστεί είναι αντίστοιχο με όλα τα άρθρα έως τρεις μέρες πίσω. Συνήθως στις τάσεις του Twitter αναρτούνται γεγονότα που έχουν συμβεί πρόσφατα, οπότε έχει νόημα να ψάχνεται κάποια συσχέτιση με πρόσφατα άρθρα και όχι παλιά.

Εφόσον έχουν προσδιοριστεί όλα τα άρθρα και τα tweets που πρέπει να ελεγχθούν για την συσχέτιση τους, τα δεδομένα οδηγούνται στην επόμενη φάση του καθαρισμού τους. Αυτό περιληπτικά είναι η αφαίρεση missing data, μετατροπή δεδομένων στην σωστή μορφή, αφαίρεση stopwords και αφαίρεση ειδικών χαρακτήρων. Στην συνέχεια πραγματοποιείται η διαδικασία stemming πάνω στις λέξεις, η οποία είναι πολύ σημαντική για την συνολική απόδοση του αλγορίθμου ομοιότητας. Ακολουθεί η διανυσματικοποίηση των κειμένων με την τεχνική Bag of Words και την κανονικοποίηση των διανυσμάτων με την l2norm. Από την στιγμή που έχουν ολοκληρωθεί αυτές οι διεργασίες τα δεδομένα είναι έτοιμα να περάσουν από την διαδικασία σύγκρισης τους με την μέθοδο cosine similarity.

Κάθε άρθρο συγκρίνεται με κάθε tweet, η πολυπλοκότητα αυτής της διαδικασίας είναι $O(n^2)$ και παίρνει μακράν τον μεγαλύτερο χρόνο να υπολογιστεί σε σχέση με τις άλλες διεργασίες. Διότι αν για παράδειγμα έχουμε 200 άρθρα που πρέπει να βρεθεί ο βαθμός συσχέτισης τους με 2000 tweets, αυτό σημαίνει ότι κάθε άρθρο θα ελεγχθεί με όλα τα 2000 tweets ξεχωριστά. Αν πολλαπλασιάσουμε το πλήθος των tweets και των άρθρων τότε θα βρεθεί ότι θα πραγματοποιηθούν 400.000 επαναλήψεις. Για όλα αυτά τα αποτελέσματα κρατούνται όλες οι μετρήσεις της συσχέτισης των tweets και των άρθρων μέσα σε ένα έγγραφο του collection μαζί με όλα τα μεταδεδομένα που τους αντιστοιχούν. Στο τέλος θα υπάρχουν, αν κρατήσουμε τους αριθμούς του παραπάνω παραδείγματος, 400.000 έγγραφα. Από αυτά θα κρατηθούν τα καλύτερα από κάθε ειδησεογραφικό και για κάθε τάση. Τα ειδησεογραφικά που επιλέχθηκαν είναι πέντε και οι τάσεις που ελέγχονται κάθε φορά είναι επίσης οι πρώτες πέντε, οπότε από όλα τα 400.000 έγγραφα στο τέλος κρατούνται μόνο τα καλύτερα 25. Αυτό ορίζεται από την μεγαλύτερη βαθμολογία που βρέθηκε με την μέθοδο cosine similarity.

Η βαθμολογία υπολογίζεται με σταθερό τρόπο κάθε φορά. Η κάθε τάση αποτελείται από το μέγιστο στα 400 tweets, οπότε κάθε τάση έχει την δικιά της ομάδα tweets που την αποτελούν. Έτσι κάθε ομάδα tweets συσχετίζεται με κάθε άρθρο ξεχωριστά. Στο τέλος οι βαθμολογίες που συγκεντρώθηκαν για κάθε tweet της ομάδας αθροίζονται και διαιρούνται με τον πλήθος των tweets της ομάδας. Οπότε, έχουμε τον μέσο όρο της βαθμολογίας όλων των tweets της ομάδας με κάποιο άρθρο.

Αφού έχει τελειώσει η διαδικασία της συσχέτισης ομοιότητας άρθρων και τάσεων, ελέγχεται στην ΒΔ μήπως υπάρχει ήδη καταχωρημένη παρελθοντική συσχέτιση για κάποια τάση. Αν υπάρχει διαθέσιμη σημαίνει ότι η τάση έχει ξαναπεράσει την όλη διαδικασία και τα νέα αποτελέσματα θα αποθηκευτούν στην ΒΔ μόνο αν έχουν καλύτερη βαθμολογία από τα παλιά. Με άλλα λόγια η καλύτερη βαθμολογία συνεπάγεται καλύτερη ομοιότητα μεταξύ τάσης και άρθρου, οπότε όσο καλύτερος είναι ο βαθμός της ομοιότητας τόσο καλύτερα θα περιγράψει το άρθρο το περιεχόμενο των tweets της τάσης.

Μέχρι στιγμής έχει τελειώσει η κεντρική λειτουργία και δραστηριότητα του κύκλου του αλγορίθμου, μένουν να γίνουν κάποιες μικρές αλλαγές στην ΒΔ. Οι αλλαγές αυτές είναι απαραίτητες μόνο για την εξυπηρέτηση του client και δεν επηρεάζουν τις διεργασίες που πραγματοποιούνται για την επεξεργασία των δεδομένων και την εξόρυξη γνώσης. Αυτές οι αλλαγές είναι η αποθήκευση κάποιων στατιστικών για το μέγεθος της βάσης δεδομένων, της αποθήκευση των πρώτων σε δημοτικότητα tweets της τάσης και την συνολική βαθμολογία που έχει συλλέξει κάθε ειδησεογραφικό.

4.6 Διαχείριση των σφαλμάτων που προκύπτουν κατά την διάρκεια εκτέλεσης του προγράμματος

Κάθε πρόγραμμα είναι επιρρεπές σε σφάλματα που μπορούν να προκύψουν κατά την εκτέλεση του κώδικα. Έτσι και σε αυτό το πρόγραμμα υπάρχουν σφάλματα που μπορούν να προκύψουν και να τερματίσουν την εκτέλεση. Ευτυχώς πολλά από τα σφάλματα που εμφανίζονται μπορούν να προβλεφθούν. Η rython έχει ενσωματωμένο

μηχανισμό διαθέσιμο για την διαχείριση των σφαλμάτων και είναι στο χέρι του σχεδιαστή να τον υλοποιήσει. Ο κώδικας που μπορεί να εμφανίσει κάποιο σφάλμα ενθυλακώνεται μέσα στην ειδική σύνταξη της python (**try: .. except: ..**). Ένα τέτοιο παράδειγμα από τα πολλά, είναι η διαίρεση με το μηδέν που δεν ορίζεται στα μαθηματικά και αν προσπαθήσουμε να κάνουμε κάτι τέτοιο στην python θα τερματιστεί το πρόγραμμα με την ένδειξη σφάλματος **ZeroDivisionError**.

```
try:
    x = 5/0
except ZeroDivisionError as err:
    # Το σφάλμα ανιχνεύεται και εδώ μέσα
    # αντιμετωπίζεται όπως επιθυμεί ο σχεδιαστής
```

Στο μπλοκ **try** μπαίνει ο κώδικας που πρέπει να εκτελεστεί, στην συνέχεια στο μπλοκ **except** ορίζεται το σφάλμα που μπορεί να προκύψει. Εκεί μέσα ο σχεδιαστής πρέπει να ορίσει τί θα συμβεί, άμα δεν οριστεί τίποτα, η μεταβλητή *x* δεν θα αποθηκευτεί στην μνήμη και θα είναι σαν να μην υπήρξε ποτέ, από εκεί και πέρα η εκτέλεση του κώδικα συνεχίζεται κανονικά.

Το πρόγραμμα που έχει δημιουργηθεί σε αυτό το εγχείρημα πρέπει να εκτελείται συνεχώς, να ανιχνεύει τυχόν σφάλματα και να τα αντιμετωπίζει. Το πρώτο πράγμα που μπορεί να πάει λάθος είναι τα ειδησεογραφικά να μην είναι προσβάσιμα, αυτό είτε σημαίνει ότι το πρόγραμμα για κάποιο λόγο δεν έχει πρόσβαση στο διαδίκτυο, είτε τα ειδησεογραφικά είναι ανενεργά για τους λόγους τους. Σε κάθε κύκλο του προγράμματος γίνεται η προσπάθεια σύνδεσης με τα ειδησεογραφικά αν επιτύχει τότε το πρόγραμμα συνεχίζει κανονικά, διαφορετικά αν αποτύχει σημαίνει ότι έχει υπάρξει το σφάλμα **ConnectionError** ή **Timeout**. Το **ConnectionError** είναι γενικό σφάλμα όταν αποτυγχάνει ένα HTTP request, το **Timeout** σφάλμα ορίζει την αποτυχία σύνδεσης μέσα σε ένα ορισμένο χρονικό περιθώριο. Αν πραγματοποιηθεί κάποια σφάλμα από αυτά τα δύο τότε δεν έχει νόημα να συνεχίσει το πρόγραμμα καθώς δεν μπορούν να υπάρξουν καινούργια άρθρα. Κάθε φορά που δεν μπορεί να εδραιωθεί σύνδεση με τα ειδησεογραφικά ο κώδικας περιμένει πέντε δευτερόλεπτα, μήπως το πρόβλημα ήταν τετριμμένο και με τις επόμενες προσπάθειες μπορεί να λυθεί αυτόματα.

Άλλα προβλήματα που μπορούν να προκύψουν είναι αυτά με την βάση δεδομένων, όπου δεν μπορεί να εδραιωθεί σύνδεση με την **mongoDB**. Αυτό μπορεί να συμβεί όταν η **mongoDB** δυσλειτουργεί λόγω περιορισμένων υπολογιστικών πόρων, όταν δεν έχει γίνει εκκίνηση ή όταν οι θύρες είναι λανθασμένες. Στην περίπτωση αυτή πάλι το πρόγραμμα δεν μπορεί να κάνει πολλά από μόνο του οπότε περιμένει το πρόβλημα να λυθεί από εξωτερικό παράγοντα. Μοιραίο σφάλμα επίσης είναι να ζητηθούν έγγραφα από την ΒΔ εκεί όπου δεν υπάρχουν ή να αποθηκευτούν αρχεία τα οποία είναι σε ακατάλληλη μορφή.

Όλα τα υπόλοιπα σφάλματα που μπορεί να προκύψουν από όλες τις διεργασίες συνολικά καταλήγουν εκεί από όπου πρωτοεκτελέστηκαν. Το πρόγραμμα ξεκινάει καλώντας μια συνάρτηση, αυτή η συνάρτηση καλεί με την σειρά της άλλες συναρτήσεις και ούτω καθεξής. Είναι ακριβώς η ίδια λογική με την δομή ενός δέντρου, υπάρχει ο αρχικός κόμβος όπου συνδέεται με τα παιδιά κόμβους. Όποτε σε κάποιο παιδί κόμβο προκύψει κάποιο σφάλμα, το σφάλμα καταλήγει πάντοτε στον αρχικό κόμβο (*parent node*). Ο αρχικός κόμβος σχεδιάστηκε για να μπορεί να εντοπίσει όλα τα σφάλματα που μπορούν να υπάρξουν. Αν καταλήξει σφάλμα στον αρχικό κόμβο, τότε αυτός σταματάει την φυσική ροή του προγράμματος και περιμένει για 15 λεπτά, μετά από την αναμονή το πρόγραμμα ξεκινάει από την αρχή. Τα πιο συνηθισμένα σφάλματα που συνέβησαν κατά την ανάπτυξη του προγράμματος ήταν τα HTTP requests σφάλματα γενικότερα και τα σφάλματα κατά την διάρκεια του **scraping** όπου υπήρξε η αλλαγή μιας ιστοσελίδας από τα ειδησεογραφικά που επιλέχθηκαν.

5. Ο διακομιστής NodeJS - express

Ο διακομιστής ιστού ή αλλιώς στα αγγλικά πιο γνωστός ο όρος web server είναι λογισμικό και υλικό υπολογιστή που χρησιμοποιεί το HTTP πρωτόκολλο για να επικοινωνεί με τον client ή αλλιώς με τον διαδικτυακό χρήστη που κάνει αίτημα για πρόσβαση στην ιστοσελίδα. Η κεντρική δουλειά ενός διακομιστή ιστού είναι να προβάλλει το περιεχόμενο της ιστοσελίδας μέσω της αποθήκευσης, επεξεργασίας και παράδοσης της ιστοσελίδας στους χρήστες της. Το υλικό του διακομιστή ιστού είναι συνδεδεμένο με το διαδίκτυο και επιτρέπει την ανταλλαγή δεδομένων μεταξύ των χρηστών και του ίδιου του διακομιστή. Στην ουσία είναι ένας κοινός υπολογιστής όπως τον ξέρουμε με τροποποιημένες ρυθμίσεις στο λογισμικό του και συνήθως υψηλών προδιαγραφών μηχανήματα ώστε να μπορούν να ανταπεξέρχονται στα αιτήματα των πολλαπλών χρηστών.

5.1 Τρόπος λειτουργίας ενός διακομιστή ιστού

Ο διακομιστής ιστού (web server) είναι προσβάσιμος μέσω του domain ονόματος της ιστοσελίδας και εξασφαλίζει την αποστολή του περιεχομένου της ιστοσελίδας στους χρήστες. Το λογισμικό αποτελείται από πολλά στοιχεία, μεταξύ τους τουλάχιστον ένας διακομιστής HTTP. Ο διακομιστής HTTP κατανοεί το πρωτόκολλο HTTP και τις διευθύνσεις URL. Στο υλικό κομμάτι, ο υπολογιστής έχει αποθηκευμένα τα απαραίτητα αρχεία που σχετίζονται με την προβολή της ιστοσελίδας, όπως για παράδειγμα HTTP, CSS, JavaScript κ.α. . Όταν ένας φυλλομετρητής όπως το Firefox ή το Google Chrome, προσπαθεί να προβάλει μια ιστοσελίδα, τότε στέλνει ένα αίτημα στον διακομιστή ιστού, αν γίνει αποδεκτό, τότε αυτός επιστρέφει το περιεχόμενο της ιστοσελίδας που ζητήθηκε μέσω του HTTP.

Υπάρχουν δύο είδη web server καθώς και ιστοσελίδων, αυτές είναι οι δυναμικές και οι στατικές. Οι στατικοί servers παρουσιάζουν το περιεχόμενο που προβάλλεται να είναι πάντα το ίδιο έτσι όπως έχει αποθηκευτεί στον server. Από την άλλη ένας δυναμικός server παράγει την στιγμή που ζητείται η ιστοσελίδα από τον χρήστη, σύμφωνα με τα δεδομένα που είναι αποθηκευμένα στην ΒΔ και την επεξεργασία που είναι αναγκαία. Ένα τέτοιο παράδειγμα λογισμικού server είναι η εφαρμογή NodeJS, αν και η NodeJS μπορεί να κάνει και άλλα πράγματα όπως άλλες γενικές γλώσσες προγραμματισμού. Παραδοσιακά σε μια στατική σελίδα, για να αλλάξουν τα δεδομένα που προβάλλονται πρέπει να γίνει χειροκίνητη ανανέωση της ιστοσελίδας από τον σχεδιαστή και αυτό φυσικά πραγματοποιείται σε μη πραγματικό χρόνο. Η NodeJS είναι χρήσιμη για την ανάπτυξη εφαρμογών που χρειάζονται συνεχή σύνδεση μεταξύ server και client και συνήθως χρησιμοποιείται για την προβολή δεδομένων σε πραγματικό χρόνο. Η γλώσσα python εκτελεί τον κώδικα της από πάνω προς τα κάτω έτσι όπως είναι γραμμένος, όταν τελειώσει μια εργασία πηγαίνει στην επόμενη, με την NodeJS αυτό δεν ισχύει. Οι εφαρμογές που βασίζονται στην NodeJS λειτουργούν ασύγχρονα και με βάση γεγονότων (event listeners) που μπορεί να συμβούν. Εφαρμόζεται η ίδια λογική στην σύνταξη καθώς και στις λειτουργίες της JavaScript όπου και η ίδια λειτουργεί ασύγχρονα και με βάση γεγονότων. Και αυτό έχει λογική βάση αν αναλογιστούμε πόση σημασία έχει ο χρόνος φόρτωσης μια ιστοσελίδας. Τις περισσότερες φορές που βλέπουμε μια ιστοσελίδα, ο ίδιος ο ιστότοπος μπορεί να μην έχει φορτώσει τελειώς να κατεβάζει όλα τα απαραίτητα αρχεία που την αποτελούν, ωστόσο ο φυλλομετρητής να μας δείχνει κάποια από αυτά. Αυτό οφείλετε στην ασύγχρονη φύση της JavaScript, η NodeJS έχει υιοθετήσει την ίδια λογική.

5.2 REST API ή αλλιώς ο τρόπος επικοινωνίας μεταξύ server και client

Ο όρος REST (Representational State Transfer) είναι μία αρχιτεκτονική για την αποστολή δεδομένων σε όλο το διαδίκτυο. Έκανε την εμφάνισή της με την δημοσίευση του άρθρου «Architectural Styles and the Design of Network-based Software Architecture»¹⁹ την χρονιά 2000 από τον Roy Fielding, ενός Αμερικανού επιστήμονα των υπολογιστών (Fernando Doglio, 2015). Είναι περισσότερο αρχιτεκτονικό στυλ παρά κάποιο πρότυπο, με την έννοια ότι δεν είναι υποχρεωτικό να υπάρχει αυστηρή συμμόρφωση σε κανόνες. Για να υπάρξει επικοινωνία μεταξύ δύο μηχανημάτων και να την ονομάζουμε REST, πρέπει να τηρούνται κάποια κριτήρια.

Αρχική προϋπόθεση είναι να χρησιμοποιείται το πρωτόκολλο HTTP από όλα τα μηχανήματα που μεταφέρουν δεδομένα μεταξύ τους. Επιπλέον πρέπει να είναι stateless, αυτό σημαίνει ότι δεν υπάρχει γνώση προηγούμενης πληροφορίας μεταξύ των αιτημάτων επικοινωνίας. Κάθε αίτημα πρέπει να είναι απομονωμένο από τα υπόλοιπα και να λειτουργεί ανεξάρτητα, για παράδειγμα μία αναζήτηση σε μία μηχανή αναζήτησης λειτουργεί έτσι, εισάγεται το κλειδί της αναζήτησης και επιστρέφονται τα αποτελέσματα. Αν αποτύχει το αίτημα της αναζήτησης, γίνεται καινούργιο και ούτω καθεξής. Άλλο κριτήριο είναι τα δεδομένα μεταξύ των δύο μηχανημάτων να μπορούν να ανταλλάζουν δεδομένα με τέτοια μορφή που να είναι αναγνώσιμη από κοινού.

Ο όρος API (Application programming interface) είναι ένα σύνολο από ορισμούς και πρωτόκολλα σχεδιασμένα να παρέχουν την γλώσσα επικοινωνίας μεταξύ του πάροχου πληροφορίας και παραλήπτη πληροφορίας. Για παράδειγμα ένα μικρό μέρος API μίας μετεωρολογική υπηρεσία θα μπορούσε να ήταν ο χρήστης να δίνει το όνομα μίας πόλης και για απάντηση να παίρνει την θερμοκρασία. Οπότε ένα API που ακολουθεί την αρχιτεκτονική REST ονομάζεται REST API. Αυτή η αρχιτεκτονική είναι πιο γνωστή σαν την μεταφορά δεδομένων μέσω HTTP και την αποστολή και λήψη δεδομένων σε μορφή JSON (JavaScript Object Notation). Η JSON είναι ένα πρότυπο δομής δεδομένων η οποία είναι ανεξάρτητη από κάποια γλώσσα προγραμματισμού και είναι δυνατόν με εύκολο τρόπο να αναγνωριστούν τα δεδομένα από οποιαδήποτε γλώσσα προγραμματισμού.

5.3 REST API με την express.js βιβλιοθήκη

Για την εύκολη δημιουργία REST API χρησιμοποιήθηκε η βιβλιοθήκη «**express**»²⁰, μία μινιμαλιστική και γρήγορη βιβλιοθήκη φτιαγμένη αποκλειστικά για την NodeJS. Η συγκεκριμένη βιβλιοθήκη παρέχει το routing σε μια σελίδα. Το routing ή αλλιώς η δρομολόγηση είναι η διαδικασία επιλογής διαδρομής σε ένα δίκτυο ή σε πολλά. Οι χρήστες επιλέγουν μία διαδρομή μέσω ενός HTTP request και η express απαντάει σε αυτούς ανάλογα με δεδομένα ή ανατροφοδότηση. Η διαδρομή αποτελείται από URLs (Uniform Resource Locator), δηλαδή μία συμβολοσειρά με συγκεκριμένη δομή που την καταλαβαίνουν εύκολα όλα τα μηχανήματα των χρηστών ώστε να μπορούν να στείλουν κάποια αίτηση. Η εφαρμογή «ακούει» (listens) για αιτήματα των χρηστών οποιαδήποτε στιγμή και αν τα κάνουν και αν υπάρχει ταίριασμα της διαδρομής (URL) στην αίτηση από πλευράς server και client τότε εκτελούνται οι μέθοδοι αυτής της διεργασίας και επιστρέφεται το επιθυμητό αποτέλεσμα.

