ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεσιακή Ανάλυση Δεδομένων με χρήση Γράφων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Παντελεήμων Κρασαδάκης

Επιβλέπων : **Ιωάννης Βασιλείου**
Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2017

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεσιακή Ανάλυση Δεδομένων με χρήση Γράφων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Παντελεήμων Κρασαδάκης

Επιβλέπων : **Ιωάννης Βασιλείου**
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 15η Φεβρουαρίου 2017

.....	....	.....
Ιωάννης Βασιλείου	Ανδρέας-Γεώργιος Σταφυλοπάτης	Γιώργος Στάμου
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2017

Παντελεήμων Κρασαδάκης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2017

Περίληψη

Η έλευση του Διαδικτύου δημιούργησε νέα μέσα επικοινωνίας, ενημέρωσης και ανταλλαγής απόψεων. Η ανάγκη εξαγωγής χρήσιμων συμπερασμάτων αναλύοντας με αυτοματοποιημένο τρόπο τον τεράστιο όγκο πληροφοριών οδήγησε στην εμφάνιση πολλών μεθόδων εξαγωγής δεδομένων. Ειδικότερα, τα δεδομένα των Κοινωνικών Μέσων Δικτύωσης λόγω των εγγενών χαρακτηριστικών τους δημιουργούν σημαντικές δυσκολίες με αποτέλεσμα να αναζητούνται νέες μέθοδοι επεξεργασίας.

Σε αυτό το πλαίσιο, η παρούσα εργασία μελετά την εφαρμογή ενός μοντέλου το οποίο χρησιμοποιεί γράφους, με σκοπό τον εντοπισμό συσχετίσεων τόσο σε ένα δημοσιογραφικό περιβάλλον, όσο και σε ένα κοινωνικό δίκτυο, το Twitter. Διερευνάται ο τρόπος με τον οποίο ανταποκρίνεται το σύστημα μας στην προσθήκη νέων συσχετίσεων και στην εύρεση και εξαγωγή των ισχυρότερων δεσμών μεταξύ λέξεων κλειδιών από αυτές.

Λέξεις κλειδιά

Ανάλυση δεδομένων, κοινωνικά δίκτυα, Twitter, βάσεις δεδομένων γράφου, Neo4j, Cypher, πλατφόρμα Node.js

Abstract

The advent of the Internet created new channels of communication, information and opinion exchange. The need to extract useful insight through automated analysis of the vast amount of user-generated content gave rise to many data extraction methods. Social Media content, in particular, due to its inherent characteristics poses serious challenges that call for new processing techniques.

In this context, this thesis focuses on the application of a graph database model, in order to find relationships both in a journalist environment and in a Social Network like Twitter. It also explores how the system responds to the addition of new relationships and to the finding and extraction of powerful connections between keywords.

Key words

Data analysis, social networks, Twitter, geospatial data, graph databases, Neo4j, Cypher, Node.js platform

Περιεχόμενα

1. Εισαγωγή

1.1 Διατύπωση Προβλήματος

Την τελευταία δεκαπενταετία έχει παρατηρηθεί παγκοσμίως μια τεράστια αύξηση του αριθμού των χρηστών που χρησιμοποιούν το Διαδίκτυο.Στην αύξηση αυτή των χρηστών του Διαδικτύου σημαντικό ρόλο παίζουν και οι υπηρεσίες κοινωνικής δικτύωσης,όπως το Facebook και το Twitter. Οι ιστοσελίδες κοινωνικής δικτύωσης κερδίζουν συνεχώς περισσότερους χρήστες, καθώς τους παρέχουν την δυνατότητα να αλληλεπιδρούν και να επικοινωνούν μεταξύ τους. Γίνεται εύκολα αντιληπτό ότι αυτές οι ιστοσελίδες κατακλύζονται από πληθώρα δεδομένων. Είναι λοιπόν εμφανής η ανάγκη να μπορέσουμε να συλλέξουμε αυτά τα δεδομένα και να τα διαχειριστούμε κατάλληλα ώστε να εξάγουμε χρήσιμες πληροφορίες.Έτσι, το πρόβλημα που περιγράφεται σε αυτή την εργασία είναι η δημιουργία μιας αυτόματης διαδικασίας που προσπαθεί να αντιμετωπίσει τον συνεχώς αυξανόμενο όγκο πληροφοριών στο Διαδίκτυο,τόσο σε δημοσιογραφικό περιβάλλον όσο και από δεδομένα που προέρχονται από μέσα κοινωνικής δικτύωσης (συγκεκριμένα το Twitter). Θέλουμε να έχουμε τη δυνατότητα να παίρνουμε άρθρα από εφημερίδες, να τα επεξεργαζόμαστε κατάλληλα, με σκοπό να βρούμε συσχετισμό μεταξύ λέξεων από όλα τα άρθρα και από αυτό το συσχετισμό να μπορεί ο χρήστης να εξάγει κάποια συμπεράσματα, με βάση κάποια λέξη κλειδί που επιλέξει. Από τη στιγμή που ασχολούμαστε με δημοσιογραφικό περιβάλλον (RSS Feeds) περιμένουμε και τα αποτελέσματα που θα βγάλουμε να είναι αντίστοιχης φύσης, δηλαδή να μην παρατηρηθούν σχολιασμοί, αλλά γεγονότα. Επίσης, θέλουμε να βγάλουμε τα αποτελέσματα με στατιστικά εργαλεία και όχι με Επεξεργασία Φυσικής Γλώσσας (Natural Language Processing = NLP), ώστε να μην εξαρτόμαστε από τη γλώσσα και το περιεχόμενο των πηγών που χρησιμοποιούμε.

1.2 Προσέγγιση

Η προσέγγιση που ακολουθούμε για την αντιμετώπιση του παραπάνω προβλήματος συνοψίζεται στη συνέχεια. Αρχικά, μαζεύονται δεδομένα από το διαδίκτυο (RSS και Twitter), αποθηκεύονται στη βάση δεδομένων Firebase[19] για τον έλεγχο της μοναδικότητας τους και την αναδημιουργία του συστήματος σε περίπτωση λάθους. Στη συνέχεια, από τα άρθρα αυτά θα εξάγουμε λέξεις-κλειδιά και θα τις δομήσουμε σε μορφή γράφου, χρησιμοποιώντας τη graph database Neo4j[5]. Αυτή η διαδικασία ακολουθείται, για τη διευκόλυνση της στατιστικής μελέτης που επιθυμούμε να εφαρμόσουμε στα δεδομένα μας. Ο γράφος έχει ως κόμβους τις λέξεις-κλειδιά και οι ακμές του ενώνουν κόμβους που έχουν συναντηθεί στο ίδιο άρθρο. Οι ακμές έχουν σαν βασικό χαρακτηριστικό τους το βάρος τους και το οποίο υποδηλώνει τη δύναμη της σύνδεσης μεταξύ των κόμβων που συνδέει. Έτσι αυτό που θέλουμε να επιτύχουμε από αυτή τη δομή, είναι να έχουμε λέξεις που έχουν βρεθεί πολλές φορές μαζί στα ίδια άρθρα, να σχετίζονται με μεγάλα βάρη, ώστε να μπορούμε με queries να βρούμε αυτούς τους ισχυρούς δεσμούς. Οι συσχετίσεις αυτές παρουσιάζονται στους χρήστες με τη βοήθεια μιας διεπαφής, όπου έχουν τη δυνατότητα να επιλέξουν τη λέξη-κλειδί που επιθυμούν να αναζητήσουν είτε στη Βάση Δεδομένων είτε σε ζωντανή αναζήτηση στο Twitter και το βαρός των κόμβων που θα φαίνονται στο γράφο.