Ο συνδυασμός του τύπου του αιτήματος HTTP (GET, POST, DELETE, PUT) και η συμβολοσειρά URL ορίζουν τις διαδρομές του REST API. Η δομή της συμβολοσειράς είναι παρόμοια με το σύστημα διαδρομής των UNIX, όπου κάθε συμβολοσειρά είναι μια διαδρομή ανάμεσα από το σύμβολο (/) που διαχωρίζει τις διαφορετικές διαδρομές. Τα ονόματα των διαδρομών έχουν νόημα να ονομάζονται σχετικά με την λειτουργία που παρέχουν ώστε να είναι πιο ανθρώπινως κατανοητά. Για παράδειγμα σε μια υπηρεσία μετεωρολογικού σταθμού REST API θα είχε νόημα να υπάρχει η διαδρομή (URL) « /<πόλη>/temperature», όπου «πόλη» εισάγεται μία από τις πόλεις που είναι διαθέσιμες. Έτσι η διαδρομή « /athens/temperature» είναι λογικό να δίνει την θερμοκρασία που υπάρχει αυτήν την

¹⁹ Το άρθρο του συγγραφέα που πρωτοέγραψε για το REST μπορεί να βρεθεί εδώ:

https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

²⁰ Περισσότερες πληροφορίες για την βιβλιοθήκη express εδώ: <https://expressjs.com/>

στιγμή στην πόλη Αθήνα. Αφού η συγκεκριμένη διαδρομή έχει σχεδιαστεί να επιστρέφει μία τιμή (την θερμοκρασία) ταιριάζει η μέθοδος HTTP GET. Συνδυαστικά για να δημιουργηθεί η συγκεκριμένη διαδρομή στην express θα γράφαμε:

```
app.get('/athens/temperature', (req, res) => {  
  //Η μέθοδος get_temp(), ας πούμε ότι επιστρέφει την θερμοκρασία μίας πόλης  
  res.send(get_temp('athens'));  
});
```

Ας αποδημήσουμε βήμα προς βήμα τί σημαίνει όλη αυτή η μέθοδος. Αρχικά η λέξη app είναι η αρχικοποίηση της βιβλιοθήκης express, εκεί μέσα υπάρχουν όλοι οι παράμετροι και οι μέθοδοι που ορίζουν όλη την βιβλιοθήκη. Μία από αυτές τις μεθόδους είναι η get. Οι σχεδιαστές της βιβλιοθήκης express την ορίζουν σαν μία συνάρτηση routing HTTP GET αιτήματος με συγκεκριμένη διαδρομή και ορισμένης callback μέθοδο²¹:

```
app.get(path, callback [, callback ...])
```

Η get δηλαδή παίρνει υποχρεωτικά δύο παραμέτρους, πρώτη είναι η διαδρομή και η δεύτερη είναι η callback μέθοδος. Η callback μέθοδος είναι χαρακτηριστικό της γλώσσας JavaScript και είναι ο ορισμός μιας συνάρτησης που εισχωρεί σαν παράμετρος σε μία άλλη συνάρτηση. Οι callback συναρτήσεις συνήθως είναι ανώνυμες συναρτήσεις ομοίως με τις lambda συναρτήσεις της python και της JAVA, αυτές είναι ίδιες με τις κανονικές γνωστές συναρτήσεις μόνο που δεν ορίζεται αναγνωριστικό όνομα και γι' αυτό ονομάζονται ανώνυμες. Ένα παράδειγμα είναι όταν μία ανώνυμη συνάρτηση εκχωρείται σε μία μεταβλητή και η σύνταξη της είναι έτσι:

```
var welcome = function () {  
  //Εκτύπωσε το παρακάτω μήνυμα  
  console.log("Γεια σας!!!");  
};
```

Αυτός είναι έγκυρος JavaScript κώδικας, συνήθως όμως χρησιμοποιείται άλλη σύνταξη για την ανώνυμη συνάρτηση. Από τον 2015 και μετά, το ECMA 2015 (παρ. 14.2)²² πρότυπο δημιούργησε για την JavaScript την σύνταξη arrow function. Σε αυτό παρακάμπτεται η δεσμευμένη λέξη function και προσθέτετε το σύμβολο (=>). Έτσι αντίστοιχα ο παραπάνω κώδικας γράφεται έτσι:

```
var welcome = () => {  
  //Εκτύπωσε το παρακάτω μήνυμα  
  console.log("Γεια σας!!!");  
};
```

Συνεχίζουμε με την επεξήγηση του παραδείγματος του μετεωρολογικού σταθμού. Το req και res που εισάγονται σαν παράμετροι στην ανώνυμη συνάρτηση είναι συντμήσεις από τις λέξεις request, response αντίστοιχα, δηλαδή αίτημα και ανταπόκριση. Στην παράμετρο req υπάρχουν ορισμένες ιδιότητες που μπορεί να αποστείλει με αίτημα το client, ενώ το res έχει όλες τις ιδιότητες που θα απαντήσει ο server στον client. Επειδή το αίτημα εδώ είναι GET και το client δεν έδωσε περισσότερες πληροφορίες, περιμένει μόνο απάντηση. Οπότε χρησιμοποιείται η μέθοδος «sent» έτσι όπως ορίζεται στην res παράμετρο, για να αποστείλει την απάντηση πίσω στον client.

²¹ Ο ορισμός της get μεθόδου, και άλλες πληροφορίες <http://expressjs.com/en/5x/api.html#app.get.method>

²² Το standard ECMA-262 όπως δημοσιεύθηκε η έκτη έκδοση τον Ιούνιο του 2015: <https://262.ecma-international.org/6.0/>

Μέσα στην διαδρομή μπορούν να οριστούν παράμετροι ώστε να εξηγήσουν στον server ακριβώς τί ζητάνε. Οι παράμετροι ορίζονται από τον server και αποτελούνται από συμβολοσειρές, ένας τύπος είναι όπως περιεγράφηκε στην διαδρομή με το μετεωρολογικό παράδειγμα (`/<πόλη>/temperature`). Εδώ η παράμετρος «πόλη» είναι υποχρεωτική και ανήκει στην ίδια την διαδρομή. Ο άλλος τύπος παραμέτρου είναι οι ακολουθούμενες μεταβλητές μετά από το τέλος της διαδρομής. Στις μεταβλητές παραμέτρους κάθε μεταβλητή ορίζεται η μία μετά την άλλη και χωρίζονται από το σύμβολο (**&**). Για παράδειγμα οι μεταβλητές **unit** (μονάδα μέτρησης θερμοκρασίας) και **decimal_places** (πόσα δεκαδικά ψηφία δηλαδή πόσο ακριβής θερμοκρασία να επιστρέφεται):

```
/<πόλη>/temperature&unit=celsius&decimal_places=2
```

Σε αυτό το παράδειγμα η μονάδα μέτρησης ορίζεται σε βαθμούς κελσίου και τα δεκαδικά σε δύο. Όπως φαίνεται η μεταβλητή παράμετρος χωρίζεται με το σύμβολο εκχώρησης (=), αριστερά του είναι το όνομα της μεταβλητής και δεξιά η τιμή του. Ο client θα πρέπει να ξέρει εκ των προτέρων όλες τις επιτρεπτές και αναγνωρίσιμες μεταβλητές παραμέτρους καθώς και τις διαθέσιμες τιμές τους που μπορούν να εισχωρήσουν σε μία διαδρομή.

5.4 Το REST API του εγχειρήματος

Στην διαδρομή «/» του εγχειρήματος που έχει οριστεί και σαν αρχική σελίδα της διεπαφής επιστρέφεται όλη η ιστοσελίδα (HTML, CSS, JS, εικόνες, εικονίδια κ.α.). Οπότε αν για παράδειγμα η IP διεύθυνση του server που φιλοξενεί την διεπαφή είναι η 8.7.6.1 τότε η διαδρομή **8.7.6.1/** επιστρέφει την ιστοσελίδα. Η ιστοσελίδα ωστόσο είναι δυναμική και έχει περισσότερες ανάγκες από ότι να δίνεται μόνο η διεπαφή. Πρέπει να αποστέλλονται με κάποιο τρόπο τα αποτελέσματα της επεξεργασίας των δεδομένων από τα ειδησεογραφικά και twitter. Καθώς και να δέχεται ο server αιτήματα για αναζήτηση αποτελεσμάτων αποθηκευμένα στην ΒΔ με βάση τα ερωτήματα του client. Έχοντας υπόψιν τα παραπάνω είναι προφανές ότι τα απαραίτητα αιτήματα στον server είναι τύπου GET και POST.

Αρχικά είναι απαραίτητο να προσφέρονται τα δεδομένα με όλα τα αποτελέσματα ώστε να είναι διαθέσιμα στον client. Δημιουργήθηκαν τρεις ξεχωριστές REST API διαδρομές με GET αίτημα για αυτό το σκοπό, οι **/trends**, **/articles**, **/tweets**, δίνουν έγγραφα με τις τάσεις, άρθρα και tweets αντίστοιχα. Ουσιαστικά γίνονται τρία διαφορετικά αιτήματα στην ΒΔ και συγκεκριμένα στο collection **results** (εκεί αποθηκεύονται όλα τα αποτελέσματα και έχουν όλες τις απαραίτητες πληροφορίες). Θα μπορούσε να γίνει μία διαδρομή που να τα δίνει όλα τα δεδομένα κατευθείαν σε πακέτο. Ωστόσο για πιο “καθαρό” κώδικα και μελλοντική αποσφαλματοποίηση επιλέχθηκε να διαχωριστούν τα δεδομένα. Έτσι είναι δουλειά του client να προβάλει τα αποτελέσματα όπως αυτός θέλει. Κάθε διαδρομή επιστρέφει πληροφορίες για τα αποτελέσματα μόνο με πεδία που αφορούν τα trends, articles, tweets αντίστοιχα. Για να φανούν οι συσχετίσεις και όλες οι πληροφορίες από τα αποτελέσματα τότε πρέπει να γίνει ένωση (inner join) των δεδομένων με το κοινό κλειδί τους. Κάθε φορά που ολοκληρώνεται ο κύκλος της επεξεργασίας των δεδομένων τότε έχουμε πει, ότι δημιουργούνται 25 έγγραφα που αναπαριστούν τα αποτελέσματα. Έτσι αυτές οι τρεις διαδρομές είναι σχεδιασμένες να βρίσκουν τα αποτελέσματα σε πακέτα όπως ολοκληρώθηκαν και αποθηκεύτηκαν σε ένα κύκλο.

Μία άλλη λειτουργία του client είναι η προβολή σε χρονοδιάγραμμα της πορείας των τάσεων εντός της τελευταίας εβδομάδας. Έτσι φτιάχτηκε η διαδρομή **/searchquery** με GET αίτημα, σκοπός αυτής της διαδρομής είναι να βρίσκει τις τελευταίες ξεχωριστές τάσεις της εβδομάδας. Και δεύτερη του λειτουργία είναι να επιστρέφει αν δοθεί μία συγκεκριμένη τάση, όλες τις ξεχωριστές ημερομηνίες που καταγράφηκαν εντός της τελευταίας εβδομάδας (ώστε να προβληθεί το χρονοδιάγραμμα της). Για την διαχείριση των ημερομηνιών χρησιμοποιήθηκε η βιβλιοθήκη **dayjs**²³. Είναι βασισμένη σε φυσικές λειτουργίες της NodeJS και συγκεκριμένα στην ενσωματωμένη βιβλιοθήκη **Date**. Χρησιμοποιήθηκε για τον λόγο ότι είναι εύκολο να αφαιρεθούν, προστεθούν ημερομηνίες καθώς και η ανεκτικότητα του να αναγνωρίζει ημερομηνίες το ίδιο ακόμα και όταν είναι σε διαφορετικές μορφές.

²³ Η Day.js είναι μια μινιμαλιστική βιβλιοθήκη στηριγμένη στην NodeJS για την διαχείριση δεδομένων που αφορούν τον χρόνο, έχει μέγεθος μόλις 2kB: <https://day.js.org/>

Υπάρχει η επιλογή στον client να αναζητηθούν αποτελέσματα με βάση την ημερομηνία τους. Ο server αφού έχει δώσει πίσω στον client όλες τις διαθέσιμες μέρες που υπήρξαν αποτελέσματα, περιμένει από τον client να επιλεγεί μία μέρα, ούτως ώστε να του επιστρέψει πίσω όλες τις διαθέσιμες χρονικές σημάνσεις εκείνης της ημέρας. Για την λειτουργία αυτής της δραστηριότητας δημιουργήθηκαν δύο διαδρομές η `/daterangeresults` και η `/resultdates`. Η πρώτη δίνει την μέγιστη και ελάχιστη ημερομηνία για να οριστούν όλες οι διαθέσιμες ημέρες που μπορούν να επιλεγθούν (επειδή ο αλγόριθμος δεν σταματάει να τρέχει θα πρέπει να υπάρχουν διαθέσιμα αποτελέσματα για όλο το ημερομηνιακό φάσμα). Η δεύτερη επιστρέφει τις χρονοσημάνσεις, το φυσικό ακόλουθο είναι να ζητηθούν από τον server οι πρώτες τρεις διαδρομές που συζητήσαμε (`/trends`, `/articles`, `/tweets`) με παράμετρο την επακριβή ημερομηνία του αποτελέσματος.

Άλλη λειτουργία που είχε ανάγκη ο client είναι μία μηχανή αναζήτησης αποτελεσμάτων. Δηλαδή με την αναζήτηση μίας τάσης ή άρθρου να εμφανίζεται το αποτέλεσμα που έχει εξαχθεί. Αν αναζητηθεί μία τάση θα εμφανιστεί η καλύτερη συσχέτιση ομοιότητας που βρέθηκε με ένα άρθρο, το ίδιο ισχύει και αντίστροφα. Έτσι δημιουργήθηκε η διαδρομή `/searcharticlesbykeywords`, με POST αίτημα μιας και αποστέλλεται πληροφορία από τον client στον server, η οποία είναι το κλειδί ή κλειδιά αναζήτησης και αν ψάχνει τάσεις ή άρθρα. Για αυτό το ερώτημα στην ΒΔ δημιουργήθηκε ειδική συνάρτηση Regular Expressions που φτιάχνει το κατάλληλο ερώτημα με βάση τα κλειδιά που δόθηκαν. Τέλος, επιστρέφονται τα αποτελέσματα της αναζήτησης έτσι όπως ανακτήθηκαν από την ΒΔ.

Τέλος, τελευταίες διαδρομές που φτιάχτηκαν ήταν απλά GET αιτήματα από έτοιμα δεδομένα που έχουν προέλθει ειδικά από την επεξεργασία δεδομένων μέσω της python διαδικασίας. Αυτές είναι η `/countstats` που επιστρέφει πόσα tweets, άρθρα υπάρχουν αποθηκευμένα στην ΒΔ καθώς και πόσοι συνολικά κύκλοι της επεξεργασίας δεδομένων έχουν ολοκληρωθεί. Δεύτερη διαδρομή είναι η `/sourcesscore` η οποία επιστρέφει το άθροισμα των βαθμολογιών της συσχέτισης όλων των άρθρων μεταξύ των tweets για κάθε ειδησεογραφικό ξεχωριστά. Ωστε να δοθεί μια εικόνα του κατά πόσο τα ειδησεογραφικά γράφουν περισσότερο σχετικά με τις τάσεις του twitter. Άλλες διαδρομές αφορούσαν την επιστροφή ζευγαριών τάσεων και άρθρων των τελευταίων τριών ημερών. Αυτό έγινε με σκοπό την παρουσίαση επαναλαμβανόμενων τάσεων και από την άλλη μη επαναλαμβανόμενων τάσεων δηλαδή καινούργιες τάσεις.

5.5 Ρυθμίσεις της NodeJS και express εφαρμογής

Ο NodeJS server πρέπει να συνδεθεί με την ΒΔ για να είναι σε θέση να επικοινωνεί μαζί της και να επιστρέφει τα αποτελέσματα στον client. Για αυτό τον σκοπό χρειάζονται κάποιες πληροφορίες για την σύνδεση με την ΒΔ όπως η διεύθυνση που ακούει (listens) η mongo ΒΔ η θύρα (port) της καθώς και πληροφορίες αυθεντικοποίησης. Αυτές οι πληροφορίες επιλέχθηκαν να μουν σε ξεχωριστό αρχείο το οποίο έχει πρόσβαση η NodeJS εφαρμογή. Για αυτό το σκοπό χρησιμοποιήθηκε η βιβλιοθήκη **dotenv**²⁴, η οποία φορτώνει τις μεταβλητές περιβάλλοντος από ένα αρχείο **.env** και είναι προσβάσιμες στην εφαρμογή node. Με αυτόν τον τρόπο αποφεύγεται η χρήση hard typed (αυστηρά γραμμένων τιμών μέσα στον κώδικα) μεταβλητών και μπορούν να αλλάξουν εύκολα ώστε να γίνει η προσαρμογή από ένα περιβάλλον σε άλλο.

Παλιότερα για την αποστολή δεδομένων από τον client στον server γινόταν η χρήση **body-parser**²⁵, πλέον δεν είναι αναγκαία η χρήση της τελευταίας αν χρησιμοποιείτε στην εφαρμογή η express. Η express έχει ενσωματωμένο middleware (διαμεσολαβητή) που ονομάζεται **express.json**. Επεξεργάζεται JSON δεδομένα και τα μετατρέπει αυτόματα σε Object, έτοιμα προσβάσιμα στην NodeJS. Ουσιαστικά κάνει το ίδιο πράγμα με την body-

²⁴ Για περισσότερες πληροφορίες σχετικά με την βιβλιοθήκη dotenv: <https://www.npmjs.com/package/dotenv>

²⁵ Η βιβλιοθήκη body-parser είναι middleware της NodeJS και αυτό που κάνει είναι να αποθηκεύει την αποσπελόμενη πληροφορία από τον client μέσα στην ιδιότητα (property) της μεταβλητής req : <http://expressjs.com/en/resources/middleware/body-parser.html>

parser, καθώς έχει βασιστεί σε αυτήν ούτως ή άλλως²⁶. Έτσι σε αυτό το εγχείρημα χρησιμοποιείτε η ενσωματωμένη middleware `express.json`.

Ένα URL αν έχει διαφορετικούς χαρακτήρες από το σύνολο χαρακτήρων ASCII όπως για παράδειγμα να έχει ελληνικούς χαρακτήρες, οι τελευταίοι κωδικοποιούνται με το σύμβολο % ακολουθούμενο από δεκαεξαδικά ψηφία. Οπότε, χρησιμοποιήθηκε η ενσωματωμένη λειτουργία `urlencoded` με την επιλογή να είναι **extended**. Αυτό που κάνει στην ουσία αυτή η βιβλιοθήκη είναι να μετατρέπει Objects σε συμβολοσειρές και το αντίθετο, καθώς και πολύ σημαντικό ειδικά στην ελληνική γλώσσα την μετατροπή URL κωδικοποιημένων συμβολοσειρών σε utf-8²⁷. Η τελευταία είναι κοινή κωδικοποίηση για πάρα πολλές φυσικές γλώσσες.

Το REST API έχει φτιαχτεί για τον client αποκλειστικά και όχι σαν κάποια εξωτερική υπηρεσία, οπότε ο server έχει διττό σκοπό αρχικά να κάνει διαθέσιμο το REST-API στον χρήστη και την σελίδα διεπαφή του client. Οι τελευταίες δύο υπάρχουν στο ίδιο domain (διεύθυνση διαδικτύου) και επικοινωνούν μεταξύ τους. Για λόγους ασφαλείας οι φυλλομετρητές (browsers) περιορίζουν τέτοιου είδους επικοινωνίας (HTTP requests) από κοινή πηγή ή αλλιώς cross-origin. Αυτός ο μηχανισμός ονομάζεται CORS (Cross-Origin Resource Sharing) και για να επιτραπεί η επικοινωνία cross-origin πρέπει να οριστούν τα κατάλληλα headers μέσα στα HTTP requests του REST API μας. Για να επιτραπεί από τους browsers η επικοινωνία μέσω cross-origin χρησιμοποιήθηκε το ενσωματωμένο middleware για την NodeJS **cors**²⁸. Έχει ρυθμιστεί να δημιουργεί τα κατάλληλα cors headers σε όλες τις διαδρομές του REST-API που φτιάχτηκαν μέσω της `express`, οπότε και να επιτρέπει την επικοινωνία server-client από την ίδια διεύθυνση.

5.6 Το «πακετάρισμα» του όλου εγχειρήματος (Dockerize)

Κάθε εφαρμογή έχει δύο στάδια, το πρώτο είναι η ανάπτυξη του σε ελεγχόμενο περιβάλλον και το δεύτερο στάδιο είναι αυτό της παραγωγής. Στο στάδιο ανάπτυξης της εφαρμογής είναι στο χέρι του σχεδιαστή να διαμορφώσει το δικό του περιβάλλον με όποια εργαλεία επιθυμεί. Από την άλλη το στάδιο της παραγωγής ουσιαστικά είναι η ολοκληρωμένη εφαρμογή και έτοιμη για χρήση, ενδεχομένως σε οποιοδήποτε προγραμματιστικό περιβάλλον. Το πρόγραμμα που αναπτύχθηκε σε αυτό το εγχείρημα ήταν δεδομένο ότι θα χρειαστεί να μεταφερθεί σε διαφορετικό περιβάλλον από αυτό που δημιουργήθηκε. Για αυτό το σκοπό χρησιμοποιήθηκε η πλατφόρμα Docker²⁹.