Τα βήματα που ακολουθούνται είναι τα εξής :

i) Δημιουργία RSS Scrapper και Twitter Streamer
ii) Χρήση Firebase για αποθήκευση και έλεγχο
iii) 'Κατέβασμα' από Firebase και εξόρυξη λέξεων-κλειδιών
iv) Δημιουργία δεδομένων σε μορφή για επεξεργασία από Neo4j
v) Προσθήκη σε Neo4j
vi) Αναπαράσταση με σύνδεση Neo4j σε sigma.js
vii) Δημιουργία Διεπαφής για εύκολη χρήση

1.3 Οργάνωση Κειμένου

Στο δεύτερο κεφάλαιο, γίνεται μια εισαγωγή στις Βάσεις Δεδομένων και η διαφοροποίηση των σχεσιακών Βάσεων Δεδομένων και των No SQL Databases. Στη συνέχεια, επικεντρωνόμαστε στη *Neo4j*, τη βάση δεδομένων που χρησιμοποιήσαμε για την αποθήκευση των δεδομένων μας. Θα δούμε τα χαρακτηριστικά των βάσεων δεδομένων αυτού του τύπου (*graph databases*), σε ποια από αυτά η *Neo4j* διαφέρει από τις υπόλοιπες βάσεις δεδομένων του ίδιου τύπου καθώς και τα ιδιαίτερα χαρακτηριστικά της που οδήγησαν στην επιλογή της. Τέλος, γίνεται μια συνοπτική αναφορά της γλώσσας ερωτημάτων *Cyphe*, γλώσσα που χρησιμοποιείται για τα queries από τη Neo4j.

Στο τρίτο κεφάλαιο της παρούσας διπλωματικής, περιγράφεται η πλατφόρμα *Node.js* και η γλώσσα Javascript. Βλέπουμε αναλυτικά τα προτερήματα που μας δίνει η χρήση της *Javascript* στην πλευρά του εξυπηρετητή (*server side*), και το που οφείλεται η ικανότητα του *Node* να πετυχαίνει πολύ υψηλά επίπεδα κλιμάκωσης (*scalability*). Στην συνέχεια, περιγράφουμε τον διαχειριστή πακέτων του *Node* (*Node Package Manager – NPM).* Στο τέλος του κεφαλαίου ορίζονται και εξηγούνται ο οδηγούμενος από γεγονότα προγραμματισμός και ο βρόχος γεγονότων.

Στο τέταρτο κεφάλαιο της παρούσας διπλωματικής, περιγράφουμε το κοινωνικό δίκτυο του *Twitter*. Παρουσιάζουμε τον τρόπο με τον οποίο συνδέονται μεταξύ τους οι χρήστες σε αυτό, καθώς και το είδος των πληροφοριών που ρέουν στο εσωτερικό του. Αναλύεται η χρησιμότητα των πληροφοριών αυτών και για ποιον λόγο επιβάλλεται η ανάπτυξη εφαρμογών που αξιοποιούν αυτές τις πληροφορίες. Παρουσιάζουμε τον τρόπο με τον οποίο μπορούμε να αποκτήσουμε πρόσβαση στα δεδομένα του *Twitter* μέσω 4 διεπαφών προγραμματισμού εφαρμογών (*Application Programming Interfaces – APIs*). Αναλύονται κάθε μία από αυτές τις διεπαφές και τους λόγους που μας οδήγησαν στην τελική επιλογή.

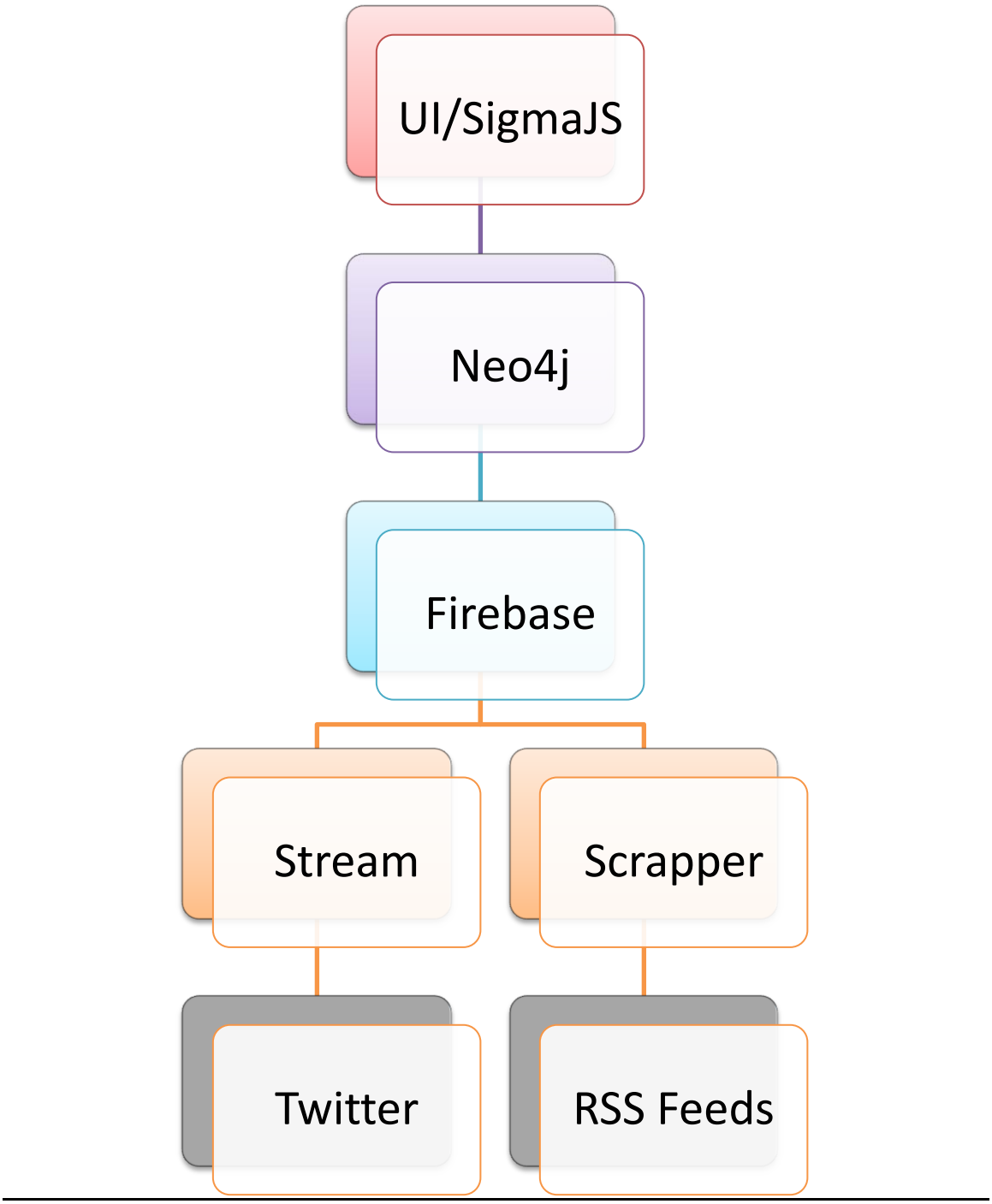
Στο πέμπτο κεφάλαιο και έχοντας δει αναλυτικά τα βασικά συστατικά της εφαρμογής μας, είμαστε πλέον σε θέση να περιγράψουμε πως αυτά συνδυάζονται για να δημιουργηθεί η εφαρμογή μας. Στο κεφάλαιο αυτό, περιγράφονται βήμα βήμα τα στάδια της εργασίας. Ξεκινάμε με τη διαδικασία συλλογής δεδομένων, με τη δημιουργία ενός RSS scrapper που τρέχει στο παρασκήνιο για ένα διάστημα 2 μηνών. Στη συνέχεια, αναφέρεται η Βάση στην οποία αποθηκεύονται τα δεδομένα, για τον έλεγχο και την ασφάλεια, σε περίπτωση λάθους. Έπειτα, εξηγείται η μέθοδος εξόρυξης των λέξεων-κλειδιών και η σύνδεση τους με

το Neo4j, για τη δημιουργία του γράφου. Παράλληλα, αναλύεται η αντίστοιχη διαδικασία, για τη συλλογή δεδομένων από το Twitter. Τέλος, περιγράφεται αναλυτικά η διεπαφή που δημιουργήθηκε για την εύκολη εμφάνιση των αποτελεσμάτων (των γράφων) και οι δυνατότητες που δίνει στους χρήστες.

Στο έκτο κεφάλαιο παρουσιάζονται τα αποτελέσματα που βρήκαμε μέσω της εφαρμογής. Αρχικά, περιγράφεται η λογική μέσα από την οποία, από τον αρχικό γράφο εντοπίζονται ισχυροί υπογράφοι και με την μελέτη αυτών προκύπτουν τα όποια αποτελέσματα. Έτσι, παρατίθενται παραδείγματα τόσο από την αρθρογραφία όσο και από το Twitter και με ποικίλια θεματολογία από πολιτικά ζητήματα, σε κοινωνικά σε αθλητικά θέματα. Στη συνέχεια, αξιολογείται το σύστημα τόσο από άποψη αποτελεσμάτων όσο και αρχιτεκτονικής και ευκολίας χρήσης τους, από χρήστες που δεν έχουν κάποιες ειδικές γνώσεις. Τέλος, παρουσιάζονται κάποια στατιστικά δεδομένα που παρατηρήθηκαν και καταμετρήθηκαν στη διάρκεια περάτωσης της εργασίας.

Στο τελευταίο κεφάλαιο συνοψίζονται τα βασικά στοιχεία της εφαρμογής και των αποτελεσμάτων της και τα συμπεράσματα που εξάγονται από την όλη διαδικασία. Δίνονται, επίσης, προβληματισμοί που προέκυψαν κατά τη διάρκεια της διπλωματικής εργασίας και ιδέες για μελλοντικές επεκτάσεις αυτής.

Στην επόμενη σελίδα παρατίθεται το διάγραμμα της δομής της εφαρμογής , όπου έχουμε σαν πηγές (sources) τα RSS Feeds και το Twitter, τα back end components που είναι ο Scrapper και ο Streamer, αντίστοιχα για τις 2 πηγές, που τρέχουν από πίσω και συλλέγουν δεδομένα. Στη συνέχεια περνάνε τα δεδομένα τους, στις βάσεις δεδομένων και καταλήγουν στην front end πλευρά της διεπαφής που εξυπηρετούν το χρήστη, με τη χρήση του User Interface(UI) που φτιάχτηκε και της βιβλιοθήκης Sigma.Js.



Block Diagram

2. Βάσεις Δεδομένων

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο εξετάζεται πως θα αποθηκεύσουμε κατάλληλα τα δεδομένα ώστε στην συνέχεια να μπορέσουμε να τα επεξεργαστούμε. Για τον σκοπό αυτό θα χρησιμοποιήσουμε βάσεις δεδομένων. Μια βάση δεδομένων είναι μια συλλογή δεδομένων οργανωμένη με τέτοιο τρόπο ώστε να καθιστά εύκολη την πρόσβαση, την διαχείριση και την ενημέρωση των δεδομένων αυτών. Αν και ο πιο διαδεδομένος τύπος είναι οι σχεσιακές (*relational*) βάσεις δεδομένων, δηλαδή η οργάνωση των δεδομένων σε μορφή πινάκων με τα ξένα κλειδιά να φανερώνουν τις σχέσεις μεταξύ αυτών, θα ακολουθηθεί στην προκειμένη περίπτωση μια διαφορετική προσέγγιση.
Τα τελευταία χρόνια έχει παρατηρηθεί μια ραγδαία αύξηση στη δημοτικότητα μιας οικογένειας τεχνολογιών αποθήκευσης δεδομένων, γνωστών και ως *NoSQL* (ακρωνύμιο του *Not Only SQL*). Αλλά ο όρος *NoSQL* χαρακτηρίζει περισσότερο τι δεν είναι – δεν είναι *SQL*-κεντρικές σχεσιακές βάσεις δεδομένων – παρά τι στην ουσία είναι αυτές οι βάσεις δεδομένων.
Ιστορικά οι περισσότερες διαδικτυακές εφαρμογές τρέχουν πάνω σε σχεσιακές βάσεις δεδομένων. Την τελευταία δεκαετία όμως τα δεδομένα που αντιμετωπίζουμε είναι πολύ μεγαλύτερα σε μέγεθος, αλλάζουν γρηγορότερα, και έχουν μεγάλη ποικιλία στη δομή τους, κάνοντας τα παραδοσιακά *RDBMS* συστήματα δύσκολο να ανταποκριθούν. Αυτές είναι και οι προκλήσεις που αντιμετωπίζουν οι *NoSQL* βάσεις δεδομένων.
Ο χρόνος εκτέλεσης των ερωτημάτων (*queries*) στις σχεσιακές βάσεις δεδομένων αυξάνεται, καθώς αυξάνεται το μέγεθος των πινάκων και ο αριθμός των ενώσεων (*joins*) που εκτελούνται. Αυτό δεν είναι λάθος των ίδιων των βάσεων, αλλά του τρόπου που αντιμετωπίζουν τα δεδομένα καθώς για να απαντήσουν σε ένα ερώτημα αρχικά βρίσκουν όλα τα πιθανά αποτελέσματα και στην συνέχεια τα φιλτράρουν. Αντίθετα οι *NoSQL* βάσεις δεδομένων υιοθετούν διαφορετικές προσεγγίσεις για να αντιμετωπίσουν μεγάλο όγκο δεδομένων, προκειμένου να αποφύγουν τις πολλές ενώσεις.
Εκτός από το μεγάλο τους μέγεθος, τα σημερινά δεδομένα συχνά αλλάζουν πολύ γρήγορα. Αυτός ο υψηλός ρυθμός αλλαγής των δεδομένων έχει υψηλό υπολογιστικό κόστος στις σχεσιακές βάσεις δεδομένων, καθώς για να γίνει μια αλλαγή πρέπει να προσπελάσουμε πολλούς πίνακες.
Πέρα όμως από τον υψηλό ρυθμό αλλαγής των ίδιων των δεδομένων, μπορεί να αλλάξει ακόμα και η δομή τους. Με άλλα λόγια, εκτός από την τιμή μιας συγκεκριμένης ιδιότητας μπορεί να αλλάξει και η ίδια η δομή των στοιχείων που έχουν αυτή την ιδιότητα. Οι σχεσιακές βάσεις δεδομένων έχουν αυστηρά καθορισμένο σχήμα, και απαιτούν τον αυστηρό ορισμό αυτού του σχήματος πριν οποιοδήποτε δεδομένο αποθηκευτεί σε αυτές. Η αλλαγή του σχήματος, αφού μπουν δεδομένα στην βάση, απαιτεί μεγάλο λειτουργικό κόστος και είναι μια διαδικασία που συχνά αποφεύγεται. Αντίθετα οι *NoSQL* βάσεις είναι απελευθερωμένες από αυτόν τον περιορισμό, καθώς δεν έχουν κάποιο προκαθορισμένο σχήμα (είναι *schema-free*). Έτσι επιτρέπουν στους προγραμματιστές εύκολα να ενσωματώνουν νέους τύπους δεδομένων ώστε να εμπλουτίζουν τις εφαρμογές τους.
Οι *NoSQL* βάσεις δεδομένων χωρίζονται σε 4 κατηγορίες με βάση τον τρόπο που μοντελοποιούν τα δεδομένα. Είναι οι *Document Stores*, οι *Key-Value Stores*, οι *Column Family* και οι βάσεις δεδομένων γράφου (*graph databases*). [1]

Η βάση δεδομένων που θα χρησιμοποιήσουμε πρέπει να ταιριάζει με την φύση των δεδομένων που θα συλλεχθούν. Πρώτον, τα δεδομένα που μαζεύονται, προέρχονται από άρθρα εφημερίδων και ένα κοινωνικό δίκτυο, το *Twitter*. Τα δεδομένα αυτής της μορφής έχουν πυκνές σχέσεις μεταξύ τους και μπορούν εύκολα να αναπαρασταθούν από έναν γράφο. Έτσι μια βάση δεδομένων γράφου ταιριάζει από την φύση της σε αυτόν τον έντονα δομημένο, γύρω από τις σχέσεις (*relationship-centered*) τομέα. Δεύτερον, επιθυμούμε τη χρήση του γράφου για την οπτική διάσταση που δίνει στα δεδομένα , καθώς σε αυτή την οπτικοποίηση θα βασιστεί η εξαγωγή αποτελεσμάτων. Έχοντας αυτά κατά νου, επιλέξαμε μια βάση δεδομένων γράφου, καθώς θεωρήσαμε ότι μπορεί να μοντελοποιήσει καλύτερα τον τομέα του προβλήματος που αντιμετωπίζουμε. Τελικά καταλήξαμε στην *Neo4j,* μάλλον την δημοφιλέστερη βάση δεδομένων γράφου αυτήν τη στιγμή.