Η πλατφόρμα Docker επιτρέπει στους σχεδιαστές να «πακετάρουν» την εφαρμογή τους, ούτως ώστε να είναι δυνατόν με πολύ εύκολο τρόπο να λειτουργεί σε οποιοδήποτε σύστημα που υποστηρίζει την πλατφόρμα. Το Docker παρέχει απομόνωση και ασφάλεια μεταξύ των διαφορετικών κομματιών που μπορεί να έχει μία εφαρμογή. Αυτό το πετυχαίνει με την χρήση των containers, μπορούμε να φανταστούμε ότι αυτά εκτελούν κάποια λειτουργία και στην σύνθεση τους είναι η τελική εφαρμογή. Τα containers εμπεριέχουν όλα τα απαραίτητα στοιχεία για την εκτέλεση τους, με αυτόν τον τρόπο δεν χρειάζεται να βασίζονται σε αυτά που είναι ήδη εγκατεστημένα στο σύστημα, καθώς υπάρχει πλήρης απομόνωση και λειτουργούν ανεξάρτητα. Έτσι κάθε φορά που μεταφέρεται η εφαρμογή σε νέο σύστημα είναι βέβαιο ότι θα λειτουργεί με τον ίδιο τρόπο.

Υπάρχουν τέσσερα συστατικά που συντελούν το Docker και αυτά είναι τα images, containers, volumes και networks. Τα images περιέχουν όλες τις οδηγίες για την δημιουργία ενός container και είναι έτοιμα οποιαδήποτε στιγμή να εκτελεστούν. Το container είναι μία μορφή εικονικού συστήματος το οποίο περιέχει όλο τον κώδικα και τις εξαρτήσεις του, ώστε η εφαρμογή να εκτελείται γρήγορα και αξιόπιστα. Φυσικά είναι αναγκαίο να αποθηκεύονται δεδομένα που είναι απαραίτητα για την ορθή λειτουργία της εφαρμογής καθώς και η επικοινωνία μεταξύ των

²⁶ Στην πρώτη πρόταση αναφέρεται ότι το middleware `express.json` βασίστηκε στην `body-parser`:
<https://expressjs.com/en/api.html>

²⁷ HTML Uniform Resource Locators: https://www.w3schools.com/html/html_urlencode.asp

²⁸ Η βιβλιοθήκη middleware CORS: <https://www.npmjs.com/package/cors>

²⁹ Περισσότερες πληροφορίες για την πλατφόρμα Docker: <https://www.docker.com/>

containers. Και σε αυτή την περίπτωση υπάρχουν ξεχωριστές οντότητες volumes (εικονικό αποθηκευτικό μέσω) και networks (δίκτυα) αντίστοιχα.

Το παρόν εγχείρημα χωρίζεται σε τρία διαφορετικά containers. Αυτά είναι ο server, η βάση δεδομένων και ο αλγόριθμος python για την επεξεργασία των δεδομένων. Το container του server και του αλγόριθμου σε python, πρέπει να επικοινωνούν με το container της βάσης δεδομένων, οπότε πρέπει να οριστεί ένα κοινό δίκτυο (network). Η ΒΔ είναι υπεύθυνη για την αποθήκευση των δεδομένων οπότε αυτό το αντίστοιχο container χρειάζεται να έχει volume. Για την εγκατάσταση των απαραίτητων εξαρτήσεων έχουν οριστεί αρχεία που περιέχουν όλες τις βιβλιοθήκες που πρέπει να εγκατασταθούν καθώς και την έκδοση τους ώστε να επιτευχθεί σταθερότητα³⁰.

Για να επιτευχθούν όλα αυτά με αυτόματο τρόπο η πλατφόρμα Docker έχει δημιουργήσει την δικιά της γλώσσα εντολών ώστε ο σχεδιαστής να ορίσει τι θα εγκατασταθεί και τι θα περιέχεται σε κάθε container καθώς και πως θα συνδέονται μεταξύ τους. Υπάρχουν δύο γλώσσες, η πρώτη είναι αυτή του αρχείου κειμένου Dockerfile που περιέχει όλες τις εντολές/οδηγίες για να κατασκευαστεί ένα image. Η άλλη είναι η docker-compose που έχει βασιστεί στην σύνταξη YAML³¹ και αυτή έχει σκοπό να ορίσει και να εκτελέσει πολλά containers μαζί τα οποία συντελούν μια εφαρμογή. Οπότε τα τρία διαφορετικά containers της εφαρμογής εδώ έχουν το καθένα ένα **Dockerfile** που ορίζει τις οδηγίες για να κατασκευαστεί το image τους. Και όλα αυτά ενώνονται και λειτουργούν μαζί με την χρήση ενός **docker-compose** αρχείου που περιέχει όλες τις ρυθμίσεις.

³⁰ Επειδή όλες οι βιβλιοθήκες επί των πλείστων ανανεώνονται συνεχώς με την πάροδο του χρόνου, μπορεί κώδικας που έχει γραφτεί για αυτές να μην είναι συμβατός με μελλοντικές εκδόσεις της ίδιας της βιβλιοθήκης καθώς αυτή ενδεχομένως να έχει αναθεωρηθεί.

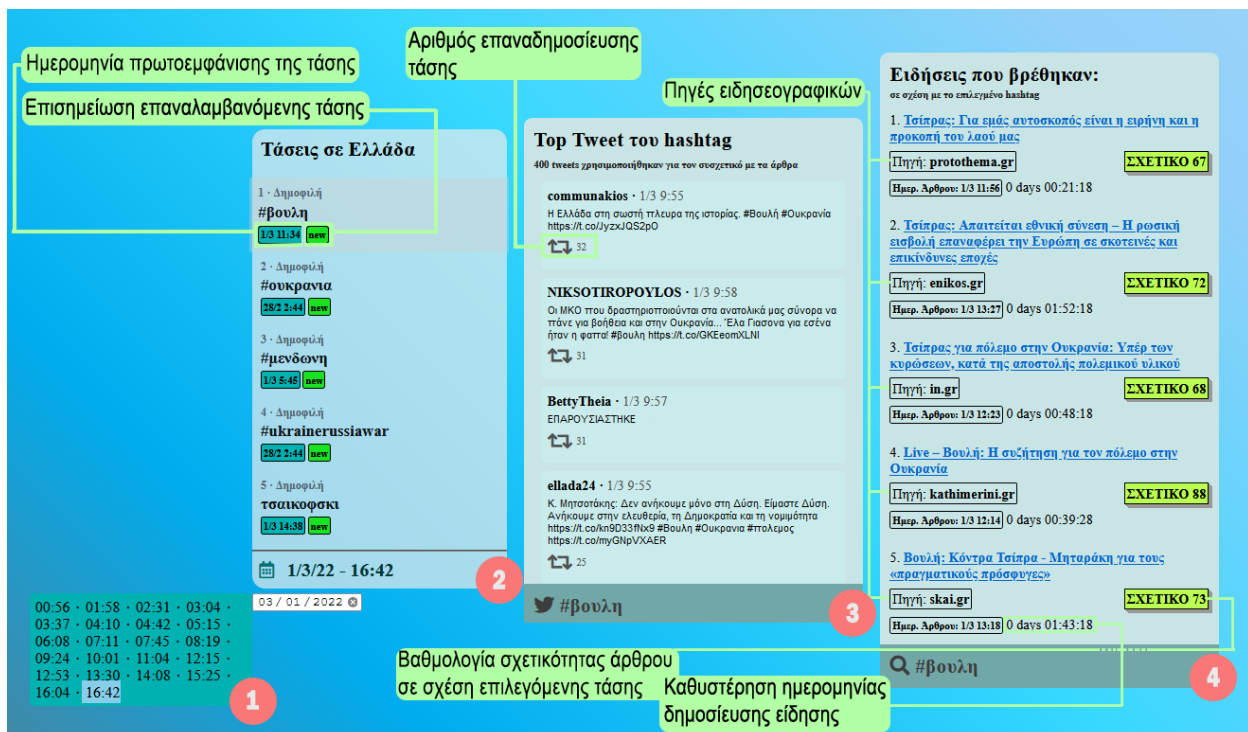
³¹ Η YAML είναι αναδρομικό ακρωνύμιο (YAML Ain't Markup Language) και είναι μια φιλική προς τον άνθρωπο γλώσσα για την δόμηση δεδομένων.

6. Οπτικοποίηση & Συμπεράσματα των δεδομένων

Για την παρουσίαση και προβολή των αποτελεσμάτων που έχουν εξαχθεί από την επεξεργασία δεδομένων κατασκευάστηκε μια εφαρμογή – ιστοσελίδα. Σκοπός της είναι να προβάλει στους χρήστες διάφορες πτυχές και πληροφορίες σχετικά με τις συσχετίσεις άρθρων από ειδησεογραφικά και τάσεις από tweets. Καθώς και διάφορες πληροφορίες σχετικά με τις τάσεις του Twitter, βαθμολογία των ειδησεογραφικών σχετικά με το πόσο ακολουθούν τις τάσεις στο Twitter και γενικά στατιστικά για το μέγεθος της εφαρμογής. Το κομμάτι της οπτικοποίησης των δεδομένων είναι πολύ σημαντικό ώστε να μπορούν να γίνουν κατανοητά τα αποτελέσματα από τον άνθρωπο. Είναι ευκολότερο για το ανθρώπινο εγκέφαλο να επεξεργαστεί την πληροφορία χρησιμοποιώντας μεθόδους οπτικοποίησης μεγάλου πλήθους πολύπλοκων δεδομένων. Θα γίνει περιγραφή των τεχνολογιών που χρησιμοποιήθηκαν για την κατασκευή της διεπαφής ιστοσελίδας καθώς και επεξήγηση κάθε οπτικοποίησης ξεχωριστά.

6.1 Περιήγηση στην εφαρμογή

Η ιστοσελίδα είναι με μία και μοναδική σελίδα και όλες οι πληροφορίες της εφαρμογής βρίσκονται εκεί. Ο χρήστης με το που ανοίξει την σελίδα του παρουσιάζονται οι πρώτες πέντε τάσεις του Twitter περίπου εκείνη την στιγμή καθώς και διάφορες πληροφορίες με τις συσχετίσεις που πραγματοποιήθηκαν με τα άρθρα.

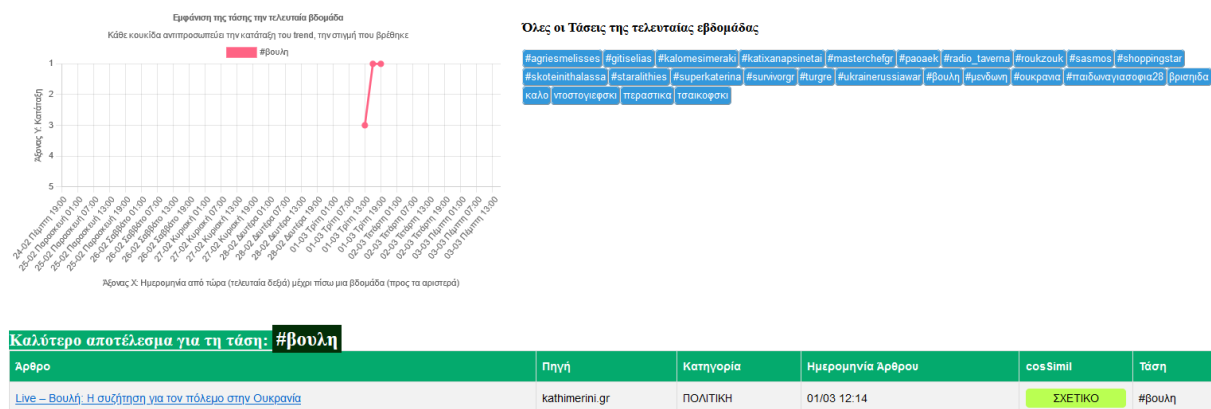


Εικόνα 5 Η αρχική εικόνα και κύρια λειτουργία της διεπαφής. Υπάρχουν επισημειωμένα διάφορα στοιχεία που επεξηγούν τις λειτουργίες.

Στην ουσία είναι τρεις πίνακες (επισημειωμένοι με τους αριθμούς στα κόκκινα κυκλάκια 2,3,4) και η επιλογή ημερομηνίας και ώρας (1). Η επιλογή που δίνεται στον χρήστη εδώ είναι να διαλέξει μια ημερομηνία και ώρα (1) και η εφαρμογή θα του εμφανίσει τις τάσεις που υπήρχαν τότε. Στην συνέχεια μπορεί να επιλέξει πατώντας με τον κέρσορα κάποια τάση από τον πίνακα με τις τάσεις (2). Εδώ εμφανίζονται πέντε τάσεις (#βουλη, #ουκρανια, #μενδωνη, #ukrainerrussiawar, τσαικοφοσκι) για την ημερομηνία 03/01/2022 16:42 . Οι πληροφορίες που προβάλλονται σε αυτόν τον πίνακα είναι ο τίτλος της τάσης, η ημερομηνία που πρωτο-εντοπίστηκε η τάση καθώς και αν είναι καινούργια για την εφαρμογή ή αν έχει ξαναεμφανιστεί στο παρελθόν. Στην εικόνα φαίνεται να έχει επιλεγεί η τάση **#βουλη** (στο κάτω μέρος του πίνακα 3,4 φαίνεται ότι έχει επιλεγεί η συγκεκριμένη τάση). Και η ημερομηνία πρωτοεμφάνισης της στην κατάταξη τάσεων του Twitter είναι στις **1/3 11:34** και επισημαίνεται σαν καινούργια τάση για την εφαρμογή, δηλαδή δεν έχει ξαναεμφανιστεί στο παρελθόν. Στον επόμενο πίνακα (3) φαίνονται τα κορυφαία τέσσερα tweets για αυτήν την τάση με βάση τον αριθμό αναδημοσιεύσεων της (re-tweets), ώστε ο χρήστης να πάρει μια ιδέα για το περιεχόμενο που γράφουν οι χρήστες του Twitter αναφορικά με την επιλεγμένη τάση. Ο τελευταίος πίνακας είναι ο πιο σημαντικός καθώς εκεί παρουσιάζονται τα άρθρα κάθε ειδησεογραφικού που σχετίζονται με την τάση. Παρέχονται έξι πληροφορίες σχετικά με την συσχέτιση, αρχικά ο τίτλος του άρθρου μαζί με το link ώστε ο χρήστης να μπορεί να δει περί τίνος πρόκειται το άρθρο και αν έχει σχέση με την τάση. Αναγράφεται η πηγή του άρθρου, στην συγκεκριμένη υλοποίηση επιλέχθηκαν πέντε ειδησεογραφικά αλλά θα μπορούσε να ήταν διαφορετικά ή περισσότερα. Σημειώνεται η ημερομηνία που δημοσιεύθηκε το άρθρο και δίπλα σε αυτήν η καθυστέρηση της σε σχέση με την πρωτοεμφάνιση της ημερομηνίας της τάσης, σε αυτό το παράδειγμα όλα τα ειδησεογραφικά καθυστέρησαν λιγότερο από δύο ώρες να δημοσιεύσουν τα άρθρα. Και το πιο σημαντικό από όλα, η σχετικότητα του άρθρου με την τάση μαζί με την βαθμολογία τους (όσο μεγαλύτερη είναι τόσο το καλύτερο), εδώ έχουν επισημειωθεί όλα τα άρθρα σαν σχετικά και το πιο σχετικό άρθρο φαίνεται να είναι της *kathimerinis.gr* . Η ίδια λειτουργία παρέχεται για όλες τις τάσεις στον πίνακα 2, καθώς και για όλες τις τάσεις πίσω στον χρόνο μαζί με τα συσχετισμένα άρθρα, έτσι όπως είχαν υπολογιστεί τότε.

Στην συνέχεια, ακολουθεί χρονοδιάγραμμα της εξέλιξης των τάσεων σχετικά με την κατάταξη τους στον πίνακα κορυφαίων τάσεων του Twitter. Με σκοπό να δίνεται στον χρήστη η δημοτικότητα ή μη μιας τάσης, για παράδειγμα κάποιες τάσεις βρίσκονται μέχρι και μέρες στον πίνακα των τάσεων άλλες βρίσκονται για πολύ μικρό χρονικό διάστημα.

» Πατήστε πάνω σε κάποια τάση για να δείτε την πορεία του στην κατάταξη σε σχέση με το χρόνο



Εικόνα 6 Χρονοδιάγραμμα των τάσεων έτσι όπως εμφανίζεται στην εφαρμογή. Ο χρήστης πατάει σε κάποια τάση και του εμφανίζεται η πορεία της κατάταξης της τάσης σε σχέση με τον χρόνο.

Δεξιά, υπάρχουν κουμπιά με τις τάσεις της τελευταίας εβδομάδας, πατώντας κάποιο από αυτό θα φανεί στα αριστερά ένα χρονοδιάγραμμα της πορείας του. Στον οριζόντιο άξονα βρίσκεται ο χρόνος, χωρισμένος ανά τρεις ώρες και στον κάθετο άξονα βρίσκεται η κατάταξη (μόνο πρώτες πέντε). Κάθε φορά που πατιέται κάποια τάση στο κάτω μέρος εμφανίζεται το καλύτερο αποτέλεσμα για αυτήν, δηλαδή το πιο σχετικό άρθρο που βρέθηκε μέχρι εκείνη την στιγμή. Οι τάσεις στον πίνακα του Twitter δεν ανεβοκατεβαίνουν κατατάξεις γραμμικά, στο παραπάνω παράδειγμα η τάση **#βουλη** εμφανίζεται κατευθείαν στην τρίτη θέση την ώρα 13:00 και αμέσως μετά στις 16:00 εμφανίζεται πρώτη. Από την μία, αυτό μπορεί να συμβαίνει γιατί πολύ απλά η εφαρμογή τραβάει δεδομένα από το Twitter API περίπου ανά ώρα, οπότε είναι πιθανό να μην έχουν καταγραφεί όλες οι θέσεις της τάσης στην κατάταξη. Από την άλλη, μετά από προσωπικές παρατηρήσεις την πλατφόρμας Twitter, διαπιστώθηκε ότι ούτε το ίδιο εμφανίζει την ομαλή προσπέραση μιας τάσης από άλλη. Πιθανή εξήγηση είναι το Twitter να επιστρέφει στην πλατφόρμα ανά σταθερό χρόνο την κατάσταση του πίνακα τάσεων για να κάνει εξοικονόμηση σε υπολογιστικούς πόρους. Με αποτέλεσμα να μην καταγράφεται με απόλυτη ακρίβεια η κατάσταση στο χρονοδιάγραμμα η πορείας της τάσης στην κατάταξη σε σχέση με τον χρόνο αλλά στο περίπου (σε διαστήματα τριών ωρών σε αυτό το εγχείρημα).

Για την εύρεση παρελθοντικών τάσεων ή άρθρων δημιουργήθηκε στην εφαρμογή, μηχανή αναζήτησης. Ο χρήστης πληκτρολογεί μια τάση ή κάποια λέξη κλειδί από τον τίτλο κάποιου άρθρου και του εμφανίζονται τα καλύτερα αποτελέσματα αντίστοιχα.

πολεμο
● άρθρο ○ τάση

Άρθρο	Πηγή	Κατηγορία	Ημερ. Άρθρου	Σχετικότητα	Τάση
Ουκρανία: Γιατί ο Πούτιν έχει ήδη χάσει αυτόν τον πόλεμο – Η ανάλυση διάσημου ιστορικού	enikos.gr	international	28/02 18:21	ΣΧΕΤΙΚΟ	#ukrainerussiawar
Δείτε live τη συζήτηση στη Βουλή για τις επιπτώσεις του πολέμου στην Ουκρανία στην Ελλάδα	in.gr	Βουλή	01/03 11:14	ΣΧΕΤΙΚΟ	#βουλη
Τοίπρας για πόλεμο στην Ουκρανία: Υπέρ των κυρώσεων, κατά της αποστολής πολεμικού υλικού	in.gr	Βουλή	01/03 12:23	ΣΧΕΤΙΚΟ	#βουλη
Live – Βουλή: Η συζήτηση για τον πόλεμο στην Ουκρανία	kathimerini.gr	ΠΟΛΙΤΙΚΗ	01/03 12:14	ΣΧΕΤΙΚΟ	#βουλη
Πόλεμος στην Ουκρανία: Αντίποινα της Ρωσίας σε 36 χώρες – Κλείνει τον εναέριο χώρο της	kathimerini.gr	ΚΟΣΜΟΣ	28/02 16:35	ΣΧΕΤΙΚΟ	#ukrainerussiawar
Ο πόλεμος στην Ουκρανία μέσα από τα πρωτοσέλιδα του διεθνούς τύπου	kathimerini.gr	ΚΟΣΜΟΣ	28/02 19:08	ΣΧΕΤΙΚΟ	#ukrainerussiawar
Live: Πέμπτη ημέρα πολέμου στην Ουκρανία – Όλες οι εξελίξεις	kathimerini.gr	ΚΟΣΜΟΣ	28/02 08:34	ΣΧΕΤΙΚΟ	#ukrainerussiawar
Πόλεμος στην Ουκρανία: Είσοδος στη νέα παγκόσμια αταξία	kathimerini.gr	ΚΟΣΜΟΣ	28/02 14:50	ΣΧΕΤΙΚΟ	#ουκρανια
Αλ. Τσίπρας: Η Ελλάδα μετατρέπεται σε εμπλεκόμενη του πολέμου	kathimerini.gr	ΠΟΛΙΤΙΚΗ	01/03 13:54	ΣΧΕΤΙΚΟ	τσαικοφοκι
Βουλή: Στην Ολομέλεια ο πόλεμος στην Ουκρανία κι οι επιπτώσεις του για την Ελλάδα	kathimerini.gr	ΒΟΥΛΗ	01/03 08:10	ΣΧΕΤΙΚΟ	#βουλη

Εικόνα 7 Υλοποιημένη μηχανή αναζήτησης που είναι διαθέσιμη στην εφαρμογή αυτού του εγχειρήματος.