2.2 Neo4j

Αφού είδαμε τους λόγους για τους οποίους επιλέξαμε μια βάση δεδομένων γράφου (*graph database*) για την ανάπτυξη της εφαρμογής μας, στην ενότητα αυτή θα δούμε τα βασικά χαρακτηριστικά των βάσεων δεδομένων αυτού του τύπου, επικεντρώνοντας στην *Neo4j*.
Γράφος είναι μια συλλογή κορυφών και ακμών ή πιο απλά ένα σύνολο κόμβων και σχέσεων που ενώνουν αυτούς τους κόμβους. Μια βάση δεδομένων γράφου είναι μια βάση δεδομένων ή οποία χρησιμοποιεί δομές γράφου για να αναπαραστήσει και να αποθηκεύσει δεδομένα. Ένα σύστημα διαχείρισης μιας βάσης δεδομένων γράφου (*Graph Database Management System*) είναι ένα σύστημα διαχείρισης βάσεων δεδομένων με *CREATE*, *READ*, *UPDATE*, *DELETE* (*CRUD*) μεθόδους, οι οποίες εφαρμόζονται πάνω σε δεδομένα τα οποία είναι δομημένα σε μορφή γράφου. Μια ιδιαιτερότητα των βάσεων δεδομένων γράφου είναι ο τρόπος που μοντελοποιούν το εκάστοτε πρόβλημα. Μοντελοποίηση είναι η αφαιρετική διαδικασία κατά την οποία περνάμε συγκεκριμένες πτυχές ενός τομέα του πραγματικού κόσμου σε έναν τομέα στον οποίο μπορούν να δομηθούν και να τις χειριστούμε. Ο γράφος είναι ο φυσικός τρόπος με τον οποίο ο άνθρωπος προσπαθεί να αναπαραστήσει ένα πρόβλημα, χρησιμοποιώντας κύκλους και τετράγωνα και στην συνέχεια βελάκια που τα ενώνουν για να δείξει τις σχέσεις μεταξύ αυτών. Με αυτόν τον τρόπο οι βάσεις δεδομένων γράφου, μας επιτρέπουν να δημιουργήσουμε εξελιγμένα μοντέλα που ταιριάζουν πολύ καλά με τον τομέα του προβλήματος που επιθυμούμε να λύσουμε. Αυτά τα μοντέλα είναι απλά και συγχρόνως πιο εκφραστικά από τα μοντέλα που δημιουργούνται από τις παραδοσιακές σχεσιακές βάσεις δεδομένων ή άλλες *NoSQL* δομές.
Το βασικό χαρακτηριστικό των βάσεων δεδομένων γράφου είναι ότι οι σχέσεις μεταξύ των κόμβων είναι κανονικές οντότητες. Αυτό σημαίνει ότι μπορούμε να τις δούμε απευθείας, απλά κοιτώντας την μνήμη. Αντίθετα σε άλλες τεχνολογίες βάσεων δεδομένων οι σχέσεις μεταξύ των οντοτήτων υπονοούνται, και πρέπει εμείς να τις συμπεράνουμε συχνά χρησιμοποιώντας επινοημένες ιδιότητες όπως το ξένο κλειδί (*foreign key*), ή επεξεργασίες όπως η *map-reduce*.
Ειδικά όταν μιλάμε για δεδομένα πολύ στενά συνδεδεμένα με σχέσεις διαφορετικών τύπων ή για ερωτήματα διάσχισης του γράφου, οι βάσεις αυτές έχουν πολύ υψηλή απόδοση . Αυτό συμβαίνει επειδή τα ερωτήματα εξετάζουν ένα τμήμα του γράφου αντί το σύνολο των δεδομένων και έτσι αποφεύγονται οι πολλές ενώσεις (*joins*) των σχεσιακών

βάσεων. Έτσι η καθυστέρηση απόκρισης (*latency*) εξαρτάται από το μέγεθος του τμήματος του γράφου που επιλέγουμε να εξερευνήσουμε και όχι από το όγκο των αποθηκευμένων δεδομένων.
Όπως έχουμε ήδη αναφέρει οι βάσεις δεδομένων γράφου ταιριάζουν εξαιρετικά καλά στον τομέα των κοινωνικών δικτύων. Τα δεδομένα αυτών των δικτύων σχετίζονται στενά και με πολλαπλούς τρόπους μεταξύ τους. Περιμένουμε λοιπόν οι βάσεις δεδομένων γράφου να έχουν πολύ καλύτερη απόδοση από τις σχεσιακές σε ερωτήματα πάνω σε αυτά τα δεδομένα. [*2*]
Πέρα από την απόδοση τους, ένα άλλο χαρακτηριστικό των βάσεων δεδομένων γράφου είναι το μη προκαθορισμένο σχήμα τους (*schema-free*). **Έτσι, είναι πολύ εύκολο να προσθέσουμε νέου τύπου δεδομένα στην υπάρχουσα δομή, όπως κάποιο νέο είδος κόμβων, σχέσεων και ιδιοτήτων ή ακόμα και κάποιον ολόκληρο υπογράφο χωρίς να πειράξουμε τα υπάρχοντα ερωτήματα και την λειτουργικότητα της εφαρμογής. Αυτό οδηγεί σε αύξηση της ευελιξίας**. Λόγω αυτής της ευελιξίας δεν χρειάζεται να μοντελοποιήσουμε από την αρχή πλήρως το πρόβλημα μας.

Τα βασικά χαρακτηριστικά της Neo4j είναι τα εξής:

- Περιέχει κόμβους, σχέσεις και ιδιότητες (ζευγάρια *"key-value"*).
- Οι κόμβοι αντιπροσωπεύουν τις οντότητες, έχουν ταμπέλες και περιέχουν ιδιότητες. Τα κλειδιά (*keys)* των ιδιοτήτων είναι συμβολοσειρές (*strings)* και οι τιμές τους
  (*values)* είναι αυθαίρετοι τύποι δεδομένων.
- Οι σχέσεις συνδέουν τους κόμβους. Έχουν κατεύθυνση, μια ταμπέλα, και έναν αρχικό και τελικό κόμβο - δεν υπάρχει αιωρούμενη σχέση.
- Όπως και οι κόμβοι έτσι και οι σχέσεις μπορούν να έχουν ιδιότητες. Αυτή η ικανότητα των σχέσεων να έχουν ιδιότητες είναι πολύ χρήσιμη τόσο σε διάφορους αλγορίθμους που εφαρμόζονται πάνω στους γράφους όσο και στον περιορισμό των ερωτημάτων καθώς αυτά εκτελούνται, καθώς προσθέτει στις σχέσεις σημασιολογία (βάρος, ποιότητα).

Το μοντέλο πάνω στο οποίο είναι δομημένη η *Neo4j* δεν διαφέρει σε πολλά από αυτό των περισσότερων βάσεων δεδομένων. Η *Neo4j* έχει επιπλέον δύο χαρακτηριστικά. Χρησιμοποιεί κανονική επεξεργασία γράφου (*native graph processing*) και κανονική αποθήκευση γράφου (*native graph storage*). Το πρώτο αφορά στον τρόπο που διαχειρίζεται τα ερωτήματα ενώ το δεύτερο στον τρόπο που αποθηκεύει την πληροφορία.[3]
Μια βάση δεδομένων γράφου λέμε ότι χρησιμοποιεί **κανονική επεξεργασία γράφου** (*native graph processing*) αν διαθέτει μια ιδιότητα που ονομάζεται γειτνίαση χωρίς ευρετήριο (*index-free adjacency*). Αυτό σημαίνει ότι κάθε κόμβος διατηρεί απευθείας αναφορές σε όλους τους γειτονικούς του κόμβους, και έτσι λειτουργεί σαν ένα μικρό ευρετήριο για την γειτονία του. Έτσι ο χρόνος απόκρισης σε ερωτήματα (*queries*) εξαρτάται αποκλειστικά από το μέγεθος αυτής της γειτονιάς και όχι από το μέγεθος του γράφου.
Αντίθετα, μια βάση δεδομένων που δεν έχει αυτήν την ιδιότητα χρησιμοποιεί ευρετήρια για να για να συνδέσει τους κόμβους μεταξύ τους. Αυτά τα ευρετήρια αυξάνουν το υπολογιστικό κόστος όταν προσπαθούμε να διασχίσουμε τον γράφο. [4]
Μια βασική πτυχή του σχεδιασμού μια βάσης δεδομένων είναι και ο τρόπος που αποθηκεύονται τα δεδομένα (στην περίπτωση μας οι γράφοι) στην μνήμη. Όπως αναφέραμε η *Neo4j* χρησιμοποιεί **κανονική αποθήκευση γράφου** (*native graph storage*),