Έτσι όπως φαίνεται στην παραπάνω εικόνα το κλειδί ή κλειδιά εισάγονται χωρισμένα από κενό (spacebar) στο πεδίο αναζήτησης (1). Στην πορεία επιλέγει αν αυτό που εισάγει είναι τάση ή άρθρο (2) και τα αποτελέσματα του εμφανίζονται απευθείας καθώς πληκτρολογεί κάτι. Στο παραπάνω παράδειγμα εισάχθηκε το κλειδί *πόλεμος* και πραγματοποιείτε αναζήτηση στους τίτλους των άρθρων. Όπως φαίνεται παρουσιάζονται τα αποτελέσματα με την μορφή πίνακα (3). Επιπλέον υπάρχει η επιλογή να ταξινομήσει κατά φθίνουσα ή αύξουσα σειρά κάθε στήλη πατώντας πάνω στο λευκό βελάκι που υπάρχει δίπλα από τον τίτλο κάθε στήλης. Με αυτόν τον τρόπο ο χρήστης μπορεί να ψάξει κάποιο άρθρο ώστε να δει με ποια τάση σχετίζεται και αντίστροφα να ψάξει κάποια τάση ώστε να δει με ποιο άρθρο είναι πιο κοντά.

Επιπλέον έχουν χωριστεί οι τάσεις σε δύο κατηγορίες, σε αυτές που επαναλαμβάνονται και σε αυτές που είναι καινούργιες για την εφαρμογή. Οι τάσεις που επαναλαμβάνονται συνήθως αφορούν γενική θεματολογία και έχει νόημα οι χρήστες να την επαναχρησιμοποιούν για διάφορους λόγους.

Οι τάσεις που βρέθηκαν με μεγάλη σχετικότητα

Πιθανώς το άρθρο να δείχνει για ποιά πράγμα μιλάει η τάση

καλο	MasterChef: Το δημοφιλέστερο γεμιστό φαΐ στον κόσμο - Αυτοί είναι οι υποψήφιοι προς αποχώρηση
#παιδοναγισσοφια28	Τρία χρόνια Cinemaethis: Η Γιάννα Δολιγάνη μας καλεί να απολαύσουμε κινηματογράφο
βρεσηδα	Survivor: Βρέθηκε Μπέη η ασυλία - Υποψήφια η Βρεσηίδα

Οι τάσεις που επαναλαμβάνονται

Τάσεις των τελευταίων 3 ημερών που έχουν ξαναεμφανιστεί στο παρελθόν

περαστικά	Κινέζοι: Ουάιερ με νεκρό άνδρα
#μενδονη	Ο Τσεϊκόφσκι απαγορεύτηκε η όλη σκόνη: - Χωρίς με την απόφαση του υπουργείου Πολιτισμού να ακυρώσει τη «Λίμνη των Κίρκων»
#gitiselias	Η Γα της Ελλάς: Η Αρετή συνεχίζει να πιέζει τον Στάθη να τη βρουν
#agriesmelisses	Λυγρές Μελισσές: Η Ελένη θα αποκαλύψει στον Λάμπρο ένα μεγάλο μυστικό
#masterchefgr	MasterChef: Απαρηγόρητος ο Δημήτρης: - «Κεν» το λένε - BINTO
#sasmos	Σασμός: Ένα στέγιο για την εξόντωση του Μοθού βρίσκεται στο σκαμνί από τους Βρονιάδες

Οι καινούργιες τάσεις των τελευταίων 3 ημερών

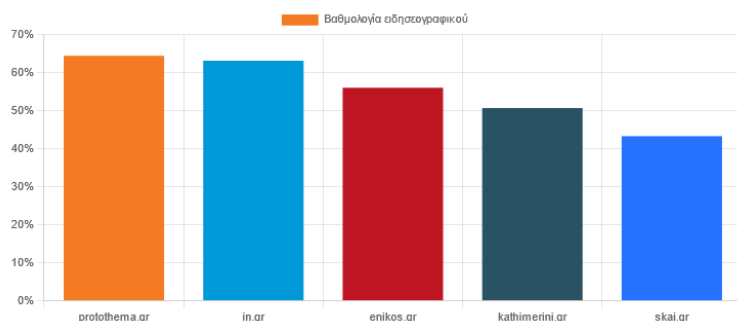
Δείχνει τι θέματα απασχολούν το twitter τις τελευταίες 3 μέρες

τσαικοφσκι	Ο Τσεϊκόφσκι απαγορεύτηκε η όλη σκόνη: - Χωρίς με την απόφαση του υπουργείου Πολιτισμού να ακυρώσει τη «Λίμνη των Κίρκων»
#βουλη	Live - Βουλή: Η εκδήλωση για τον πόλεμο στην Ουκρανία
#ukrainerussiawar	Το Ρωσοουκρανικό «casus belli»
#ουκρανία	Βενζέζουε: Η απειλή γρήγορης πυρηνικής από την Ρωσία, προκαλεί τη Δύση
ντοστογιεφσκι	600 λέξεις με την Διζάνθρα Γρηγορίου

Εικόνα 8 Αριστερά είναι ο πίνακας με τάσεις που βρέθηκαν με μεγάλη ομοιότητα με κάποιο άρθρο. Στο κέντρο παρουσιάζονται τάσεις των τελευταίων ημερών που επαναλήφθηκαν και δεξιά τάσεις των τελευταίων τριών ημερών που είναι καινούργιες.

Στην εφαρμογή υπάρχουν τρεις πίνακες, ο αριστερός πίνακας αφορά τάσεις που βρέθηκαν με μεγάλη ομοιότητα με κάποιο άρθρο. Αν ένα άρθρο έχει σημειώσει μεγάλη βαθμολογία τότε είναι πολύ πιο πιθανό να είναι πιο κοντά εννοιολογικά με την τάση που έχει συσχετιστεί, από ότι τα άρθρα που έχουν σημειώσει μικρότερη βαθμολογία. Ο κεντρικός πίνακας αφορά τις τάσεις των τελευταίων τριών ημερών που έχουν ξαναεμφανιστεί στο παρελθόν. Ο δεξιά πίνακας αφορά τάσεις των τελευταίων τριών ημερών που δεν επαναλαμβάνονται, με αποτέλεσμα να προβάλλονται τα θέματα που απασχολούν το Twitter τις τελευταίες τρεις μέρες. Σε όλους τους πίνακες υπάρχουν ζεύγη της τάσης με το πιο σχετικό τίτλο άρθρου που βρέθηκε από την επεξεργασία δεδομένων.

Ποσοστό των άρθρων που βρέθηκαν με μεγάλη σχετικότητα με κάποια τάση.



Εικόνα 9 Ποσοστό των άρθρων που βρέθηκαν με μεγάλη σχετικότητα με κάποια τάση.

Όσο περνάει ο χρόνος η εφαρμογή συλλέγει πολλές συσχετίσεις από όλα τα ειδησεογραφικά. Για κάθε άρθρο των ειδησεογραφικών υπάρχει συνδεδεμένη κάποια τάση με μία συγκεκριμένη βαθμολογία. Αν πάρουμε αυτές τις συνδέσεις που έχουν υψηλή βαθμολογία (άρα και μεγαλύτερη πιθανότητα να υπάρχει πραγματική συσχέτιση άρθρου-τάσης) και τις αθροίσουμε για κάθε ξεχωριστό ειδησεογραφικό, παίρνουμε την βαθμολογία του ειδησεογραφικού στο σύνολο του. Κατά κάποιο τρόπο δηλαδή μετριέται κατά πόσο ακολουθούν τα ειδησεογραφικά τις τάσεις του Twitter. Τα

αποτελέσματα που προβάλλονται στην παραπάνω εικόνα βασίζονται σε αποτελέσματα που υπολογίστηκαν σε αυτό το εγχείρημα με βάση 46.000 άρθρα και 220.000 tweets μεταξύ του Νοέμβρη και Δεκέμβρη του έτους 2021.

6.2 Η επιλογή του Web Framework Angular

Σε κάθε σχεδίαση λογισμικού οι σχεδιαστές πρέπει να ερευνήσουν ποια εργαλεία και τεχνολογίες είναι κατάλληλες για το πρόβλημα τους ώστε να τα χρησιμοποιήσουν. Στα παλαιότερα χρόνια οι σχεδιαστές διαδικτυακών εφαρμογών βασιζόντουσαν κυρίως στην φυσική γλώσσα JavaScript με ελάχιστες προεκτάσεις της. Ωστόσο όσο οι διαδικτυακές εφαρμογές γινόντουσαν όλο και πιο πολύπλοκες, δημιουργήθηκαν διάφορα εργαλεία για την διευκόλυνση τους. Τα Web Frameworks αναπτύχθηκαν για να απλοποιήσουν την διαδικασία σχεδίασης μιας ιστοσελίδας. Παρέχουν σχεδιαγράμματα-πρότυπα ικανά να παρουσιάσουν την πληροφορία σε ένα φυλλομετρητή, δίχως να χρειάζεται ο σχεδιαστής να σκεφτεί τόσο πολύ τις επαναλαμβανόμενες διαδικασίες και χαρακτηριστικά που είναι αναγκαία για την ανάπτυξη μιας ιστοσελίδας. Όπως για παράδειγμα το caching, routing, αυθεντικοποίηση κ.α., αυτό σημαίνει ότι ο σχεδιαστής θα επικεντρώσει τον περισσότερο χρόνο του στην παρουσίαση του περιεχομένου της ιστοσελίδας παρά σε ρυθμίσεις που αφορούν το παρασκήνιο. Εν ολίγης, τα Web Frameworks είναι ένα σύνολο από εργαλεία προσανατολισμένα στην κατασκευή μιας ιστοσελίδας. Η ιστοσελίδα που κατασκευάστηκε για αυτό το εγχείρημα είναι γραμμένη με το Web Framework Angular.

Με την Angular μπορούν να κατασκευαστούν PWA (Progressive Web Applications) εφαρμογές, οι οποίες είναι συμβατές σε κάθε πλατφόρμα, όπως για παράδειγμα εφαρμογή σε κινητό, εφαρμογές σε λειτουργικό σύστημα, ιστοσελίδα κ.α. . Στο παρών εγχείρημα η διεπαφή σχεδιάστηκε αποκλειστικά μόνο σαν ιστοσελίδα, χωρίς όμως να αποκλείεται στο μέλλον η μετατροπή λίγου κώδικα για την υποστήριξη σε άλλες μορφές. Χρησιμοποιεί την MVC (Model View Controller) αρχιτεκτονική, που το model (μοντέλο) αφορά την διαχείριση των δεδομένων, το View είναι υπεύθυνο για την προβολή τους και το Controller είναι η σύνδεση μεταξύ σε αυτά τα δύο. Το συγκεκριμένο framework είναι πιο γνωστό για το χαρακτηριστικό του *Two-Way Data Binding*, το επίπεδο View αναπαριστά απολύτως το επίπεδο Model που σημαίνει ότι άμα αλλάξει κάτι στα δεδομένα, αυτή η αλλαγή είναι σε τέλειο συγχρονισμό με την προβολή των δεδομένων στο View. Η κατασκευή του ίδιου framework είναι γραμμένη σε Typescript και είναι η κεντρική γλώσσα του. Η Typescript είναι ένα υποσύνολο της JavaScript, προσφέρει πιο πλούσια σύνταξη με προσανατολισμό στον αντικειμενοστραφή προγραμματισμό.

6.3 Τα χαρακτηριστικά της Angular εφαρμογής

Με την χρήση της Angular όλα τα στοιχεία της ιστοσελίδας αντιμετωπίζονται σαν αυτόνομα ξεχωριστά components. Τα components είναι το πιο βασικό δομικό συστατικό της Angular, περιέχει τρία αρχεία που χειρίζονται την συμπεριφορά του στοιχείου (typescript), το στυλ (η γλώσσα CSS ορίζει το πως φαίνεται γραφιστικά το στοιχείο) του και την βασική html δομή του. Τα components μπορούν να επικοινωνούν μεταξύ τους και να μεταφέρουν δεδομένα από το ένα στο άλλο ώστε να πραγματοποιηθεί η επιθυμητή λειτουργία. Η Angular είναι SPA (Single-Page app) Web Framework, οπότε υπάρχει η αρχική σελίδα της ιστοσελίδας και από εκεί και πέρα μέσω του DOM³² (Document Object Model) χειρισμού και του routing εμφανίζονται άλλα components κατά βούληση. Σε αυτό το εγχείρημα δεν υπάρχουν υποσελίδες και όλα τα components εμφανίζονται στην αρχική.

Τα δεδομένα φορτώνονται στα components μετά από HTTP αιτήματα στο REST API. Άλλες φορές μέσω services ή κατευθείαν μέσα από τα components. Τα services είναι χαρακτηριστικό της Angular όπου είναι διαθέσιμα μέσα από όλα τα components και συνήθως χρησιμοποιούνται για κώδικα ο οποίος έχει νόημα να

³² Το DOM συνδέει τον HTML κώδικα με προγραμματιστικό κώδικα, συνήθως με την JavaScript:
https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

επαναχρησιμοποιείται από διάφορα components. Έτσι λοιπόν εδώ, τα services χρησιμοποιούνται για να γεμίσουν με δεδομένα τους πίνακες με τις τάσεις, tweets και άρθρα. Άλλες λειτουργίες πιο σύντομες, πραγματοποιούνται απευθείας μέσα από τα components, όπως το POST αίτημα στο REST API για την αποστολή των κλειδιών αναζήτησης και την επιστροφή των αποτελεσμάτων.

Τα δεδομένα έχουν μια συγκεκριμένη μορφή όταν συλλέγονται από το REST API, για να μετατραπεί σε διαφορετική μορφή, πιο κατάλληλη για τον άνθρωπο χρησιμοποιούνται τα pipes. Τα pipes είναι λειτουργία της Angular ενσωματωμένη στο View, τα οποία έχει νόημα να υπάρχουν μόνο αν επαναχρησιμοποιούνται, με λίγα λόγια είναι μικρές συναρτήσεις με σκοπό την μετατροπή μιας μορφής δεδομένων σε μια άλλη. Εδώ χρησιμοποιήθηκαν για την μετατροπή της ημερομηνίας στην επιθυμητή μορφή, την μετατροπή string σε νούμερο με καθορισμένα δεκαδικά ψηφία κ.α. .

Για την δημιουργία των διαγραμμάτων χρησιμοποιήθηκε η βιβλιοθήκη **chart.js**³³, για την κατασκευή ενός διαγράμματος είναι υποχρεωτική η μετατροπή των δεδομένων με τέτοιο τρόπο ώστε να μπορεί να εισαχθεί στις μεθόδους δημιουργίας διαγράμματος από την chart.js . Σε γενικές γραμμές τρία πράγματα πρέπει να οριστούν, ο τύπος του διαγράμματος, τα δεδομένα και τα μεταδεδομένα του. Στην περίπτωση του χρονοδιαγράμματος ο τύπος του διαγράμματος είναι τύπου **line**³⁴ και στη περίπτωση της βαθμολογίας των ειδησεογραφικών τύπου **bar**³⁵. Στην περίπτωση του bar διαγράμματος τα πράγματα είναι απλά καθώς τα δεδομένα είναι μονοδιάστατα. Στην περίπτωση του line διαγράμματος στο χρονοδιάγραμμα τα δεδομένα είναι δισδιάστατα, οπότε πρέπει να οριστεί ό,τι αφορά τον οριζόντιο και κάθετο άξονα. Ο κάθετος άξονας αφορά την κατάταξη της τάσης και ο οριζόντιος άξονας αφορά ημερομηνίες, ωστόσο θα ήταν αδύνατο να εμφανιστούν όλες οι ημερομηνίες που ο αλγόριθμος κατέγραψε από τον πίνακα των τάσεων του Twitter. Με σύντομους υπολογισμούς, αν ο αλγόριθμος επεξεργασίας δεδομένων επαναλαμβάνεται κάθε μισή ώρα και η ολοκλήρωση του παίρνει κατά μέσο όρο περίπου 30 λεπτά, έχουμε 24 διαφορετικές καταγεγραμμένες ημερομηνίες εμφάνισης μιας τάσης την ημέρα. Μέσα σε μία βδομάδα αυτό σημαίνει 168 χρονικά σημεία, τα οποία είναι πάρα πολλά για να οπτικοποιηθούν στον οριζόντιο άξονα. Για την επίλυση αυτού του προβλήματος μια βδομάδα χωρίστηκε ανά τρεις ώρες και όλα τα χρονικά σημεία συγχωνεύτηκαν μέσα σε φάσματα τριών ωρών. Το τρίτο στοιχείο που χρειάζεται για την κατασκευή ενός διαγράμματος με chart.js είναι τα μεταδεδομένα και οι γραφικοί παράμετροι.

6.4 Ερμηνεία αποτελεσμάτων της εφαρμογής

Εμπειρικά από τις παρατηρήσεις του ερευνητή και με την βοήθεια της διεπαφής που είναι διαθέσιμη, έχει βγει το συμπέρασμα ότι, αν το άρθρο έχει προηγηθεί της τάσης τότε σημαίνει ότι το ειδησεογραφικό πρόλαβε την είδηση και η ελληνική κοινότητα στο Twitter αναμετάδωσε την είδηση. Στην αντίθετη περίπτωση, που το ειδησεογραφικό έχει καθυστερήσει να δημοσιεύσει κάποια είδηση, αυτό μπορεί να σημαίνει ότι το Twitter καθώς και άλλες πλατφόρμες κοινωνικών δικτύων, πιθανών να άσκησαν πίεση προς τη δημοσίευση κάποιου άρθρου. Γενικά παρατηρείται ότι τα ειδησεογραφικά γράφουν πρώτα την είδηση και μετά ακολουθούν οι τάσεις στο Twitter. Ωστόσο, αυτό δεν είναι εντελώς σωστό, διότι μια λέξη κλειδί (hashtag) για να γίνει τάση χρειάζεται κάποιος χρόνος ώστε να αναρριχηθεί στις κορυφαίες πέντε θέσεις. Υπολογίζεται στο περίπου ότι παίρνουν κάπου στις 4 με 16 ώρες αναδημοσιεύσεων και επιπλέον δημοσιεύσεων ώστε μια λέξη κλειδί να γίνει γνωστή ώστε να μετατραπεί σε τάση.

Κάθε φορά που γίνεται συσχέτιση άρθρου και τάσης, μπορούμε να θεωρήσουμε ότι βρέθηκε η κατηγορία της τάσης. Κάθε ειδησεογραφικό κατηγοριοποιεί τα άρθρα του σε κατηγορίες (για την απλούστευση των δεκάδων διαφορετικών κατηγοριών που έχουν τα ειδησεογραφικά συγχωνεύτηκαν σε πέντε γενικές κατηγορίες: Πολιτικά, Κοινωνικά, Οικονομικά, Αθλητικά, Ψυχαγωγία), έτσι αν συσχετιστεί μια τάση με άρθρο τότε κληρονομείται η κατηγορία του άρθρου. Το περιεχόμενο των τάσεων τις περισσότερες φορές αφορά αθλητικά γεγονότα, ακολουθούν

³³ Η chart.js είναι μια απλή και ευέλικτη βιβλιοθήκη γραμμένη σε JavaScript: <https://www.chartjs.org/>

³⁴ Περισσότερες πληροφορίες για διάγραμμα τύπου line: <https://www.chartjs.org/docs/latest/charts/line.html>

³⁵ Περισσότερες πληροφορίες για διάγραμμα τύπου bar: <https://www.chartjs.org/docs/latest/charts/bar.html>

τα οικονομικά ζητήματα, πολιτικά, κοινωνικά και τέλος η ψυχαγωγία. Οι τάσεις που αφορούν αθλητικά γεγονότα κυριαρχούν στις κορυφαίες τάσεις κατά την διάρκεια κάποιου αθλητικού αγώνα ή σημαντικού αθλητικού γεγονότος καθώς και λίγες ώρες μετά. Η κατηγορία ψυχαγωγία αφορά κυρίως τηλεοπτικές σειρές, τηλεπαιχνίδια της τηλεόρασης καθώς και άλλες εκπομπές διασκέδασης της τηλεόρασης και εμφανίζονται κυρίως τις ώρες που προβάλλονται. Οι κατηγορίες Αθλητικά και Ψυχαγωγία συνήθως έχουν τάσεις που επαναλαμβάνονται για αρκετό καιρό καθώς οι αθλητικοί αγώνες, τηλεοπτικές εκπομπές κ.α. επαναλαμβάνονται. Οι υπόλοιπες κατηγορίες συνήθως εξελίσσονται στον χρόνο και δεν μένουν στις τάσεις εντός κάποιου συγκεκριμένου χρονικού περιθωρίου.