κάτι που οδηγεί σε υψηλή απόδοση στα ερωτήματα διάσχισης. Η *Neo4j* αποθηκεύει τα διάφορα δεδομένα σε διαφορετικά αρχεία αποθήκευσης. Κάθε ένα από αυτά τα αρχεία έχει πληροφορία για ένα τμήμα του γράφου (αλλού οι κόμβοι, αλλού οι σχέσεις, αλλού οι ιδιότητες και τα λοιπά). Το γεγονός αυτό, και κυρίως το γεγονός ότι διαχωρίζει τα δομικά στοιχεία του γράφου από τις ιδιότητες, οδηγεί σε πολύ γρήγορες διασχίσεις του γράφου.

Η γλώσσα για ερωτήματα που χρησιμοποιεί η *Neo4j* είναι η **Cypher**. Η *Cypher* είναι μια δηλωτική γλώσσα ερωτημάτων για βάσεις δεδομένων γράφου. Είναι πολύ εκφραστική, από την άποψη ότι μοιάζει πολύ με τον τρόπο που ο άνθρωπος αντιλαμβάνεται διαισθητικά τους γράφους. Επειδή ακριβώς είναι δηλωτική, επικεντρώνεται στο να ορίσει με σαφήνεια τι να ανασύρουμε από τον γράφο και όχι πώς να το ανασύρουμε.[5] Για όλα τα queries αυτής της εργασίας, χρησιμοποιείται η γλώσσα Cypher.
Τέλος, η *Neo4j* είναι μια βάση δεδομένων γράφου με ιδιότητες (*property graph database*). Αυτό σημαίνει ότι κάθε σχέση έχει, πέρα από ιδιότητες και ταμπέλες, έναν κόμβο αρχής και έναν κόμβο πέρατος και συνεπώς κατεύθυνση. Πολλές φορές όμως, στα πλαίσια μιας εφαρμογής όπως και της παρούσας, δεν θέλουμε οι σχέσεις να έχουν κατεύθυνση. Στην *περίπτωση αυτή* αντί για σχέσεις διπλής κατεύθυνσης, μπορούμε απλά να αγνοήσουμε την κατεύθυνση των σχέσεων κατά την διάρκεια των ερωτημάτων (*queries*).

3. Javascript/NodeJS

Η γλώσσα JavaScript είναι μια διερμηνευτική γλώσσα προγραμματισμού η οποία αποτελεί μέρος υλοποίησης των φυλλομετρητών Ιστού ώστε να μπορεί ο χρήστης να αλληλεπιδρά με αυτούς ανταλλάσσοντας ασύγχρονα δεδομένα. Χωρίς αυτήν για παράδειγμα δεν θα μπορούσε μια ιστοσελίδα να ανιχνεύσει μια κίνηση του χρήστη, όπως πότε πατάει ένα κουμπί. Αυτή δίνει το δυναμικό χαρακτήρα στις σελίδες και κυρίως στα δομικά στοιχεία αυτής.[6]

Το Node.js είναι μια πλατφόρμα που επιτρέπει τη συγγραφή και την εκτέλεση JavaScript προγραμμάτων, χωρίς την ανάγκη ενός φυλλομετρητή ιστού (web browser). Για να το πετύχει αυτό, μεταγλωττίζει σε γλώσσα μηχανής και «τρέχει» τα προγράμματα, κάνοντας χρήση της V8, μιας εικονικής μηχανής που αναπτύχθηκε από τη Google για τις ανάγκες του Chrome και στη συνέχεια εκδόθηκε ανεξάρτητα. Η *V8* δίνει στο *Node* μεγάλη αύξηση στην απόδοση, καθώς κάνει απευθείας μεταγλώττιση σε γλώσσα μηχανής αντί να εκτελεί *bytecode* ή να χρησιμοποιεί διερμηνευτή (*interpreter*).
Βασική φιλοσοφία του Node.js είναι ότι ο κώδικας που δεν πραγματοποιεί Είσοδο/Έξοδο δεδομένων εκτελείται ταχύτατα και σύγχρονα, ωστόσο η Είσοδος/Έξοδος εκτελείται ασύγχρονα. Κάθε φορά δηλαδή που το πρόγραμμά προσπαθεί να «διαβάσει» κάτι από κάποιο αρχείο ή πηγή στο διαδίκτυο, η εφαρμογή δεν παγώνει περιμένοντας, αλλά θέτει μία συνάρτηση που θα εκτελεστεί όταν η είσοδος διαβαστεί και συνεχίζει κανονικά την εκτέλεσή του χωρίς να σταματά για ΕΕ. [7]
Το γεγονός, λοιπόν, ότι το *Node* μας δίνει την δυνατότητα να χρησιμοποιήσουμε την ίδια γλώσσα και στις δύο άκρες, τόσο στον εξυπηρετητή (*server*) όσο και στον εξυπηρετούμενο (*client*), απλοποιεί την ανάπτυξη διαδικτυακών εφαρμογών. Με αυτόν τον τρόπο, η γνώση *JavaScript* γίνεται το μόνο προαπαιτούμενο για να αναπτύξει κανείς μια τέτοια εφαρμογή, αλλά ταυτόχρονα αυξάνει και την απόδοση τους.

Ένα από τα δυνατά σημεία του Node.js είναι, εκτός των άλλων, η μεγάλη κοινότητα ανοικτού λογισμικού που το υποστηρίζει. Συνεχώς προστίθενται νέα εργαλεία, επεκτάσεις και βιβλιοθήκες που είναι γραμμένα για την πλατφόρμα Node.js. Οι επεκτάσεις αυτές δημοσιεύονται σε μία δημόσια αποθήκη πακέτων Node στο διαδίκτυο, και μπορούν να εγκατασταθούν πολύ εύκολα σε μία εφαρμογή με χρήση του διαχειριστή πακέτων Node.js, NPM (Node Package Manager) [8]. Με την εγκατάσταση του *Node*, είμαστε σε θέση να χρησιμοποιήσουμε κάποια ενσωματωμένα πακέτα (*built-in modules*), τα οποία μας παρέχουν κάποιες διεπαφές εφαρμογών χαμηλού επιπέδου (*low-level APIs*) ώστε να μπορούμε να υλοποιήσουμε κάποιες βασικές λειτουργίες που χρειάζονται οι εφαρμογές. Αυτά τα πακέτα είναι γνωστά ως ο πυρήνας του *Node* (*Node core*) [*9*]. Στα πλαίσια όμως σύνθετων εφαρμογών είναι σχεδόν απαραίτητο να συμπεριλάβουμε πακέτα τρίτων. Ο διαχειριστής πακέτων του *Node* διατηρεί μια κεντρική αποθήκη όλων των πακέτων που οι προγραμματιστές δημοσιοποιούν και μας παρέχει ένα εργαλείο γραμμής-εντολών (*command-line tool*) ώστε να μπορούμε να κατεβάζουμε, να εγκαθιστούμε και να διαχειριζόμαστε αυτά τα πακέτα.