Έχει ενδιαφέρον να δούμε μία τάση η οποία παρουσιάστηκε από τα ειδησεογραφικά καθυστερημένα. Ένα παράδειγμα που βρέθηκε από το παρόν εγχείρημα είναι η τάση **#ζαραλικος**. Τα tweets μιλούσαν για το περιστατικό διακοπής της παράστασης του γνωστού κωμικού. Το συμβάν συνέβη στις 2/12/21 το βράδυ και αναφέρθηκε μόνο από ένα ειδησεογραφικό στις 14:00 το μεσημέρι. Πιθανώς το άρθρο να δημοσιεύθηκε διότι το hashtag #ζαραλικος ήταν πρώτο στις τάσεις από τα ξημερώματα καθώς άλλα ειδησεογραφικά είχαν δημοσιεύσει το γεγονός λίγο μετά από το συμβάν και αυτό να άσκησε κάποια πίεση. Ωστόσο μόνο ένα ειδησεογραφικό αναφέρθηκε στο γεγονός και τα υπόλοιπα δεν ανέρτησαν την είδηση ποτέ. Υπάρχουν αρκετές περιπτώσεις που διάφορα περιεχόμενα από τις τάσεις που δεν αναφέρθηκαν καθόλου στα ειδησεογραφικά. Η εφαρμογή ήταν ενεργή για διάστημα δύο μηνών (Νοέμβρης, Δεκέμβρης) και σε αυτή την περίοδο εντοπίστηκαν τέτοιες τάσεις/hashtags. Περιληπτικά φτιάχτηκε ένας πίνακας με τον τίτλο της τάσης και σύντομη περιγραφή του τι γράφουν τόσο τα ειδησεογραφικά όσο και τα tweets.

Hashtag	Twitter	Ειδησεογραφικά
#παιδωναγισσοφια	Οι χρήστες μιλάνε για την χρόνια υπονόμηση της Παιδοκαρδιοχειρουργικής μονάδας του νοσοκομείου Αγία Σοφία. Η συγκριμένη τάση εμφανίζεται σχεδόν κάθε μέρα	Δεν αναφέρεται καμία είδηση.
#cancel_chatzinikolaou	Κατακραυγή για τον γνωστό δημοσιογράφο, λόγω της δημοσίευσης άρθρου του περί «Τρολς του διαδικτύου». Οι χρήστες γράφουν για λογοκρισία από την Κυβέρνηση.	Δεν αναφέρεται καμία είδηση.
#με_τον_ιασονα	Διασώστης προσφύγων έδωσε συνέντευξη στην εκπομπή «Καλύτερα» και το Twitter γέμισε υποστηρικτικά σχόλια για τον εν λόγω διασώστη, την	Δεν αναφέρεται καμία είδηση.

	παρουσιάστρια της εκπομπής και το έργο του.	
#Μητσοτακη_που_ειναι_τα_παιδια	Αναφέρονται στο μεγάλο πλήθος αγνοούμενων προσφυγόπουλων στην Ελλάδα. Και κυρίως στην αποσιωποίηση από τα ΜΜΕ.	Δεν αναφέρεται καμία είδηση.
μαρινακι	Μιλάνε για οικονομικό έγκλημα, περισσότερες πληροφορίες εδώ: https://t.co/j8K3I0JHJ6	Δεν αναφέρεται καμία είδηση.
#μαρινακης	Συνοδευόταν με το hashtag #ζαραλικος και αφορούσε το ίδιο γεγονός.	Υπάρχει μόνο η είδηση από την kathimerini.gr: https://www.kathimerini.gr/society/561613669/diekopsan-tin-parastasi-zaralikoy-oikonomoy-katadikastea-kathe-prospatheia-viaiiis-parempodisis-tis-kallitechnikis-ekfrasis/
#4000_παπαδες	Κατακραυγή για την απόφαση της ΝΔ να προσλάβει 4000 παπάδες αντί για γιατρούς, πυροσβέστες κλπ.	Δεν αναφέρεται καμία είδηση.

Βιβλιογραφία

C. Castillo, M. Mendoza, and B. Poblete, 2011, INFORMATION CREDIBILITY ON TWITTER, In Proceedings of the 20th international conference on World wide web,
Available at: <https://dl.acm.org/doi/10.1145/1963405.1963500>
[Accessed at 22 December 2021]

Rasmus Kleis Nielsen, October 2012, TEN YEARS THAT SHOOK THE MEDIA WORLD, Reuters Institute for the Study of Journalism,
Available at: https://reutersinstitute.politics.ox.ac.uk/sites/default/files/2017-09/Nielsen%20-%20Ten%20Years%20that%20Shook%20the%20Media_0.pdf
[Accessed at 8 March 2022]

Arkaitz Zubiaga - Maria Liakata - Rob Procter, 2017, EXPLOITING CONTEXT FOR RUMOUR DETECTION IN SOCIAL MEDIA, SocInfo,
Available at: <https://www.semanticscholar.org/paper/Exploiting-Context-for-Rumour-Detection-in-Social-Zubiaga-Liakata/6de66a114f2efb24d3d21c3742c47d7d0f6cc521>
[Accessed at 23 December 2021]

Μουζακίτη Μαρία, 2012, *TO TWITTER ΣΤΗΝ ΕΛΛΑΔΑ ΩΣ ΟΧΗΜΑ ΔΙΚΤΥΩΣΗΣ, ΠΑΡΑΚΙΝΗΣΗΣ ΚΑΙ ΑΥΤΟ-ΟΡΓΑΝΩΣΗΣ ΚΟΙΝΟΤΗΤΩΝ*, Πάντειο Πανεπιστήμιο Κοινωνικών και Πολιτικών Επιστημών, Available at: <http://pandemos.panteion.gr/index.php?op=record&lang=el&pid=iid:6672>
[Ανακτήθηκε 27 Ιανουαρίου 2022]

Peter Willett, 1 July 2006, THE PORTER STEMMING ALGORITHM: THEN AND NOW, Program: electronic library and information systems Vol. 40 Iss 3 pp. 219 - 223,
Available at: <https://www.emerald.com/insight/content/doi/10.1108/00330330610681295/full/html>
[Accessed at 1 February 2022]

Michael Buckland - Fredric Gey, January 1994, THE RELATIONSHIP BETWEEN RECALL AND PRECISION, Journal of the American Society for Information Science Volume 45, Issue 1
Available at: [https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/\(SICI\)1097-4571\(199401\)45:1%3C12::AID-ASI2%3E3.0.CO;2-L](https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/(SICI)1097-4571(199401)45:1%3C12::AID-ASI2%3E3.0.CO;2-L)
[Accessed at 3 February 2022]

Deepu S. - Pethuru Raj - S.Rajaraajeswari, A FRAMEWORK FOR TEXT ANALYTICS USING THE BAG OF WORDS (BOW) MODEL FOR PREDICTION, International Journal of Advanced Networking & Applications (IJANA),

Available at: [https://www.ijana.in/Special Issue/S71.pdf](https://www.ijana.in/Special%20Issue/S71.pdf)
[Accessed at 3 February 2022]

Tomas Mikolov - Kai Chen - Greg Corrado - Jeffrey Dean, 16 Jan 2013, EFFICIENT ESTIMATION OF WORD REPRESENTATIONS IN VECTOR SPACE, Cornell University
Available at: <https://arxiv.org/abs/1301.3781>
[Accessed at 7 February 2022]

Fernando Doglio, 2015, PRO REST API DEVELOPMENT WITH NODE.JS, Apress Media, DOI 10.1007/978-1-4842-0917-2
[Accessed at 14 February 2022]

Ireton Cherilyn, Posetti Julie, 2021, ΔΗΜΟΣΙΟΓΡΑΦΙΑ, “ΨΕΥΔΕΙΣ ΕΙΔΗΣΕΙΣ” & ΠΑΡΑΠΛΗΡΟΦΟΡΗΣΗ: ΕΓΧΕΙΡΙΔΙΟ ΔΗΜΟΣΙΟΓΡΑΦΙΚΗΣ ΕΚΠΑΙΔΕΥΣΗΣ ΚΑΙ ΚΑΤΑΡΤΙΣΗΣ, UNESCO - Cyprus University of Technology - Panteion University of Social and Political Sciences, ISBN: 978-92-3-000150-6
Διαθέσιμο εδώ: <https://unesdoc.unesco.org/ark:/48223/pf0000380166>
[Ανακτήθηκε στις 24 Φεβρουαρίου 2022]

Κεντερελίδου Κλειώ, 2009, ΤΗΛΕΟΠΤΙΚΑ ΚΕΝΤΡΙΚΑ ΔΕΛΤΙΑ ΕΙΔΗΣΕΩΝ ΚΑΙ ΚΥΒΕΡΝΗΣΗ: Η ΕΡΡΗΞΗ ΕΝΗΜΕΡΩΣΗΣ ΚΑΙ ΠΡΟΠΑΓΑΝΔΑΣ ΣΤΙΣ ΕΙΔΗΣΕΙΣ, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης (ΑΠΘ). Τμήμα Δημοσιογραφίας και Μέσων Μαζικής Ενημέρωσης, DOI [10.12681/eadd/20124](https://doi.org/10.12681/eadd/20124)
Διαθέσιμο εδώ: <https://phdtheses.ekt.gr/eadd/handle/10442/20124>
[Ανακτήθηκε στις 25 Φεβρουαρίου 2022]

Υπερσύνδεσμοι

<https://www.tweepy.org/>
<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>
<https://lxml.de/3.2/elementsoup.html>
<https://developer.mozilla.org/en-US/docs/Web/HTML>
https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction
<https://www.hp.com/us-en/shop/tech-takes/how-does-the-internet-work>
https://github.com/amueller/word_cloud
<https://www.statista.com/statistics/617136/digital-population-worldwide/>
<https://www.selenium.dev/>
<https://mongoosejs.com/>
<https://pymongo.readthedocs.io/en/stable/>
<https://pandas.pydata.org/>
<https://nodejs.org/en/>
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions
https://github.com/explosion/spaCy/blob/master/spacy/lang/el/stop_words.py
https://github.com/alup/python_greek_stemmer
<https://radimrehurek.com/gensim/>
https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
<https://expressjs.com/>
<http://expressjs.com/en/5x/api.html#app.get.method>
<https://expressjs.com/en/api.html>
<https://262.ecma-international.org/6.0/>
<https://day.js.org/>
<https://www.npmjs.com/package/dotenv>
https://www.w3schools.com/html/html_urlencode.asp
<http://expressjs.com/en/resources/middleware/body-parser.html>
<https://www.npmjs.com/package/cors>
<https://www.docker.com/>

[https://developer.mozilla.org/en-US/docs/Web/API/Document Object Model](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model)

<https://www.chartjs.org/>

<https://www.chartjs.org/docs/latest/charts/line.html>

<https://www.chartjs.org/docs/latest/charts/bar.html>

Παράρτημα - User Manual

Η εφαρμογή είναι σε μορφή ιστοσελίδας οπότε είναι διαθέσιμη μέσω της χρήσης φυλλομετρητή (browser). Εφόσον ο server NodeJS, η βάση δεδομένων MongoDB και ο python αλγόριθμος εκτελούνται στο σύστημα τότε η εφαρμογή θα ακούει (listens) σε μία συγκεκριμένη διεύθυνση URL στον φυλλομετρητή. Αν εκτελεστεί σε τοπικό σύστημα τότε η διεύθυνση πιθανός να είναι η “localhost” ή η τοπική διεύθυνση του συστήματος. Η θύρα που είναι διαθέσιμη η ιστοσελίδα έχει οριστεί στην “5000”. Οπότε στο σύνολο της διεύθυνσης URL της εφαρμογής εδώ, είναι η “**http://localhost:5000**”.

Η εξαγωγή αποτελεσμάτων γίνεται αυτόματα και δεν χρειάζεται καμία ενέργεια από την πλευρά του χρήστη. Ωστόσο για να εμφανιστούν τα πρώτα αποτελέσματα θα πρέπει να αφεθεί το πρόγραμμα για κάποια χρονική διάρκεια (τουλάχιστον 2-4 ώρες) σε εκτέλεση ώστε να συλλεχθούν τα δεδομένα, να αναλυθούν και να αποθηκευτούν στην βάση δεδομένων. Μετά την αναμονή, θα πρέπει να εμφανίζεται η ιστοσελίδα σωστά. Στην κορυφή της ιστοσελίδας όπως φαίνεται και στην παρακάτω εικόνα βρίσκονται οι τελευταίες πέντε τάσεις που κατέγραψε η εφαρμογή και οι συσχετίσεις τους με τα άρθρα. Στον πίνακα (2) βρίσκονται οι τάσεις, στον πίνακα (3) τα κορυφαία tweets της επιλεγμένης τάσης και τέλος στον πίνακα (4) βρίσκονται τα πιο κοντινά συσχετισμένα άρθρα από κάθε ειδησεογραφικό (protothema.gr, enikos.gr, in.gr, Kathimerini.gr, skai.gr).

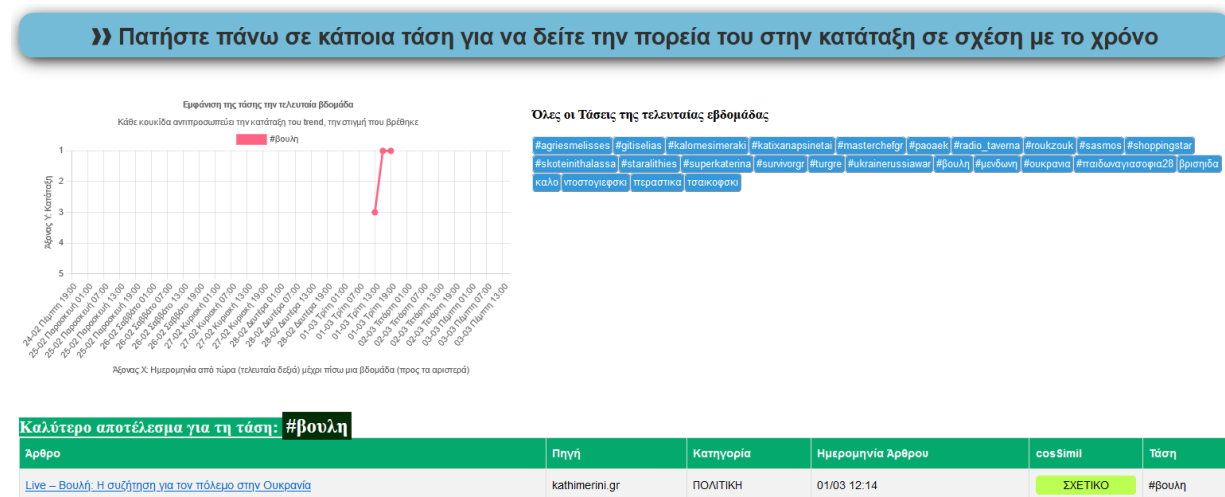
The screenshot displays the application's main interface with several key sections:

- Top Left:** A calendar view for 03/01/2022 with a time range of 00:56 to 16:42. A red circle '1' highlights the date selection area.
- Top Center:** A section titled 'Τάσεις σε Ελλάδα' (Trends in Greece) listing five trending topics: #bouli, #oukrania, #menedoni, #ukrainerussiawar, and #tsaikofski. A red circle '2' highlights the list.
- Top Right:** A section titled 'Top Tweet του hashtag' (Top Tweet of the hashtag) showing tweets from communakios, NIKSOTIROPOYLOS, BettyTheia, and ellada24. A red circle '3' highlights the tweet list.
- Bottom Left:** A section titled 'Ειδήσεις που βρέθηκαν:' (News found:) listing five news items related to the #bouli hashtag. A red circle '4' highlights the news list.
- Bottom Center:** A section titled 'Βαθμολογία σχετικότητας άρθρου σε σχέση επιλεγμένης τάσης' (Article relevance rating relative to selected trend) and 'Καθυστέρηση ημερομηνίας δημοσίευσης ειδήσης' (News publication date delay).

Εικόνα 10 Η αρχική εικόνα και κύρια λειτουργία της διεπαφής. Υπάρχουν επισημειωμένα διάφορα στοιχεία που επεξηγούν τις λειτουργίες.

Κάτω αριστερά (1) ο χρήστης μπορεί να επιλέξει παρελθοντικά αποτελέσματα, αυτό περιλαμβάνει τις τότε πρώτες πέντε τάσεις, tweets και συσχετισμένα άρθρα. Επιλέγοντας μία διαθέσιμη ημέρα από την εφαρμογή, τότε πρέπει να επιλεγεί μία από τις χρονικές σημάνσεις που θα εμφανιστούν στο πράσινο πλαίσιο. Στην συνέχεια επιλέγοντας στον πίνακα (2) κάποια τάση θα ενημερωθούν οι πίνακες (3,4) με τα αντίστοιχα tweets της τάσης και τα αντίστοιχα σχετικά άρθρα της τάσης που βρέθηκαν εκείνη την στιγμή. Στον πίνακα (3) ο χρήστης μπορεί να πατήσει σε ένα tweet και να μεταφερθεί στην πλατφόρμα Twitter με το συγκεκριμένο tweet που πάτησε. Αντιστοίχως στον πίνακα (4) ο χρήστης μπορεί να πατήσει πάνω σε ένα άρθρο και να μεταφερθεί στο ειδησεογραφικό με το συγκεκριμένο άρθρο που πάτησε.

Στην συνέχεια λίγο πιο κάτω από την κορυφή της σελίδας υπάρχει η επιλογή στον χρήστη να δει την πορεία κατάταξης κάποιας τάσεως σε σχέση με τον χρόνο.



Εικόνα 11 Χρονοδιάγραμμα των τάσεων έτσι όπως εμφανίζεται στην εφαρμογή. Ο χρήστης πατάει σε κάποια τάση και του εμφανίζεται η πορεία της κατάταξης της τάσης σε σχέση με τον χρόνο.

Ο χρήστης μπορεί να πατήσει κάποια τάση στα δεξιά (εμφανίζονται μόνο οι τελευταίες τάσης της εβδομάδας), και να του εμφανιστεί το χρονοδιάγραμμα της τάσης. Το χρονοδιάγραμμα στον κάθετο άξονα έχει την κατάταξη της τάσης (πρώτη μέχρι πέμπτη) και στο οριζόντιο άξονα έχει τον χρόνο (από την στιγμή που το βλέπει ο χρήστης έως μία εβδομάδα πίσω) χωρισμένο ανά τρεις ώρες. Στο κάτω μέρος εμφανίζεται το καλύτερο αποτέλεσμα για την επιλεγμένη τάση, δηλαδή την καλύτερη συσχέτιση (πιο κοντινό άρθρο) που βρέθηκε για αυτήν.

Πιο κάτω από αυτό υπάρχει μηχανή αναζήτησης μέσα στην βάση δεδομένων. Έτσι όπως παρουσιάζεται στην παρακάτω εικόνα, ο χρήστης πατάει ένα κλειδί ή κλειδιά στην φόρμα εισαγωγής κλειδιών (1). Στην συνέχεια επιλέγει αν ψάχνει άρθρο ή τάση (προεπιλεγμένο να ψάχνει άρθρα) και αναλόγως του επιστρέφονται τα αποτελέσματα που βρέθηκαν. Δηλαδή άμα ψαχθούν άρθρα τότε θα δοθεί η καλύτερη συσχέτιση του με κάποια τάση και αντιστρόφως, αν ψαχθούν τάσεις θα του δοθεί το πιο κοντινό άρθρο. Επιπλέον δίνεται στον χρήστη η επιλογή να ταξινομήσει τις στήλες. Οι στήλες, είναι ο τίτλος του άρθρου (Άρθρο), η πηγή του άρθρου δηλαδή η ηλεκτρονική πύλη του ειδησεογραφικού (Πηγή), η κατηγορία του άρθρου έτσι όπως την κατέγραψε κάθε ειδησεογραφικό (Κατηγορία), η ημερομηνία που δημοσιεύθηκε το άρθρο (Ημερ. Άρθρου), η σχετικότητα μεταξύ του άρθρου και της τάσης (Σχετικότητα), με πέντε διαφορετικές διακυμάνσεις:

1. **ΜΗ-ΣΧΕΤΙΚΟ** δηλαδή δεν βρέθηκε καμία σημαντική συσχέτιση

2. **ΛΙΓΟ-ΣΧΕΤΙΚΟ** δηλαδή βρέθηκε κάποια μικρή συσχέτιση
3. **ΣΧΕΤΙΚΟ** δηλαδή βρέθηκε επαρκώς κάποια σχετικότητα
4. **ΠΟΛΥ-ΣΧΕΤΙΚΟ** δηλαδή βρέθηκε με μεγαλύτερη σιγουριά σχετικότητα
5. **ΕΞΑΙΡΕΤΙΚΑ-ΣΧΕΤΙΚΟ**, εδώ η εφαρμογή βρήκε με μεγάλο βαθμό ομοιότητα μεταξύ άρθρου και τάσης και μειώθηκε εξαιρετικά πολύ η πιθανότητα να κάνει λάθος

1
2
3

☒ άρθρο ☐ τάση

Άρθρο	Πηγή	Κατηγορία	Ημερ. Άρθρου	Σχετικότητα	Τάση
Ουκρανία: Γιατί ο Πούτιν έχει ήδη χάσει αυτόν τον πόλεμο – Η ανάλυση διάσημου ιστορικού	enikos.gr	international	28/02 18:21	ΣΧΕΤΙΚΟ	#ukrainerussiawar
Δείτε live τη συζήτηση στη Βουλή για τις επιπτώσεις του πολέμου στην Ουκρανία στην Ελλάδα	in.gr	Βουλή	01/03 11:14	ΣΧΕΤΙΚΟ	#βουλη
Τσίπρας για πόλεμο στην Ουκρανία: Υπέρ των κυρώσεων, κατά της αποστολής πολεμικού υλικού	in.gr	Βουλή	01/03 12:23	ΣΧΕΤΙΚΟ	#βουλη
Live – Βουλή: Η συζήτηση για τον πόλεμο στην Ουκρανία	kathimerini.gr	ΠΟΛΙΤΙΚΗ	01/03 12:14	ΣΧΕΤΙΚΟ	#βουλη
Πόλεμος στην Ουκρανία: Αντίποινα της Ρωσίας σε 36 χώρες – Κλείνει τον εναέριο χώρο της	kathimerini.gr	ΚΟΣΜΟΣ	28/02 16:35	ΣΧΕΤΙΚΟ	#ukrainerussiawar
Ο πόλεμος στην Ουκρανία μέσα από τα πρωτοσέλιδα του διεθνούς τύπου	kathimerini.gr	ΚΟΣΜΟΣ	28/02 19:08	ΣΧΕΤΙΚΟ	#ukrainerussiawar
Live: Πέμπτη ημέρα πολέμου στην Ουκρανία – Όλες οι εξελίξεις	kathimerini.gr	ΚΟΣΜΟΣ	28/02 08:34	ΣΧΕΤΙΚΟ	#ukrainerussiawar
Πόλεμος στην Ουκρανία: Εισόδος στη νέα παγκόσμια αταξία	kathimerini.gr	ΚΟΣΜΟΣ	28/02 14:50	ΣΧΕΤΙΚΟ	#ουκρανια
Αλ. Τσίπρας: Η Ελλάδα μετατρέπεται σε εμπλεκόμενη του πολέμου	kathimerini.gr	ΠΟΛΙΤΙΚΗ	01/03 13:54	ΣΧΕΤΙΚΟ	τσαικοφσκι
Βουλή: Στην Ολομέλεια ο πόλεμος στην Ουκρανία κι οι επιπτώσεις του για την Ελλάδα	kathimerini.gr	ΒΟΥΛΗ	01/03 08:10	ΣΧΕΤΙΚΟ	#βουλη

Εικόνα 12 Υλοποιημένη μηχανή αναζήτησης που είναι διαθέσιμη στην εφαρμογή αυτού του εγχειρήματος.