Στη συνέχεια θα οριστεί ο **οδηγούμενος από γεγονότα προγραμματισμό** και ο **βρόχος γεγονότων**. Ο οδηγούμενος από τα γεγονότα (*event-driven*) προγραμματισμός είναι ένα προγραμματιστικό μοντέλο, όπου η ροή της εκτέλεσης "οδηγείται" από τα γεγονότα. Ο χειρισμός των γεγονότων αυτών γίνεται από τους χειριστές συμβάντος (*event handlers*) ή τις επιστρεφόμενες κλήσεις γεγονότων (*event callbacks*). Μια επιστρεφόμενη κλήση είναι μια συνάρτηση η οποία καλείται όταν συμβαίνει ένα γεγονός. Στα πλαίσια του εξυπηρετητή το γεγονός αυτό είναι συνήθως η ολοκλήρωση μιας λειτουργίας Ε/Ε. Αυτός ο τρόπος προγραμματισμού, όπου οι λειτουργίες Ε/Ε δεν επιστρέφουν απλά μια τιμή, αλλά συνάρτηση(callback function) ονομάζεται προγραμματισμός οδηγούμενος από τα γεγονότα ή ασύγχρονος προγραμματισμός. Με αυτόν τον τρόπο πολλές λειτουργίες Ε/Ε μπορούν να συμβαίνουν παράλληλα, και κάθε φορά η συνάρτηση επιστρεφόμενης κλήσης μας ενημερώνει για την ολοκλήρωση της αντίστοιχης λειτουργίας. Η *JavaScript* λόγω του τρόπου που αντιμετωπίζει τις συναρτήσεις, ταιριάζει φυσικά σε αυτό το είδος προγραμματισμού, καθώς υποστηρίζει το μοτίβο κλεισίματος (*closure pattern*) και το πέρασμα συναρτήσεων σαν παραμέτρους σε άλλη συνάρτηση (*first-class functions*).
Ο οδηγούμενος από τα γεγονότα προγραμματισμός συνοδεύεται από έναν βρόχο γεγονότων (***event loop***). Ο βρόχος γεγονότων είναι μια κατασκευή που είναι υπεύθυνη για 2 πράγματα σε κάθε επανάληψη: την ανίχνευση γεγονότων (*event detection*) και την ενεργοποίηση του αντίστοιχου χειριστή συμβάντος (*event handler triggering*). Ο βρόχος γεγονότων λοιπόν, ελέγχει σε κάθε επανάληψη ποια λειτουργία Ε/Ε έχει ολοκληρωθεί και καλεί την αντίστοιχη συνάρτηση επιστρεφόμενης κλήσης. Ο βρόχος γεγονότων είναι απλά ένα νήμα μέσα σε μια διεργασία και συνεπώς έχει τα εξής 2 χαρακτηριστικά [*10*]:
• Κάθε χρονική στιγμή τρέχει το πολύ ένας διαχειριστής γεγονότος.
• Κάθε διαχειριστής γεγονότος τρέχει χωρίς διακοπή μέχρι να ολοκληρωθεί

4. Twitter

4.1 Γενική Περιγραφή

Το *Twitter* είναι μια ιστοσελίδα κοινωνικής δικτύωσης. Ιδρύθηκε τον Μάρτιο του 2006 από τον *Jack Dorsey* και από τα πρώτα στάδια της ζωής του γνώρισε ραγδαία ανάπτυξη. Σήμερα μετρά περισσότερους από 270 εκατομμύρια ενεργούς χρήστες που δημοσιεύουν 500 εκατομμύρια τιτιβίσματα (*tweets*) την ημέρα.
Κατά πόσο το *Twitter* αποτελεί κοινωνικό δίκτυο (*social network*) ή δίκτυο πληροφοριών (*information network*) αποτελεί αντικείμενο αντιπαράθεσης. Οι *Myers*, *Sharma*, *Gupta* και *Lin* στην εργασία τους [*11*] μελετάνε τα χαρακτηριστικά του γράφου των ακολούθων (*follow graph*) του *Twitter* ώστε να αποφανθούν αν πρόκειται για κοινωνικό ή πληροφοριακό δίκτυο. Ο γράφος ακολούθων ενός κοινωνικού δικτύου χαρακτηρίζεται από μικρά συντομότερα μονοπάτια μεταξύ των χρηστών, μεγάλες συνδεδεμένες συνιστώσες (*connected components*), μεγάλο συντελεστή ομαδοποίησης (*clustering coefficient*) και υψηλό βαθμό αμοιβαιότητας (*reciprocity*), δηλαδή υψηλή τάση οι σχέσεις μεταξύ των χρηστών να εμφανίζονται σε ζεύγη. Το *Facebook* είναι χαρακτηριστικό παράδειγμα κοινωνικού δικτύου. Αντίθετα στα δίκτυα πληροφοριών ο γράφος ακολούθων χαρακτηρίζεται από μεγάλο αριθμό ακμών στους κόμβους (*vertex degree*), έλλειψη αμοιβαιότητας και μεγάλο αριθμό γειτόνων δύο επιπέδων (*two-hop neighbors*). Τα αποτελέσματα της έρευνας που έγινε σε 175 εκατομμύρια χρήστες του *Twitter*, έδειξε ότι αυτό παρουσιάζει χαρακτηριστικά τόσο κοινωνικού όσο και πληροφοριακού δικτύου. Αρχικά συμπεριφέρεται σαν δίκτυο πληροφοριών και εξελίσσεται σε κοινωνικό δίκτυο. Συγκεκριμένα, οι πρώτοι χρήστες τους οποίους ένας καινούριος χρήστης αποφασίζει να ακολουθήσει είναι συνήθως διάσημοι. Με τον καιρό και καθώς αποκτά εμπειρία στην χρήση του *Twitter,* γίνεται πιο επιλεκτικός και επιλέγει ποιους θα ακολουθήσει με άλλα κριτήρια πέρα από την διασημότητα. Έτσι, εισέρχεται σε κοινότητες χρηστών με τους οποίους έχει κοινά ενδιαφέροντα ή άλλες κοινωνικές σχέσεις, μετατρέποντας με αυτόν τον τρόπο το *Twitter* σε κοινωνικό δίκτυο.
Η βασική του ιδέα είναι ότι οι χρήστες μπορούν να δημοσιεύσουν ανά πάσα στιγμή μηνύματα κειμένου μήκους έως και 140 χαρακτήρων, γνωστά και ως τιτιβίσματα ή "*tweets*".
Όπως σε κάθε κοινωνικό δίκτυο έτσι και στο *Twitter* υπάρχουν σχέσεις μεταξύ των χρηστών του δικτύου. Στο *Twitter* βασική έννοια στις σχέσεις μεταξύ των χρηστών είναι αυτή του ακόλουθου (*follower*). Κάθε χρήστης μπορεί να ακολουθήσει (κάνει *follow*) οποιονδήποτε άλλον χρήστη επιθυμεί, χωρίς να χρειάζεται η συγκατάθεση του τελευταίου. Με αυτόν τον τρόπο γίνεται ακόλουθος (*follower*) και ενημερώνεται αυτόματα για τα τιτιβίσματα του χρήστη που ακολουθεί.
Άλλη μια βασική έννοια στο *Twitter* που θα μας απασχολήσει ιδιαιτέρως είναι αυτή του *hashtag*. *Hashtag* είναι η λέξη ενός τιτιβίσματος που ακολουθεί τον ειδικό χαρακτήρα "#". Το *hashtag* εκφράζει το θέμα του τιτιβίσματος και βοηθάει στην ομαδοποίηση των τιτιβισμάτων που έχουν το ίδιο θέμα.
Όμοια ο ειδικός χαρακτήρας "@" ακολουθούμενος από το όνομα κάποιου χρήστη χρησιμοποιείται για να απευθυνθούμε σε αυτόν τον χρήστη και να ξεκινήσουμε μια συζήτηση μαζί του. Επίσης υπάρχει η δυνατότητα να προωθήσουμε ένα τιτίβισμα στους ακόλουθους μας κάνοντας το αναδημοσίευση ή "*retweet*". Σε αυτήν την περίπτωση χρησιμοποιούμε το σύμβολο "*RT*".
Τα τελευταία χρόνια έχει γίνει αντικείμενο μελέτης τόσο ο τρόπος διάδοσης της