Παρακάτω από την μηχανή αναζήτησης βρίσκονται τρεις πίνακες, έτσι όπως φαίνονται στην εικόνα παρακάτω.

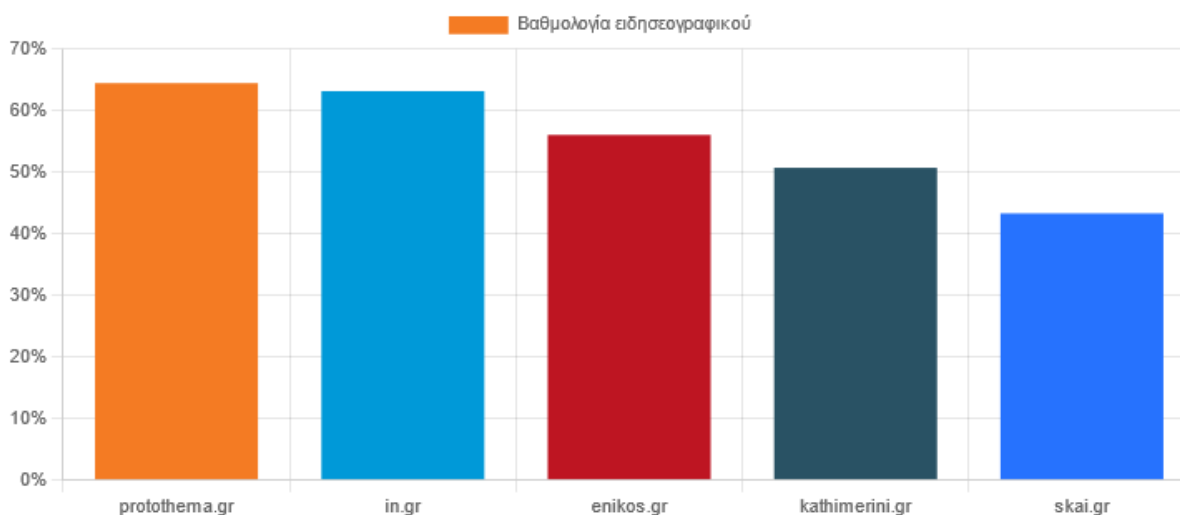
Οι τάσεις που βρέθηκαν με μεγάλη σχετικότητα Πιθανός το άρθρο να δείχνει για ποιο πράγμα μιλάει η τάση	Οι τάσεις που επαναλαμβάνονται Τάσεις των τελευταίων 3 ημερών που έχουν ξαναφανερωθεί στο παρελθόν	Οι καινούργιες τάσεις των τελευταίων 3 ημερών Δείχνει τι θέματα απασχολούν το twitter τις τελευταίες 3 μέρες
καλο MasterChef: Τα δημοφιλή γεμιστά δοτεύουν τρεις στον τόκο – Αυτοί είναι οι υποψήφιοι προς απογόρευση #παιδοναγισσοφια28 Τρία χρόνια Citizenship: Η Γιάννα Δελιγιάννη μας καλεί να σπουδάσουμε κυριαρχία γάμου βρισηδα Survivor: Βόσνιος Μπίλε η ασύζα – Υποψήφια η Βρεσηδα	περαστικα Κενάβη: Θυμάρ με νεκρό άνδρα #μενδωνη Ο Τσούτσουλας απαγορεύτηκε ή όχι ακόμα; – Χαμός με την απόφαση των υπουργείων Πολιτισμού να ακυρώσει τη «Μηνά των Κόκκων» #gitisellias Η Γη της: Ελάος: Η Αρετή αναζητεί να πιάει τον Σπύρη να το βρει #agriesmelisses Άγριας Μίσους: Η Ελένη θα αποκαλύψει στον Δάμπερο ένα #masterchefgr MasterChef: Απαγορεύτηκε ο Δημήτρης – «Κου» το λένε – BINTED #sasmos Σωμός: Ένα σόδα για την εξόντωση του Μαθίου βρίσκεται στα παρτά από τους Βρενιόκηδες	τσαικοφσκι Ο Τσούτσουλας απαγορεύτηκε ή όχι ακόμα; – Χαμός με την απόφαση των υπουργείων Πολιτισμού να ακυρώσει τη «Μηνά των Κόκκων» #βουλη Live – Βουλή: Η συζήτηση για τον πόλεμο στην Ουκρανία #ukrainerussiawar Το Ρωσοουκρανικό «casus belli» #ουκρανια Βελζέλος: Η απλή γρήση πυρηνικών από την Ρωσία, προκαλεί ντοστογιεφσκι 500 λέξεις με την Αλεξάνδρα Γρηγορίου

Εικόνα 13 Αριστερά είναι ο πίνακας με τάσεις που βρέθηκαν με μεγάλη ομοιότητα με κάποιο άρθρο. Στο κέντρο παρουσιάζονται τάσεις των τελευταίων ημερών που επαναλήφθηκαν και δεξιά τάσεις των τελευταίων τριών ημερών που είναι καινούργιες.

Ο αριστερός πίνακας αφορά τάσεις που βρέθηκαν με μεγάλη ομοιότητα με κάποιο άρθρο. Αν ένα άρθρο έχει σημειώσει μεγάλη βαθμολογία τότε είναι πολύ πιο πιθανό να είναι πιο κοντά εννοιολογικά με την τάση που έχει συσχετιστεί, από ότι τα άρθρα που έχουν σημειώσει μικρότερη βαθμολογία. Ο κεντρικός πίνακας αφορά τις τάσεις των τελευταίων τριών ημερών που έχουν ξαναεμφανιστεί στο παρελθόν. Ο δεξιά πίνακας αφορά τάσεις των τελευταίων τριών ημερών που δεν επαναλαμβάνονται, με αποτέλεσμα να προβάλλονται τα θέματα που απασχολούν το Twitter τις τελευταίες τρεις μέρες. Σε όλους τους πίνακες υπάρχουν ζεύγη της τάσης με το πιο σχετικό τίτλο άρθρου που βρέθηκε από την επεξεργασία δεδομένων.

Όσο περνάει ο χρόνος η εφαρμογή συλλέγει πολλές συσχετίσεις από όλα τα ειδησεογραφικά. Για κάθε άρθρο των ειδησεογραφικών υπάρχει συνδεδεμένη κάποια τάση με μία συγκεκριμένη βαθμολογία. Αν πάρουμε αυτές τις συνδέσεις που έχουν υψηλή βαθμολογία (άρα και μεγαλύτερη πιθανότητα να υπάρχει πραγματική συσχέτιση άρθρου-τάσης) και τις αθροίσουμε για κάθε ξεχωριστό ειδησεογραφικό, παίρνουμε την βαθμολογία του ειδησεογραφικού στο σύνολο του. Κατά κάποιο τρόπο δηλαδή μετριέται κατά πόσο ακολουθούν τα ειδησεογραφικά τις τάσεις του Twitter. Τα αποτελέσματα που προβάλλονται στην παρακάτω εικόνα βασίζονται σε αποτελέσματα που υπολογίστηκαν σε αυτό το εγχείρημα με βάση 46.000 άρθρα και 220.000 tweets μεταξύ του Νοέμβρη και Δεκέμβρη του έτους 2021.

Ποσοστό των άρθρων που βρέθηκαν με μεγάλη σχετικότητα με κάποια τάση.



Εικόνα 14 Ποσοστό των άρθρων που βρέθηκαν με μεγάλη σχετικότητα με κάποια τάση.

Παράρτημα – Deployment

Για την εγκατάσταση της εφαρμογής και όλων των απαιτήσεων της είναι απαραίτητη η εγκατάσταση της Docker πλατφόρμας καθώς και σύνδεση στο διαδίκτυο. Η Docker πλατφόρμα είναι διαθέσιμη για Windows, Mac και Linux³⁶. Εφόσον εγκατασταθεί και εκτελείται στο σύστημα πηγαίνουμε στον φάκελο που βρίσκεται το αρχείο **docker-compose-example.yml** και το μετονομάζουμε σε **docker-compose.yml**. Σε αυτό το αρχείο έχουν ήδη εισαχθεί οι IP διευθύνσεις, αν στο δικό σας σύστημα δεν δουλεύει με αυτές τις διευθύνσεις τότε πρέπει να τις αλλάξετε με αυτές την IP διεύθυνση του docker-engine σας.

Στην περίπτωση που είναι διαφορετική η IP διεύθυνση σας, στην μεταβλητή **subnet** βάζουμε την τοπική IP διεύθυνση του docker-engine, και για όλα τα containers (nodeserver, mongodb, pythonscript) στη μεταβλητή **ipn4_address** εισάγουμε την IP διεύθυνση με αυτή του subnet (Σ.τ.Σ. κάθε IP container πρέπει να έχει διαφορετικό HOST ID, δηλαδή το τελευταίο νούμερο της IP διεύθυνσης).

Στην συνέχεια, στο αρχείο **nodeServer/config/config.env** πρέπει να αλλάξετε το κομμάτι της IP στην μεταβλητή **MONGO_URI** με αυτή που εισάγετε στο container mongodb.

Το ίδιο πρέπει να γίνει για την μεταβλητή **DATABASE_URL** στο αρχείο **python/dev.env**

Τέλος πρέπει να εισαχθούν τα δικά σας κλειδιά για το Twitter API, στο αρχείο **python/getTweets/searchTweets.py** και **python/getTopHashtags/getLiveTrends.py** στις παραμέτρους

```
auth = tweepy.OAuthHandler("your_consumer_key", "your_consumer_secret")
auth.set_access_token("your_key", "your_secret")
```

Πλέον το μόνο που πρέπει να γίνει είναι να εκτελεστεί η εντολή

```
docker-compose up
```

στο φάκελο που υπάρχει το αρχείο docker-compose.yml μέσω του terminal και σύντομα (μπορεί να πάρει κάποια λεπτά) αφού εγκατασταθούν (αυτόματα) όλα τα κομμάτια της εφαρμογής, η εφαρμογή θα τρέχει στην IP και θύρα **localhost:5000** (αν το σύστημα είναι τοπικό).

Συνήθως η mongoDB έχει προκαθορισμένη την θύρα 27017. Η οποία χρησιμοποιείτε ήδη από το Docker, για να λειτουργήσει η εφαρμογή πρέπει να σταματήσει το service της mongoDB (αν το έχετε εγκαταστημένο στον υπολογιστή σας και είναι ενεργό). Για τα Windows οι εντολές σε PowerShell:

```
net stop MongoDB //για να σταματήσει το service
net start MongoDB //για να ξαναρχίσει το service
```

Αν θέλουμε να σταματήσουμε την εφαρμογή απλά κλείνουμε το docker-engine ή σταματάμε τα containers της εφαρμογής με την εντολή (στο φάκελο που βρίσκεται το docker-compose.yml) σε terminal:

```
docker-compose down
```

³⁶ <https://www.docker.com/get-started>

Παράρτημα – Source Code

Όλη η εφαρμογή είναι διαθέσιμη σε δημόσιο αποθετήριο στο link: <https://github.com/sakis475/thesis-A.-Piplikatsis>

Τα σημαντικότερα κομμάτια κώδικα της εφαρμογής:

python/init2.py

Από εδώ ξεκινάει η επεξεργασία των δεδομένων και όλες οι επιμέρους διαδικασίες.

```
##  
#DOCKER PYTHON SCRIPT  
#####  
  
from init import init  
import sys  
sys.path.append('database/results')  
sys.path.append('database/helpers')  
sys.path.append('database/stats')  
sys.path.append('stats')  
from resultsCollection import *  
from timeScriptStoppedCollection import *  
from resultsLastCollection import *  
from resultsLastAllCollection import *  
from getTimeNow import getDate  
from findPopularTweets import findPopularTweetsOfHashtags  
from calcSimilarityTrend import *  
from topSimilTrendsCollection import *  
  
from countStatsCollection import *  
from countStats import *  
  
from sourcesScoreCollection import *  
from sumSourcesScore import *  
  
import pandas as pd  
pd.options.mode.chained_assignment = None  
  
import time, datetime
```

```

import traceback
import logging

#Προσπαθεί να συνδεθεί με τα ειδησεογραφικά
def checkInternetConnection():
    import requests
    urls = ['https://www.skai.gr', 'https://www.protothema.gr',
'https://feeds.feedburner.com/kathimerini/DJpy', 'https://www.in.gr', 'https://www.enikos.gr']

    timeout = 5

    try:
        for url in urls:
            requests.get(url, timeout=timeout)
        print('Connection succesfull to all endpoints!')
    except (requests.ConnectionError, requests.Timeout) as exception:
        print("No internet connection, or a website is failing")
        print(exception)
        raise NameError('No internet connection, or a website is failing')

while True:
    #Ο χρόνος που η εφαρμογή θα αναμένει μετά το πέρας της ολοκλήρωσης του κύκλου του
    timeToSleep = 60*30

    #Αν αποτύχει η σύνδεση στα ειδησεογραφικά, τότε περιμένει 5 δευτερόλεπτα και επαναλαμβάνει μέχρι να
    συνδεθεί.
    try:
        print('Check Internet')
        checkInternetConnection()
    except Exception as e:
        print(e)
        time.sleep(5)
        continue

    try:
        #Η getDate() επιστρέφει την ημερομηνία του συστήματος την στιγμή που καλείται
        startDate = getDate()
        try:
            #Στην init() πραγματοποιείται όλη η διαδικασία της εξόρυξης γνώσης (κατέβασμα άρθρων, τάσεων,
            tweets και εξαγωγή αποτελεσμάτων)

```

```

        results = init()
    except BaseException as error:
        print('Program didnt finish init properly: {}'.format(error))
        raise NameError('Cant continue, program problem on init')

    #Αποθηκεύει τα αποτελέσματα στο collection resultsLastAll.
    saveResultsLastAllDB(results)

    #Βρίσκει όλες τις τάσεις των αποτελεσμάτων που έχουν βρεθεί με υψηλή συσχέτιση με κάποιο άρθρο και
    τα αποθηκεύει στην ΒΔ.
    calcSim = calcSimilarityTrend()
    saveTopSimilTrendsDB(calcSim)

    #Μετράει και αποθηκεύει στοιχεία της ΒΔ (πλήθος άρθρων, πλήθος tweets, πλήθος κύκλων που έτρεξε το
    πρόγραμμα, πλήθος μοναδικών τάσεων)
    saveStatsDB(gatherStats())

    #Μετράει και αποθηκεύει την βαθμολογία των ειδησεογραφικών
    saveSourcesScoreDB(calcSourcesScore())

    endDate = getDate()

    timeTook = endDate - startDate
    print('Time took: ', timeTook)
    print('Next results at: ', getDate() + datetime.timedelta(minutes=30))

    #Αν κάτι από όλα τα παραπάνω αποτύχει, περιμένει 15 λεπτά και ξαναδοκιμάζει από την αρχή.
    except BaseException as error:
        print('ERROR IN PROGRAM TRYING TO RESTART: {}'.format(error))
        print('sleeping 15 mins')
        timeToSleep = 60*15

    time.sleep(timeToSleep) #Αν ο κύκλος του προγράμματος επιτύχει τότε αναμένει 30 λεπτά και
    ξαναεκτελείται όλος ο κώδικας κάτω από την while loop μέχρι εδώ.

```

python/init.py

Το πιο σημαντικό κομμάτι κώδικα, καθώς εδώ πέρα πραγματοποιείτε όλη η διαδικασία, συλλογής δεδομένων, αποθήκευσης, καθαρισμού καθώς και εύρεσης συσχετίσεων μεταξύ άρθρων και τάσεων.

```
import sys
sys.path.append('getTopHashtags')
sys.path.append('database/trends')
sys.path.append('database/helpers')
sys.path.append('database/articles')
sys.path.append('database/tweets')
sys.path.append('database/results')
sys.path.append('getTweets')
from getLiveTrends import getLiveTrends
from articlesCollection import *
from trendsCollection import *
from getTimeNow import *
from routineCollection import *
from searchTweets import *
from tweetsCollection import *
from bowCosSim import *
from resultsLastCollection import *
from resultsLastAllCollection import *

import pandas as pd
pd.set_option('display.max_rows', 500)
pd.options.mode.chained_assignment = None

sys.path.append('scraperWebsite')
sys.path.append('database/articles')

from getArticles import getALLarticles
import time
import datetime

#Ελέγχει αν βρέθηκαν καινούργιες τάσεις που δεν υπάρχει αποθηκευμένη στην ΒΔ και τις επιστρέφει
def trendForDeepSearch(constDate, trendsTopFive):

    fromDate_results = constDate - datetime.timedelta(days = 7)
    tillDate_results = constDate

    threeDaysBackresults = getResultsLastAllDB(fromDate_results, tillDate_results)
    if(threeDaysBackresults.empty):
        return trendsTopFive['hashtag'].unique()
```

```

trendToDeepSearch = []
for trend in trendsTopFive['hashtag'].unique():
    if(not (trend in threeDaysBackresults['searchQuery'].unique())):
        #den yparxei to trend p vrethike twra, sta results!
        trendToDeepSearch.append(trend)
return trendToDeepSearch

def init():

    constDate = getDate()

    #Κατεβαίνουν οι πρώτες πέντε τάσεις από το Twitter API

    ### get hashtags ###
    trendsTopFive = getLiveTrends().sort_values(by=['rank']).head(5)
    saveTrendsDB(trendsTopFive)

    #Κατεβαίνουν tweets για αυτές τις πρώτες πέντε τάσεις

    #### get tweets ####
    tweetsToSearch = searchHashtagsWithinDay(trendsTopFive, trendsTopFive['dateDownloaded'].iloc[0])

    ##### DOWNLOAD articles #####

    #Η μεταβλητή επιλέγει πόσο πίσω να ψάξει για άρθρα στα ειδησεογραφικά με βάση την προηγούμενη φορά που
κατέβασε άρθρα.
    fromDate_lastDownloadArticles = getArticleDownloadDateLAST_DB() - datetime.timedelta(hours = 2)

    #Αποθηκεύονται όλα τα άρθρα στην μεταβλητή articles
    articles = getAllArticles(fromDate_lastDownloadArticles)

    #Αποθήκευση της ημερομηνίας που κατέβηκαν μόλις τώρα τα άρθρα.
    saveArticleDownloadDateDB( constDate )
    #Αποθήκευση των άρθρων που κατέβηκαν στην ΒΔ.
    saveArticlesDB(articles)

    ##### ΚΥΡΙΑ ΔΙΕΡΓΑΣΙΑ #####

    #Αρχικά αρχικοποιείται ένα άδειο dataframe που στην συνέχεια θα γεμίσει με τα αποτελέσματα του
αλγόριθμου

```

```

results = pd.DataFrame()

trendListDeepSearch = trendForDeepSearch(constDate , trendsTopFive)

#Για κάθε τάση, παίρνει τα tweets της και τα συγκρίνει με όλα τα άρθρα
for searchQuery in trendsTopFive['hashtag'].unique():

    groupTweetsOfHashtag = tweetsToSearch[tweetsToSearch['searchQuery'] == searchQuery]

    #Οι τάσεις που είναι καινούργιες για την εφαρμογή κάνουν εξοχυνιστική αναζήτηση σε άρθρα έως τρεις
    ημέρες πίσω.
    #Αλλιώς αν είναι τάση που έχει ξανασυγκριθεί με άρθρα τότε γίνεται αναζήτηση μόνο στα
    νεοσυλλεχθέντα άρθρα.
    if(searchQuery in trendListDeepSearch):
        print('DOING DEEP SEARCH FOR ', searchQuery, '')
        fromDate = constDate - datetime.timedelta(days = 3)
        TillDate = constDate
    else:
        fromDate = fromDate_lastDownloadArticles
        TillDate = constDate

    articles = getArticlesDB(fromDate,TillDate).sort_values(by='date')

    for source in articles['source'].unique():
        print('source: ', source, ', searchQuery: ', searchQuery)
        try:

            #Επιστρέφονται τα αποτελέσματα από την συσχέτιση άρθρων της δοθείσας πηγής
            ειδησεογραφικού (μία από τις πέντε, από την αρχή με την σειρά) και τα tweets μίας τάσης
            similResult = similarity(articles[articles['source'] == source], groupTweetsOfHashtag,
            searchQuery)

        except Exception as e:
            print('source ', source, ' with searchQuery: ', searchQuery, ', propably something
            wrong with tweet clean..', e)
            print('skipping')
            continue

        similResult['resultCompletedDate'] = constDate

        results = results.append(similResult)

#Ενα τελευταίο φιλτράρισμα στα αποτελέσματα.