πληροφορίας στο *Twitter* μέσω της δημοσίευσης και της αναδημοσίευσης περιεχομένου από τους χρήστες, όσο και η αλλαγή του υποκείμενου κοινωνικού δικτύου μέσω της δημιουργίας και της καταστροφής σχέσεων μεταξύ των χρηστών. Μεγάλο ενδιαφέρον παρουσιάζει η σύνδεση που υπάρχει μεταξύ αυτών των δυο δυναμικών. Δηλαδή πως επηρεάζει η διάδοση της πληροφορίας την δομή του ίδιου του δικτύου.
Οι *Myers* και *Leskovec* μελετούν στην εργασία τους [*12*] τον τρόπο που η δομή του δικτύου του *Twitter* αντιδρά καθώς οι χρήστες δημοσιεύουν και αναδημοσιεύουν περιεχόμενο. Όταν οι χρήστες επιλέγουν ποιους χρήστες θα ακολουθήσουν, επιλέγουν εμμέσως και σε ποιες πληροφορίες θα έχουν πρόσβαση. Συνεπώς η εξέλιξη της δομής του δικτύου δεν είναι ανεξάρτητη από την διάχυση της πληροφορίας σε αυτό.
Οι *Myers* και *Leskovec* παρατήρησαν ότι η δομή του δικτύου του *Twitter* αλλάζει με έναν σταθερό ρυθμό, ο οποίος διακόπτεται από απότομα ξεσπάσματα που οφείλονται στη διάχυση της πληροφορίας σε αυτό. Πρώτον, οι χρήστες σταματάνε να ακολουθούν κάποιον χρήστη όταν ο τελευταίος δημοσιεύει τιτιβίσματα με πολύ υψηλό ρυθμό. Αυτό ονομάζεται ξέσπασμα "*tweet-unfollow*". Δεύτερον, αν τα τιτιβίσματα ενός χρήστη αναδημοσιεύονται (*retweet*) συχνά από τους ακόλουθους του, τότε είναι πιθανόν αυτός ο χρήστης να κερδίσει περισσότερους ακόλουθους. Καθώς το τιτίβισμα διαχέεται στο δίκτυο μέσω των αναδημοσιεύσεων, όλο και περισσότεροι χρήστες έρχονται σε επαφή με το αρχικό τιτίβισμα. Αν αυτοί οι χρήστες το θεωρήσουν ενδιαφέρον είναι πιθανό να ακολουθήσουν τον χρήστη που το δημιούργησε. Αυτό ονομάζεται "*retweet-follow*". Και για τους δύο τύπους ξεσπασμάτων παρατηρήθηκε μια μεγάλη αύξηση στην ομοιότητα των χρηστών με τον αρχικό χρήστη, και συνεπώς αύξηση στην ομοιογένεια και συνεκτικότητα του δικτύου στη γειτονία αυτού του χρήστη.

4.2 Twitter API

Το *Twitter* παρέχει στους προγραμματιστές 4 διεπαφές προγραμματισμού εφαρμογών (*Application Programming Interfaces* ή πιο γνωστές ως *APIs*), ώστε οι εφαρμογές τους να μπορούν να επικοινωνούν με το *Twitter*. Οι εφαρμογές αυτές μπορούν να αλληλεπιδρούν με το *Twitter* έχοντας είτε δικαιώματα μόνο για ανάγνωση (*read only permission*) είτε δικαιώματα ανάγνωσης/εγγραφής (*read/write permission*). Στην πρώτη περίπτωση μπορούμε απλά να διαβάσουμε δεδομένα από το *Twitter*, ενώ στην δεύτερη μπορούμε επιπλέον να κάνουμε και αλλαγές στο προφίλ μας. Στα πλαίσια της εφαρμογής όπως αυτή θα παρουσιαστεί και στη συνέχεια, τα δικαιώματα μόνο για ανάγνωση είναι αρκετά, καθώς μας ενδιαφέρει μόνο να παίρνουμε δεδομένα από το *Twitter* .
Για να μπορέσουμε να δημιουργήσουμε μια εφαρμογή που να επικοινωνεί με το *Twitter* πρέπει αρχικά να έχουμε έναν λογαριασμό στο *Twitter*. Στην συνέχεια, η εφαρμογή πρέπει να δηλωθεί στην επίσημη σελίδα των Twiiter applications και να οριστούν τα δικαιώματα που θα έχει. Έπειτα, λαμβάνουμε από το *Twitter* κάποια διαπιστευτήρια για την εφαρμογή.
Το *Twitter* χρησιμοποιεί το ***OAuth*,** μια μέθοδο πιστοποίησης που δεν απαιτεί από τον χρήστη να δώσει τον κωδικό του στην εφαρμογή.
Στην συνέχεια παρουσιάζονται τα βασικά χαρακτηριστικά της καθεμιάς από τις 4 διεπαφές εφαρμογών (*APIs*) του *Twitter*.[13]

4.2.1 Website

Η συγκεκριμένη διεπαφή παρέχει μια σειρά από υπηρεσίες που προσφέρουν την δυνατότητα σε ιστοσελίδες να χρησιμοποιούν κάποιες πολύ βασικές λειτουργίες του *Twitter*. Χαρακτηριστικά παραδείγματα είναι τα κουμπιά "*tweet*" και "*follow*" που βλέπουμε πολλές φορές σε διάφορες ιστοσελίδες

4.2.2 Search API

Το *Search API* μας δίνει την δυνατότητα να υποβάλουμε στο *Twitter* ένα ερώτημα για κάποιο περιεχόμενο που μας ενδιαφέρει. Μπορούμε δηλαδή να αναζητήσουμε κάποιο τιτίβισμα με βάση κάποια λέξη ή φράσεις κλειδιά, με βάση κάποιο *hashtag*, τον χρήστη που το δημοσίευσε, τη γλώσσα στην οποία είναι γραμμένο, τη γεωγραφική θέση και άλλα. Να σημειώσουμε ότι η συγκεκριμένη διεπαφή είναι μόνο για ανάγνωση (*read only*).
Ένας σημαντικός περιορισμός της συγκεκριμένης διεπαφής είναι το όριο χρήσης (*rate limit*). Το Search API επιστρέφει κάθε φορά μέχρι 100 tweets και σε μέγιστος βάθος χρόνου μία εβδομάδα πριν τη στιγμή που γίνεται η αναζήτηση. Παράλληλα, δεν μπορούμε να υποβάλουμε στο *Twitter* περισσότερα από 180 ερωτήματα ανά 15 λεπτά.

4.2.3 REST API

Το *REST API* μας δίνει την δυνατότητα να χρησιμοποιήσουμε όλες τις βασικές λειτουργίες που μπορεί να κάνει κανείς στο *Twitter*. Μπορούμε να δημοσιεύσουμε ή να αναδημοσιεύσουμε (*retweet*) κάποιο μήνυμα και να ακολουθήσουμε (*follow*) ή να πάψουμε να ακολουθούμε (*unfollow*) κάποιον χρήστη. Το *REST API* μας δίνει ακόμα περισσότερες δυνατότητες. Μπορούμε να πάρουμε το χρονολόγιο (*timeline*), κάτι σαν το ιστορικό των τιτιβισμάτων ενός χρήστη, ή ακόμα και να βρούμε την λίστα των χρηστών που ακολουθούν ή ακολουθούνται από έναν χρήστη. Παρά τις μεγάλες δυνατότητες που προσφέρει υπόκειται και αυτό στους περιορισμούς ορίου χρήσης (*rate limit*) ,όπως αναφέρθηκαν παραπάνω.