```

#Για όλες τις πηγές ειδησεογραφικών, και για όλες τις διαφορετικές τάσεις επιλέγεται αυτό με το καλύτερο αποτέλεσμα

```
results_all_sources = pd.DataFrame()
```

```
sources = ['protothema.gr', 'enikos.gr', 'in.gr', 'kathimerini.gr', 'skai.gr']
```

```
searchQueryysToSearch = getLastTrendsInListDB()
```

```
for articleSource in sources:
```

```
    for searchQuery in searchQueryysToSearch:
```

```
        results_source = pd.DataFrame()
```

```
        results_source = results[results['articleSource'] == articleSource]
```

```
        results_source = results_source[results_source['searchQuery'] == searchQuery]
```

```
        results_source = results_source.sort_values(by='cosSimil', ascending = False).head(1)
```

```
        results_all_sources = results_all_sources.append(results_source, ignore_index=True)
```

```
return results_all_sources
```

```
##### ΤΕΛΟΣ ΚΥΡΙΑΣ ΔΙΕΡΓΑΣΙΑΣ #####
```

python/searchTweets.py

Εδώ αναζητούνται tweets από τις τάσεις μέσω του Twitter API και μετατρέπονται στην επιθυμητή μορφή.

```
import pandas as pd
import datetime

import time
import re

import sys
sys.path.append('../database/helpers')
from getTimeNow import *

def logging(e):
    import logging
    import os

    logFile = 'sample.log'
    logging.basicConfig( filename = logFile, filemode = 'w', level = logging.DEBUG, format = '%(asctime)s -
%(levelname)s: %(message)s', \
                        datefmt = '%m/%d/%Y %I:%M:%S %p' )

    logging.debug(e)

#Πραγματοποιεί σύνδεση στο Twitter API και αναζητεί tweets με συγκεκριμένο κλειδί αναζήτησης
def handleAuthTweeterAndSearch(query, max_tweets):

    import tweepy
    auth = tweepy.OAuthHandler("your_consumer_key", "your_consumer_secret")
    auth.set_access_token("your_key", "your_secret")

    api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)
    #tweepy.debug(False)
    time.sleep(1)

    #Αναζητούνται tweets για μία τάση, με τις επιλογές εξαίρεσης retweets,
    # τύπος αναζήτησης σε πρόσφατα (recent) tweets, το mode των tweet να επιστρέφει το πλήρες κείμενο
    (extended), και να επιλέγει tweets στα Ελληνικά.
    #Τα max_tweets είναι το μέγιστο όριο tweets που ζητούνται να κατεβαστούν.
    searched_tweets = [status._json for status in tweepy.Cursor(api.search, q=query+'-filter:retweets',
result_type="recent", tweet_mode="extended", lang='el').items(max_tweets)]

    return searched_tweets
```

```

#Κατεβάζει tweets με συγκεκριμένο κλειδί αναζήτησης και μέγιστο πλήθος
def searchTweets(query, max_tweets):

    tweetsData = pd.DataFrame()

    try:

        searched_tweets = handleAuthTweeterAndSearch(query, max_tweets)

    except Exception as e:
        #second try
        time.sleep(30)
        searched_tweets = handleAuthTweeterAndSearch(query, max_tweets)

        logging(e)

    #Μετατροπή των δεδομένων στην επιθυμητή μορφή
    for searchFound in searched_tweets:

        search_tweetsFullText = searchFound["full_text"]
        search_tweetID = searchFound["id_str"]
        created_at = searchFound["created_at"]
        created_at = datetime.datetime.strptime(created_at, '%a %b %d %H:%M:%S %z %Y')
        created_at = created_at.strftime('%Y-%m-%d %H:%M:%S')
        created_at = datetime.datetime.strptime(created_at, '%Y-%m-%d %H:%M:%S')
        dateSearched = getDate()

        if search_tweetsFullText[0:2] != "RT":
            hashtags = re.findall("#([\w_-]+)", search_tweetsFullText)
            tweetsData = tweetsData.append({'dateSearched': dateSearched, "date": created_at, "fullText":
search_tweetsFullText, "tweetID": search_tweetID, "userID": searchFound["user"]["id_str"],
"userScreenName": searchFound["user"]["screen_name"], "RT": False, "RTCount":
searchFound["retweet_count"], "searchQuery": query, "hashtags": hashtags}, ignore_index=True, sort=False)
            elif search_tweetsFullText[0:2] == "RT":
                searchHelper = searchFound["retweeted_status"]["full_text"]
                hashtags = re.findall("#([\w_-]+)", searchHelper)
                tweetsData = tweetsData.append({"dateSearched": dateSearched, "date": created_at, "fullText":
searchHelper, "tweetID": search_tweetID, "userID": searchFound["user"]["id_str"], "userScreenName":

```

```

searchFound["user"]["screen_name"], "RT": True, "RTCount": searchFound["retweet_count"], "searchQuery":
query, "hashtags": hashtags}, ignore_index=True, sort=False)
    else:
        print("error..")
    return tweetsData

```

python/cleanTweets.py

Εδώ “καθαρίζονται” τα κείμενα από stopwords, hashtags, urls, αριθμούς κ.α. .

```

from stopwords import stopWords
import re
#metatroph kai sort apo set se list
def convert(set):
    return sorted(set)

#Δημιουργία regex μοτίβου για καθαρισμό κειμένων από stopwords
def rmStopwords():
    #εισαγωγή του αρχείου με τα stopwords
    #https://github.com/explosion/spaCy/blob/master/spacy/lang/el/stop_words.py
    sw = set(stopWords)
    sw = convert(sw)
    #μετατροπή σε regex όλων των stopwords, τέτοιο ώστε να αφαιρεί από άλλα κείμενα όλα τα παραπάνω
    stopwords
    pat = r'\b(?:{})\b'.format('|'.join(sw))
    pat = pat.replace("ά", 'α')
    pat = pat.replace("έ", 'ε')
    pat = pat.replace("ή", 'η')
    pat = pat.replace("ι", 'ι')
    pat = pat.replace("ό", 'ο')
    pat = pat.replace("ύ", 'υ')
    pat = pat.replace("ώ", 'ω')

    return pat

#καθαρισμός κειμένου: stopword, new line, tabs, links, punctuation..
def cleanText(tweetsData):

```

```

#remove links
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[*\\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+", ' ', regex=True)
#remove hashtags
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("#[\w_-]+", ' ', regex=True)
#remove mentions
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("@[\w_-]+", ' ', regex=True)
#remove html encoded characters amp (&)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("&[\w_-]+", ' ', regex=True)
#remove special characters
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("[^\w\s]", ' ', regex=True)
#remove tabs, new lines chars, space
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("\t\n\r", ' ', regex=True)
#transform words to be without intonation
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("á", 'a', regex=True)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("é", 'e', regex=True)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("ή", 'i', regex=True)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("ί", 'i', regex=True)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("ϊ", 'i', regex=True)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("ĩ", 'i', regex=True)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("ó", 'o', regex=True)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("ú", 'u', regex=True)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("ώ", 'o', regex=True)
#remove stopwords
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace(rmStopwords(), ' ', regex=True)
#remove numbers
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("[\d-]", ' ', regex=True)
#tweetsData["fullText"] = tweetsData["fullText"].apply(lambda fullText: re.sub(r'\b\w{1,2}\b', '',
fullText))

return tweetsData

```

#Καθαρισμός κειμένου, δίχως να αφαιρεί stopwords

```
def cleanTextNoSwords(tweetsData):
```

```

    #remove links
    tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[*\\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+", ' ', regex=True)
    #remove hashtags
    tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("#[\w_-]+", ' ', regex=True)
    #remove mentions
    tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("@[\w_-]+", ' ', regex=True)
    #remove html encoded characters amp (&)
    tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("&[\w_-]+", ' ', regex=True)
    #remove special characters

```

```

tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("[^\w\s]", ' ', regex=True)
#remove tabs, new lines chars
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("\t\n\r", ' ', regex=True)
#transform words to be without intonation
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("á", 'α', regex=True)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("é", 'ε', regex=True)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("ή", 'η', regex=True)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("ί", 'ι', regex=True)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("ϊ", 'ι', regex=True)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("ĩ", 'ι', regex=True)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("ó", 'ο', regex=True)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("ú", 'υ', regex=True)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("ü", 'υ', regex=True)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("Û", 'υ', regex=True)
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("ŵ", 'ω', regex=True)

#remove numbers
tweetsData["fullText"] = tweetsData['fullText'].str.lower().str.replace("[\d-]", ' ', regex=True)
return tweetsData

```

python/scrapperWebsite/getArticles.py

Εδώ πραγματοποιείται η εξαγωγή άρθρων και των μεταδεδομένων τους από όλα τα ειδησεογραφικά

```

from getArticles_ingr import getArticles_ingr
from getArticles_kathimerinigr import getArticles_kathimerinigr
from getArticles_protothemagr import getArticles_protothemagr
from getArticles_skaigr import getArticles_skaigr
from getArticles_enikosgr import getArticles_enikosgr

import sys

sys.path.append('../database/helpers')

from getTimeNow import getDate
import datetime
import pandas as pd
pd.set_option('display.max_rows', 500)

```

```

def getAllArticles(fromDate):

    allArticles = pd.DataFrame()

    ingr = getArticles_ingr(fromDate, 50)
    kathim = getArticles_kathimerinigr(fromDate, 1)
    proto = getArticles_protothemagr(fromDate, 50)
    skai = getArticles_skaigr(fromDate, 50)
    enikos = getArticles_enikosgr(fromDate, 50)

    allArticles = allArticles.append([ingr,kathim,proto,skai,enikos])

    return allArticles

```

python/scraperWebsite/getArticles_skaigr.py

Αυτός είναι εξατομικευμένος κώδικας εξαγωγής των στοιχείων της σελίδας skai.gr, ώστε να εξαχθούν τα άρθρα του μαζί με τα μεταδεδομένα τους. Κάθε ένα ειδησεογραφικό έχει εξατομικευμένο κώδικα, διότι η HTML δομή τους διαφέρει.

```

from bs4 import BeautifulSoup
import requests
import pandas as pd
from checkTimeFormat import checkTimeFormat
import datetime
import sys
import time

import traceback
import logging

sys.path.append('../database/helpers')
from getTimeNow import getDate

def getHttpReq(url, retries):

```

```

r = requests.get(url)
return r

#Κατέβασμα όλων των άρθρων και των μεταδεδομένων από μία σελίδα του skai.gr
def getArticleInPage_skaigr(page):

    #Κατεβαίνει μια σελίδα του ειδησεογραφικού
    try:
        source = requests.get(' https://www.skai.gr/newsfeed?page='+str(page)).text
    except:
        return pd.DataFrame(), getDate()

    #Εισαγωγή της σελίδας του ειδησεογραφικού στην BeautifulSoup βιβλιοθήκη με lxml parser
    soup = BeautifulSoup(source, 'lxml')

    df = pd.DataFrame(columns = ['title', 'summary', 'date', 'category', 'link', 'article', 'source'])

    #Αναζήτηση και αποθήκευση σε μεταβλητές όλων των επιμέρων στοιχείων του ειδησεογραφικού
    for article in soup.find_all('div', class_= 'cmnArticleTitlePad'):

        category = ''
        try:
            title = article.find('a', class_='title mainLink').text.strip()
            link = 'https://www.skai.gr' + article.find('a', class_='title mainLink')['href']
            summary = article.find('div', class_='lead').text.strip()

            date = article.find('div', class_='date').text.strip()
            date = datetime.datetime.strptime(date, '%d/%m/%Y - %H:%M')
            date = date.strftime('%Y-%m-%d %H:%M:%S')
            date = datetime.datetime.strptime(date, '%Y-%m-%d %H:%M:%S')

            #Κατεβαίνει η σελίδα που περιέχει το άρθρο
            articleLink = requests.get(link).text.strip()
            soupArticleMain = BeautifulSoup(articleLink, 'lxml')

        except:
            print('error parsing, title or link or summary or date')
            continue

        try:
            articleText = soupArticleMain.find('div', id='articleHidder').text.strip();
        except Exception as e:
            #print('cat error: ',category)

```

```
print('title error: ', title)
logging.error(traceback.format_exc())
print('#####')
continue

try:
    category = soupArticleMain.find('div', class_='article-category').find('a').text.strip()
except:
    print('Didnt find category')

data = {'title': title, 'summary': summary, 'date': date, 'category': category, 'link': link,
'article': articleText, 'source': 'skai.gr'}
df = df.append(data, ignore_index = True )

minDate = df['date'].min()

return df, minDate
```

#Κατέβασμα όλων των άρθρων και των μεταδεδομένων από εισαγόμενη ημερομηνία και μετά του skai.gr

```
def getArticles_skaigr(fromDate, limitPage):

    df = pd.DataFrame(columns = ['title', 'summary', 'date', 'category', 'link', 'article', 'source'])
    print( '%%%/%%%/%%%/%%%/%%%/%%%/%%%/%%%/%%%/%%%--skai.gr--%%/%%%/%%%/%%%/%%%/%%%/%%%/%%%/%%%/' )
    for i in range(0, limitPage):
        print('at page: ' + str(i+1) + '/' + str(limitPage) )
        singlePagedf, minDateSinglePage = getArticleInPage_skaigr(str(i))

        df = df.append(singlePagedf, ignore_index = True )

    try:
        if(minDateSinglePage <= fromDate ):
            print('stopped searching at page: ' + str(i+1))
            break
    except:
        print('catch date type: ', type(minDateSinglePage))
        print('catch date: ', minDateSinglePage)

    filtDateRange = df['date'] >= fromDate

    df = df.loc[filtDateRange]

    print('Number of articles found: ', df.shape[0])

    return df
```

python/bowCosSim.py

Εδώ τα κείμενα λημματοποιούνται, μετατρέπονται με BoW και κανονικοποίηση l2norm και συγκρίνονται μεταξύ τους με την χρήση συνημιτονοειδής απόστασης ώστε να υπολογιστούν οι συσχετίσεις άρθρων-τάσεων.

```
import math

import sys
sys.path.append('getTweets')
from cleanTweets import cleanText

sys.path.append('greek_stemmer')
from greek_stemmer import stemmer

sys.path.append('database/helpers')
from getTimeNow import getDate

#Επιστρέφει το lemmatization κειμένου
def lemmatizer2(text):

    lemText = ''
    for token in text.split():
        lemText += ' ' + stemmer.stem_word(token, 'VBG')
    return lemText

#Επιστρέφει την διανυσματικοποίηση κειμένου δοθείσας το κείμενο και το λεξικό του.
def bagOfWords(text, voc):
    wob = dict.fromkeys(voc,0)
    text = text.split()

    #δημιουργία bagOfWords
    for j in text:
        wob[j] += 1

    l2normSum = 0

    #εφαρμογή l2norm στο διάνυσμα
    for i in list(wob.values()):
        l2normSum += i*i
    l2normSum = math.sqrt( l2normSum )

    if(l2normSum != 0):
        for i in wob.keys():
            wob[i] = wob[i]/l2normSum
```

```

    return list(wob.values() )

#Επιστρέφει την συνμητιονοειδής απόσταση μεταξύ δύο διανυσμάτων
def cosineSimil(vec1, vec2):
    distance = 0
    for i in range(len(vec1)):
        distance += vec1[i]*vec2[i]
    return distance

#Συγκρίνει άρθρα με tweets και επιστρέφει τα αποτελέσματα
def similarity(articles, tweets, searchQuery):
    #Διαγράφει τα τυχόν διπλότυπα άρθρα
    articles.drop_duplicates(subset=["article", 'link'], inplace=True)
    #Διαγράφει κενά άρθρα
    articles = articles[~articles['article'].isna() ]
    #Κρατάει αντίγραφο του πραγματικού άρθρου
    articles['articleOriginal'] = articles['article']
    #Καθαρισμός άρθρου
    articles = cleanText(articles.rename(columns={'article': 'fullText'}))
    #Πραγματοποιεί lemmatization στο κείμενο του άρθρου
    articles['fullText'] = articles['fullText'].apply(lambda text: lemmatizer2(text))

    #Κρατάει αντίγραφο του πραγματικού tweet
    tweets['tweetOriginal'] = tweets['fullText']
    #Καθαρισμός tweet
    tweets = cleanText(tweets)
    #Διαγράφει τα τυχόν διπλότυπα tweets
    tweets.drop_duplicates(subset=["fullText"], inplace=True)
    #Διαγράφει κενά tweets
    tweets = tweets[~tweets['fullText'].str.split().str.len().lt(1) ]
    #Πραγματοποιεί lemmatization στο tweet
    tweets['fullText'] = tweets['fullText'].apply(lambda text: lemmatizer2(text))

    #Δημιουργεί το λεξικό των άρθρων
    articlesVocub = set()
    articles['fullText'].str.split().apply(articlesVocub.update)

    #Δημιουργεί το λεξικό των tweets
    tweetsVocub = set()
    tweets['fullText'].str.split().apply(tweetsVocub.update)

    #Ενώνει το λεξικό των tweets και άρθρων
    vocubConcatenated = articlesVocub.union(tweetsVocub)

    #Δημιουργείται στήλη με την διανυσματικοποίηση άρθρου και tweet αντίστοιχα

```

```

articles['bow'] = articles['fullText'].apply(lambda text: bagOfWords(text, vocabConcatenated))
tweets['bow'] = tweets['fullText'].apply(lambda text: bagOfWords(text, vocabConcatenated))
import pandas as pd
pd.options.mode.chained_assignment = None
pd.set_option('display.max_rows', 200)
pd.set_option('display.max_colwidth', None)

similResults = pd.DataFrame()

#Αθροίζονται όλες οι συνημιτονοειδής αποστάσεις και διαιρούνται με το πλήθος των κειμένων για να
πάρουμε τον μέσο όρο.
for i in range(articles.shape[0]):
    similarity = 0
    for j in range(tweets.shape[0]):
        tweet = tweets['bow'].iloc[j]
        article = articles['bow'].iloc[i]

        similarity += cosineSimil(tweet, article)

similarity = (similarity/(tweets.shape[0])) * 1000
#print(str(similarity) + ' ' + articles['summaryOriginal'].iloc[i])

#Όλες οι πληροφορίες των αποτελεσμάτων αποθηκεύονται σε dataframe
similResults = similResults.append({'cosSimil': similarity, 'articleOriginal':
articles['articleOriginal'].iloc[i], 'articleTitle': articles['title'].iloc[i], 'articleDate':
articles['date'].iloc[i], 'articleCategory': articles['category'].iloc[i], 'articleLink':
articles['link'].iloc[i], 'articleSource': articles['source'].iloc[i], 'searchQuery': searchQuery}
,ignore_index=True)

#clear_output(wait=True)
#print(similResults[similResults['cosSimil'].max() == similResults['cosSimil']])
#print('Done ' + str(i+1) + '/' + str(articles.shape[0]))
print('calculated ', articles.shape[0], ' articles X ', tweets.shape[0], ' tweets')
return similResults

```

nodeServer/server.js

Ο κύριος κορμός του NodeJS server και όλες οι λειτουργίες του REST API του εγχειρήματος.

```
const express = require('express');
const dotenv = require('dotenv');
const cors = require('cors');
const dayjs = require('dayjs');

const errorHandler = require('./middleware/error');
const connectDB = require('./config/db');

//Load env vars
dotenv.config({ path: './config/config.env' });

const ResultsSchema = require('./models/Results');
const TweetsSchema = require('./models/Tweets');
const searchQuerySchema = require('./models/SearchQuery');
const TopSimilTrendsSchema = require('./models/TopSimilTrends');
const countStatsSchema = require('./models/countStats');
const sourcesScoreSchema = require('./models/cosSimilScores');

//Σύνδεση στο database
connectDB();

const app = express();

app.use(cors());

app.use(express.json());
app.use(express.urlencoded({ extended: true }));

app.use(errorHandler);

app.use(express.static('dist/twitter-articles'));

const PORT = process.env.PORT;

const server = app.listen(
  PORT,
  console.log(`Server running in ${process.env.NODE_ENV} mode on port ${PORT}`)
);

let results = ResultsSchema;
```

```

let topTweets = TweetsSchema;

let searchQuery = searchQuerySchema;

let trends;

let topSimilTrends = TopSimilTrendsSchema;

let countStats = countStatsSchema;

let sourcesScore = sourcesScoreSchema;

app.get('/', (req, res) => {
  res.sendFile('index.html', { root: __dirname });
});

let bestResultsDate = '';

//Επιστρέφει τα αποτελέσματα που έχουν επιλεγεί για μία ημερομηνία
//Μόνο τα fields που αφορούν τα trends
//Αν δεν δοθεί ημερομηνία τότε επιστρέφονται τα αποτελέσματα
//της τελευταίας ημερομηνίας
app.get('/trends', function (req, res, next) {
  if (req.query.data == '') {
    results
      .findOne({})
      .sort({ bestResultsDate: -1 })
      .limit(1)
      .then((maxDate) => {
        results.find(
          //resultCompletedDate: resultCompletedDate
          { bestResultsDate: maxDate.bestResultsDate },
          '-_id searchQuery rank volume freshness_trend dateDiscovered_trend bestResultsDate',
          {
            skip: 0, // Starting Row
            limit: 25, // Ending Row
            sort: {
              rank: 1 //Sort by Rank ASC
            }
          },
          function (err, result) {
            trends = [...new Set(result.map((item) => item.searchQuery))];
            bestResultsDate = result[0].bestResultsDate;
            res.send(JSON.stringify(result));
          }
        );
      });
  }
});

```

```

    }
  );
});
} else if (req.query.data) {
  bestResultsDate = req.query.data;

  results.find(
    { bestResultsDate: bestResultsDate },
    '-_id searchQuery rank volume freshness_trend dateDiscovered_trend bestResultsDate',
    {
      skip: 0, // Starting Row
      limit: 25, // Ending Row
      sort: {
        rank: 1 //Sort by Rank ASC
      }
    },
    function (err, result) {
      trends = [...new Set(result.map((item) => item.searchQuery))];

      res.send(JSON.stringify(result));
    }
  );
}
});

//Επιστρέφει τα αποτελέσματα που έχουν επιλεχτεί για μία ημερομηνία
//Μόνο τα fields που αφορούν τα articles
app.get('/articles', function (req, res, next) {
  results.find(
    { bestResultsDate: bestResultsDate },
    '-_id searchQuery articleTitle articleLink articleDate articleCategory articleSource cosSimil diffDate',
    bestResultsDate',
    {
      skip: 0, // Starting Row
      limit: 25 // Ending Row
    },
    function (err, result) {
      res.send(JSON.stringify(result));
    }
  );
});

//Επιστρέφει τα αποτελέσματα που έχουν επιλεχτεί για μία ημερομηνία
//Μόνο τα fields που αφορούν τα tweets
app.get('/tweets', function (req, res, next) {

```

```

topTweets.find(
  { bestResultsDate: bestResultsDate },
  '-_id RTCount date dateSearched fullText hashtags searchQuery tweetID userID userScreenName
bestResultsDate',
  {
    limit: 5000,
    sort: {
      RTCount: -1 //Sort by RTCount DESC
    }
  },
  function (err, result) {
    res.send(JSON.stringify(result));
  }
);
});

```

//Ψάχνει όλες τις τάσεις της τελευταίας εβδομάδας ή
 //την πορεία μίας συγκεκριμένης τάσης σε σχέση με τον χρόνο

```

app.get('/searchquery', function (req, res, next) {
  const hashtag = req.query.data;

  if (hashtag) {
    searchQuery.find(
      { hashtag: hashtag },
      '-_id dateDownloaded hashtag rank',
      {
        limit: 5000,
        sort: {
          dateDownloaded: 1 //Sort by dateDownloaded ASC
        }
      },
      function (err, result) {
        res.send(JSON.stringify(result));
      }
    );
  } else {
    searchQuery
      .find({
        dateDownloaded: {
          $gte: dayjs().subtract(7, 'day').format('YYYY-MM-DD')
        }
      })
      .distinct('hashtag', function (err, result) {
        res.send(JSON.stringify(result));
      });
  }
});

```

```

    }
  });

  //Ψάχνει στην ΒΔ τάσεις που βρέθηκαν με μεγάλη σχετικότητα με κάποιο άρθρο
  app.get('/topsimiltrends', function (req, res, next) {
    topSimilTrends.find({}, function (err, result) {
      res.send(JSON.stringify(result));
    });
  });

String.prototype.allReplace = function (obj) {
  var retStr = this;
  for (var x in obj) {
    retStr = retStr.replace(new RegExp(x, 'g'), obj[x]);
  }
  return retStr;
};

function makeRegexSearch(searchKeyword) {
  //^(?=.*λΞξηΑ)(?=.*λΞξηΒ)(?=.*λΞξηΓ)...
  queryRegex = '^(*.*)';

  searchKeyword = searchKeyword.map((keyword) => {
    return keyword.allReplace({
      α: '[αά]',
      ε: '[εέ]',
      η: '[ηή]',
      ι: '[ιί]',
      ο: '[οό]',
      υ: '[υύ]',
      ω: '[ωώ]'
    });
  });

  for (let index in searchKeyword) {
    if (parseInt(index) + 1 === searchKeyword.length) {
      queryRegex += searchKeyword[index] + ')...';
    } else {
      queryRegex += searchKeyword[index] + ')(*.*';
    }
  }
  return queryRegex;
}

const searchKeywords = ['ελλάδα'];

```

```

// Ψάχνει στην βάση δεδομένων αποτελέσματα με κλειδιά στα πεδία τίτλος του άρθρου ή τίτλο τάσης
// και επιστρέφει τα δεδομένα στον client
app.post('/searcharticlesbykeywords', function (req, res, next) {
  const searchKeywords = req.body.searchKeywords.split(' ');
  const hashtagKeyword = req.body.searchKeywords;
  const searchFieldToggle = req.body.searchType; //(articleTitle, searchQuery)
  const sortBy = JSON.parse(req.body.sortObject);
  console.log(searchKeywords);
  console.log(sortBy);
  if (searchFieldToggle === 'articleTitle' && searchKeywords.length > 0) {
    results
      .aggregate([
        {
          $match: {
            articleTitle: {
              $regex: makeRegexSearch(searchKeywords),
              $options: 'i'
            }
          }
        },
        {
          $sort: { cosSimil: -1 }
        },
        {
          $group: {
            _id: '$articleLink',
            doc: { $first: '$$ROOT' }
          }
        },
        {
          $replaceRoot: {
            newRoot: '$doc'
          }
        }
      ])
      .sort(sortBy)
      .exec(function (err, result) {
        if (result) {
          res.send(JSON.stringify(result.slice(0, 10)));
        } else {
          res.send('none_found');
        }
      })

    //console.log(result);
  }
});

```

```

    });
} else if (searchFieldToggle === 'searchQuery' && hashtagKeyword.length > 0) {
  results
    .aggregate([
      {
        $match: {
          searchQuery: {
            $regex: hashtagKeyword,
            $options: 'i'
          }
        }
      },
      {
        $sort: { cosSimil: -1 }
      },
      {
        $group: {
          _id: '$searchQuery',
          doc: { $first: '$$ROOT' }
        }
      },
      {
        $replaceRoot: {
          newRoot: '$doc'
        }
      }
    ])
    .sort(sortBy)
    .exec(function (err, result) {
      if (result) {
        res.send(JSON.stringify(result.slice(0, 10)));
      } else {
        res.send('none_found');
      }
    });
}
});

```

```

//Επιστρέφει στον client την max και min ημερομηνίες των αποτελεσμάτων
//αυτό με σκοπό για να δωθεί το ημερομηνιακό φάσμα διαθέσιμων αποτελεσμάτων
app.get('/daterangeresults', (req, res) => {
  results.aggregate(
    [
      {
        $group: {

```

```

        _id: null,
        max_date: { $max: '$bestResultsDate' },
        min_date: { $min: '$bestResultsDate' }
    }
},
{
    $project: {
        _id: 0
    }
}
],
function (err, result) {
    res.send(JSON.stringify(result));
}
);
});

//Επιστρέφει τις χρονοσημάνσεις διαθέσιμων αποτελεσμάτων για μία συγκεκριμένη ημέρα
app.post('/resultdates', (req, res) => {
    const date = dayjs(new Date(req.query.data));

    results
        .find({
            bestResultsDate: {
                $gte: date,
                $lt: date.add(1, 'day')
            }
        })
        .distinct('bestResultsDate', function (err, timestamps) {
            res.send(JSON.stringify(timestamps));
        });
});

//πλήθος tweets, πλήθος articles...
app.get('/countstats', (req, res) => {
    countStats
        .find({})
        .select({ _id: 0 })
        .exec(function (err, counts) {
            res.send(JSON.stringify(counts));
        });
});

//Επιστρέφει την βαθμολογία των ειδησεογραφικών για το πόσο ακολουθούν την πλατφόρμα Twitter
app.get('/sourcesscore', (req, res) => {

```

```

sourcesScore
  .find({})
  .select({ _id: 0 })
  .exec(function (err, scores) {
    res.send(JSON.stringify(scores));
  });
});

//Βρίσκει το καλύτερο αποτέλεσμα συσχέτισης μίας τάσης με κάποιο άρθρο και το επιστρέφει στον client
app.get('/gettrendscorrespondingarticle', (req, res) => {
  const trend = req.query.data;
  results
    .findOne({
      searchQuery: trend,
      dateDownloaded: {
        $gte: dayjs().subtract(10, 'day').format('YYYY-MM-DD')
      }
    })
    .sort('-cosSimil') //find the max cosSimil of that trend
    .exec(function (err, scores) {
      res.send(JSON.stringify(scores));
    });
});

//Επιστρέφει τις επαναλαμβανόμενες τάσεις των τελευταίων τριών ημερών
// ή επιστρέφει τις MH επαναλαμβανόμενες τάσεις των τελευταίων τριών ημερών
app.get('/freshnesstrend', (req, res) => {
  const freshness = req.query.data;

  let matchOption;

  if (freshness == 'old') {
    matchOption = {
      freshness_trend: freshness,
      bestResultsDate: {
        $gte: dayjs().subtract(3, 'day').toDate()
      }
    };
  } else if (freshness == 'new') {
    matchOption = {
      freshness_trend: freshness,
      bestResultsDate: {
        $gte: dayjs().subtract(3, 'day').toDate()
      }
    };
  }
});

```

```

    } else {
      res.send('bad query, freshness value not (new | old)');
      return;
    }
    console.log(matchOption);

```

```

results
  .aggregate(
    [
      {
        $match: matchOption
      },
      { $project: { _id: 0, articleOriginal: 0 } },
      {
        $sort: { cosSimil: -1 }
      },
      {
        $group: {
          _id: '$searchQuery',
          doc: { $first: '$$ROOT' }
        }
      },
      {
        $replaceRoot: {
          newRoot: '$doc'
        }
      }
    ],
    { allowDiskUse: true }
  )
  .sort({ dateDiscovered_trend: -1 })
  .exec(function (err, result) {
    if (err) {
      console.log(err);
    }

    if (result) {
      res.send(JSON.stringify(result));
    } else {
      res.send('none_found');
    }
  });
});

```

```

//Handle unhandled promise rejections

```

```
process.on('unhandledRejection', (err, promise) => {
  console.log(`Error: ${err.message}`);
  //Close server & exit process
  server.close(() => process.exit(1));
});
```

frontend/twitter-articles/src/app/date-chart/ date-chart.component.ts

Εδώ δημιουργείτε το χρονοδιάγραμμα των τάσεων.

```
import { HttpClient } from '@angular/common/http';
import {
  Component,
  OnInit,
  Inject,
  Input,
  OnChanges,
  SimpleChanges,
} from '@angular/core';
import { Chart, registerables, ChartType } from 'chart.js';
import { Observable } from 'rxjs';
import { SearchQuery } from '../interfaces';
import { GetDataService } from '../services/get-data.service';
Chart.register(...registerables);
import * as dayjs from 'dayjs';
import { ChangeArticlesService } from '../services/change-articles.service';
import { environment } from 'src/environments/environment';

@Component({
  selector: 'app-date-chart',
  templateUrl: './date-chart.component.html',
  styleUrls: ['./date-chart.component.scss'],
})
export class DateChartComponent implements OnInit {
  constructor(
    private http: HttpClient,
    private changeArticle: ChangeArticlesService
  ) {}
```

```

public queryData;

public searchQuery: string;

public myChart: any;

public availableHashtags: string[];

public hashtagSelected;

public singleBestResult;

onClick(searchQuery) {
  this.searchQuery = searchQuery;
  //edw na stelnete kai search query gia tash na emfanizontai kai ekei ta apotelesmata.
  this.makeChart();

  this.getTrendsCorrespondingArticle(searchQuery);
}

ngOnInit(): void {
  this.changeArticle.data$.subscribe((data) => {
    this.hashtagSelected = data;
    this.onClick(this.hashtagSelected);
  });

  this.searchQuery = 'Επιλέξτε κάποιο hashtag';
  this.makeChart();

  //pare ola ta diathesima hashtags
  this.getAvailableSearchQueryys();
}

async makeData() {
  let dates = [];
  let ranks = [];

  //apo to megalitero sto mikrotero
  let dateRange = this.giveDateRange();
  let rankInRange = [];

  await this.getSearchQuery(this.searchQuery);

  for (let trend of this.queryData) {

```

```

    let date = dayjs(trend.dateDownloaded).subtract(2, 'hour');

    dates.push(date);

    //μεσα σε μια bdomada tha prepei na yparxoun ranks (1,2,3,4,5) kai 0 pou na dilwnei thn eksafanish
    tou apo ta trends
    ranks.push(trend.rank);
  }

  for (
    let dateRanIndex = 0;
    dateRanIndex < dateRange.length;
    dateRanIndex += 1
  ) {
    for (let dateIndex in dates) {
      if (
        dates[dateIndex].isBefore(dateRange[dateRanIndex]) &&
        dates[dateIndex].isAfter(dateRange[dateRanIndex + 1])
      ) {
        rankInRange.push(ranks[dateIndex]);
        break;
      }
    }
    if (dateRanIndex == rankInRange.length) {
      rankInRange.push('notFound');
    }
  }

  return [dateRange, rankInRange];
}

async makeChart() {
  let makeData = await this.makeData();
  let dateRange = makeData[0];
  let rankInRange = makeData[1];

  var ctx = document.getElementById('myChart') as HTMLCanvasElement;

  const labels = dateRange
    .map((x) => {
      if (x.day() === 0) {
        return x.format('DD-MM') + ' Κυριακή ' + x.format('HH:mm');
      } else if (x.day() === 1) {
        return x.format('DD-MM') + ' Δευτέρα ' + x.format('HH:mm');
      } else if (x.day() === 2) {

```

```

        return x.format('DD-MM') + ' Τρίτη ' + x.format('HH:mm');
    } else if (x.day() === 3) {
        return x.format('DD-MM') + ' Τετάρτη ' + x.format('HH:mm');
    } else if (x.day() === 4) {
        return x.format('DD-MM') + ' Πέμπτη ' + x.format('HH:mm');
    } else if (x.day() === 5) {
        return x.format('DD-MM') + ' Παρασκευή ' + x.format('HH:mm');
    } else {
        return x.format('DD-MM') + ' Σάββατο ' + x.format('HH:mm');
    }
})
.reverse();
const data = {
  labels: labels,
  datasets: [
    {
      label: this.searchQuery,
      backgroundColor: 'rgb(255, 99, 132)',
      borderColor: 'rgb(255, 99, 132)',
      //data= request.get(rank)
      data: rankInRange.reverse(),
      pointRadius: 4,
      pointHoverRadius: 8,
    },
  ],
};
const config = {
  responsive: true,
  maintainAspectRatio: false,
  type: 'line' as ChartType,
  data: data,
  options: {
    plugins: {
      title: {
        display: true,
        text: 'Εμφάνιση της τάσης την τελευταία βδομάδα',
      },
      subtitle: {
        display: true,
        text: 'Κάθε κουκίδα αντιπροσωπεύει την κατάταξη του trend, την στιγμή που βρέθηκε',
      },
    },
    pointStyle: 'circle',
    //pointBackgroundColor: 'white',
  },
  scales: {

```

```

        x: {
            display: true,
            title: {
                display: true,
                text: 'Άξονας Χ: Ημερομηνία από τώρα (τελευταία δεξιά) μέχρι πίσω μια βδομάδα (προς τα
αριστερά)',
            },
        },
        y: {
            display: true,
            title: {
                display: true,
                text: 'Άξονας Υ: Κατάταξη',
            },
            ticks: {
                maxTicksLimit: 5,
                stepSize: 1,
            },
            afterDataLimits: (scale) => {
                scale.max = 5;
                scale.min = 1;
            },
            reverse: true,
        },
    },
    },
};
if (this.myChart) {
    this.myChart.data.datasets = data.datasets;
    this.myChart.update();
} else {
    this.myChart = new Chart(ctx, config);
}
}

giveDateRange() {
    let datenow: any = dayjs();
    datenow = datenow
        .subtract(datenow.minute(), 'minute')
        .subtract(datenow.millisecond(), 'millisecond')
        .subtract(datenow.second(), 'second')
        .add(3, 'hour');
    let createDateRange = [];
    //ana 3 wres dimiourgei date
    for (let i = 0; i < 7 * 24; i += 3) {

```

```

        createDateRange.push(datenow.subtract(i, 'hours'));
    }
    return createDateRange;
}

async getSearchQuery(searchQuery: string) {
    const pathSearchQuery = 'http://' + environment.HOST + ':5000/searchquery';
    this.queryData = await this.http
        .get(pathSearchQuery + '?data=' + encodeURIComponent(searchQuery))
        .toPromise();
}

async getAvailableSearchQueryys() {
    const pathSearchQuery = 'http://' + environment.HOST + ':5000/searchquery';
    let results: string[] = await this.http
        .get<string[]>(pathSearchQuery)
        .toPromise();
    this.availableHashtags = results;
}

async getTrendsCorrespondingArticle(trend) {
    const pathSearchQuery =
        'http://' + environment.HOST + ':5000/gettrendsCorrespondingArticle';
    this.singleBestResult = await this.http
        .get(pathSearchQuery + '?data=' + encodeURIComponent(trend))
        .toPromise();
}
}

```

frontend/twitter-articles/src/app/services/ get-data.service.ts

Εδώ ορίζονται όλα τα GET αιτήματα από τον client στον server ώστε να ληφθούν τα δεδομένα για τους πίνακες των τάσεων, tweets, άρθρων

```

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

```

```

import { SearchQuery } from '../interfaces';
import { environment } from 'src/environments/environment';

@Injectable({
  providedIn: 'root',
})
export class GetDataService {
  pathTrends: string = 'http://' + environment.HOST + ':5000/trends';
  pathArticles: string = 'http://' + environment.HOST + ':5000/articles';
  pathTweets: string = 'http://' + environment.HOST + ':5000/tweets';
  pathTopSimilTrends: string =
    'http://' + environment.HOST + ':5000/topsimiltrends';

  pathFresh_OldTrends: string =
    'http://' + environment.HOST + ':5000/freshnesstrend';

  constructor(private http: HttpClient) {}

  // get data
  getTrendsApi(dateBack: string): Observable<any> {
    console.log('dateBack', dateBack);

    return this.http.get(this.pathTrends + '?data=' + dateBack);
  }

  getArticlesApi(dateBack: string): Observable<any> {
    return this.http.get(this.pathArticles + '?data=' + dateBack);
  }

  getTweetsApi(dateBack: string): Observable<any> {
    return this.http.get(this.pathTweets + '?data=' + dateBack);
  }

  getTopSimilTrends(): Observable<any> {
    return this.http.get(this.pathTopSimilTrends);
  }

  getFreshTrends(): Observable<any> {
    return this.http.get(this.pathFresh_OldTrends + '?data=new');
  }

  getRepeatedTrends(): Observable<any> {
    return this.http.get(this.pathFresh_OldTrends + '?data=old');
  }
}

```

frontend/twitter-articles/src/app/pipes/ similarity.pipe.ts

Ένα από τα pipes που χρησιμοποιήθηκαν για την μετατροπή αριθμητικών σε Strings. Φαίνεται ότι ανάλογα το νούμερο (συννημιτονοειδής απόσταση) μετατρέπεται στο ανάλογο μήνυμα.

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'similarity',
})
export class SimilarityPipe implements PipeTransform {
  transform(value: number): string {
    if (isNaN(value)) {
      return 'error';
    } else if (value < 19) {
      return 'MH-ΣΧΕΤΙΚΟ';
    } else if (value < 35) {
      return 'ΛΙΓΟ-ΣΧΕΤΙΚΟ';
    } else if (value < 100) {
      return 'ΣΧΕΤΙΚΟ';
    } else if (value < 150) {
      return 'ΠΟΛΥ-ΣΧΕΤΙΚΟ';
    } else {
      return 'ΕΞΑΙΡΕΤΙΚΑ-ΣΧΕΤΙΚΟ';
    }
  }
}
```