4.2.4 Streaming API

Το *Streaming API* απευθύνεται σε εφαρμογές που ενδιαφέρονται για μεγάλο όγκο δεδομένων σε πραγματικό χρόνο. Ουσιαστικά μας δίνει ένα πραγματικού χρόνου (*real time*) δείγμα της τάξης του 1% του δημοσίου χρονολογίου (*public timeline*). Το δημόσιο χρονολόγιο είναι το σύνολο των τιτιβισμάτων που γίνονται στο *Twitter*, γνωστό και ως *Firehose*. Αν και σε πρώτη ανάγνωση το νούμερο μοιάζει μικρό, τα δεδομένα που μας παρέχονται με αυτόν τον τρόπο είναι υπεραρκετά για τα πλαίσια των περισσοτέρων εφαρμογών. Όπως γίνεται κατανοητό από τα παραπάνω, η συγκεκριμένη διεπαφή είναι μόνο για ανάγνωση (*read only*). [14]

Η συγκεκριμένη διεπαφή διαφέρει από τις δύο προηγούμενες στον τρόπο που επικοινωνεί με το *Twitter*. Δημιουργεί αρχικά μια μόνιμη σύνδεση με το *Twitter*, και έκτοτε τραβάει τιτιβίσματα από εκεί για όσο χρόνο η σύνδεση παραμένει ενεργή. Εδώ πρέπει να προσέξουμε τους λόγους για τους οποίους το *Twitter* μπορεί να διακόψει μια σύνδεση. Πρώτον, δεν επιτρέπεται να δημιουργήσουμε δύο συνδέσεις με τα ίδια διαπιστευτήρια. Αυτό θα οδηγήσει την πρώτη σύνδεση σε διακοπή. Δεύτερον, αν ο ρυθμός με τον οποίο λαμβάνουμε τα δεδομένα είναι πολύ αργός πάλι θα οδηγηθούμε σε διακοπή. Κάθε σύνδεση έχει μια ουρά με τιτιβίσματα τα οποία πρέπει να παρέχει στην εφαρμογή. Αν αυτή η ουρά αυξάνεται πολύ γρήγορα με τον χρόνο, τότε αυτή η σύνδεση θα διακοπεί.
Το *Streaming API* χωρίζεται στο *Sample Stream* και στο *Filter Stream*. Το *Sample stream* τραβάει τυχαία [*15*] τιτιβίσματα από το δημόσιο χρονολόγιο (*public timeline*).
Αντίθετα το *Filter stream* αποτελεί ένα υποσύνολο του *Firehose* με βάση κάποια κριτήρια. Τα κριτήρια αυτά μπορεί να είναι μια λέξη ή φράση κλειδί, ο δημιουργός του τιτιβίσματος, ή η τοποθεσία που γίνεται αυτό. Να σημειώσουμε εδώ ότι αν τα τιτιβίσματα που ταιριάζουν στα κριτήρια μας, ξεπερνάνε το 1% όλων των τιτιβισμάτων που γίνονται στο *Twitter*, τότε η απάντηση που λαμβάνουμε περιορίζεται ώστε να μην ξεπερνάει αυτό το όριο.
Τέλος, η συγκεκριμένη διεπαφή δεν υπόκειται σε περιορισμούς ορίου χρήσης (*rate limit*), με την μορφή που τους συναντήσαμε στις δύο προηγούμενες διεπαφές. Εδώ ο περιορισμός αφορά το πλήθος των φορών που δοκιμάζουμε να δημιουργήσουμε μια σύνδεση στο *Twitter* με τα ίδια διαπιστευτήρια. Σε περίπτωση που ξεπεράσουμε το όριο, το *Twitter* μας στέλνει μια *HTTP 420* απάντηση σε κάθε αίτημα μας για νέα σύνδεση.

Από τις τέσσερις διεπαφές εφαρμογών (*APIs*) που παρέχει το *Twitter*,πρέπει να επιλεχθεί ποια καλύπτει καλύτερα τις ανάγκες της συγκεκριμένης εφαρμογής. Η διεπαφή εφαρμογών που θα διαλέξουμε πρέπει να μπορεί να εφοδιάσει την εφαρμογή μας με μεγάλο όγκο δεδομένων .
Αρχικά αποκλείουμε το *Twitter for Websites* καθώς ο σκοπός του δεν είναι να παρέχει δεδομένα του *Twitter* σε εφαρμογές. Από τα υπόλοιπα, το *Search API* και το *REST API* επικοινωνούν με το *Twitter* μέσω ερωτημάτων σε μια βάση δεδομένων. Αυτό δημιουργεί δύο σημαντικούς περιορισμούς. Πρώτον, αυτά τα ερωτήματα απαιτούν την δοσοληψία με μια βάση δεδομένων και συνεπώς είναι χρονοβόρα. Δεύτερον, όλα τα δεδομένα που μαζεύουμε με αυτόν τον τρόπο συλλέγονται με πολύ αργούς ρυθμούς εξαιτίας των ορίων χρήσης (*rate limits*) που έχουν οι δύο αυτές διεπαφές .
Αντίθετα, το *Streaming API* λειτουργεί διαφορετικά. Δημιουργεί αρχικά μια σύνδεση με το *Twitter* μέσω ενός *HTTP* αιτήματος (*HTTP request*) και στη συνέχεια δεν ξαναστέλνουμε κάτι σε αυτό. Το *Twitter* φροντίζει από μόνο του, να μας στείλει όσα τιτιβίσματα από το *Firehose* ικανοποιούν τα κριτήρια μας. Με αυτόν τον τρόπο καταφέρνουμε να τροφοδοτήσουμε την εφαρμογή μας με μεγάλο όγκο πραγματικού χρόνου δεδομένα, μιας και είμαστε απαλλαγμένοι από τους δύο βασικούς περιορισμούς των άλλων διεπαφών (δεδομένα παρελθόντος, *rate limits.* Έτσι, για την επικοινωνία μας με το *Twitter* επιλέχθηκε το *Streaming API* και πιο συγκεκριμένα το Filter Stream ,καθώς βασίζουμε την αναζήτηση σε μια λέξη-κλειδί.

5. Ανάλυση Δεδομένων με Γράφους

Έχοντας εξηγήσει πλέον τις τεχνολογίες που θα χρησιμοποιηθούν και τους λόγους για τους οποίους επιλέχθηκε κάθε μία από αυτές, θα περιγραφεί τώρα η μεθοδολογία που ακολουθείται για την αντιμετώπιση του προβλήματος. Η γενική λογική που οδήγησε στην συγκεκριμένη μέθοδο υπαγορεύεται από την ανάγκη δημιουργίας ενός συστήματος, μέσω του οποίου θα μπορεί οποιοσδήποτε χρήστης εύκολα να ερευνήσει για μια λέξη που επιθυμεί (είτε στο Twitter είτε στο σύνολο των αποθηκευμένων άρθρων) και να εξάγει αποτελέσματα από τις δοσμένες πληροφορίες (το γράφο). Στη συνέχεια, περιγράφονται αναλυτικά τα βήματα της εν λόγω μεθόδου.

5.1 Εξόρυξη/Συλλογή Δεδομένων

5.1.1 RSS Feeds

Θα ξεκινήσουμε περιγράφοντας τι είναι το RSS. Το RSS (Really Simple Syndication) αποτελεί το σύγχρονο τρόπο ανάγνωσης του διαδικτύου. Είναι μια τεχνολογία που επιτρέπει στους χρήστες να πάρουν και να διαβάσουν πληροφορίες που έχουν σταλεί σε αυτούς αντί να επισκεφτούν μόνοι τους τον αντίστοιχο ιστότοπο για να τις αναζητήσουν και να τις προσπελάσουν. Έτσι, ενημερώνονται αυτόματα για νέες καταχωρήσεις σε ιστοσελίδες, ιστολόγια, podcasts κλπ. Ένα κανάλι τροφοδοσίας RSS (RSS feed) αποτελείται από μία λίστα στοιχείων που περιέχουν ένα τίτλο, μια περιγραφή καθώς και το σύνδεσμο προς την αντίστοιχη ιστοσελίδα ή αρχείο. Τα RSS feeds προβάλλονται από προγράμματα ανάγνωσης ειδήσεων (news aggregators ή news readers ή feed readers), εφαρμογές οι οποίες ανακτούν και εμφανίζουν τα περιεχόμενα των RSS feeds που έχει επιλέξει ο χρήστης[16]. Όταν οι χρήστες επισκέπτονται τα προσωπικά τους feed readers βρίσκουν ενημερωμένο περιεχόμενο, δηλαδή μια συλλογή από feeds στα οποία έχουν γίνει συνδρομητές[17]. Έτσι υπάρχει η δυνατότητα λήψης κατευθείαν στον υπολογιστή και ανάγνωσης τίτλων και μιας μικρής περίληψης των θεμάτων ενδιαφέροντος, αμέσως μόλις αυτά γίνουν διαθέσιμα. Χρησιμοποιώντας τον ανάλογο σύνδεσμο που υπάρχει σε κάθε αντικείμενο ενός RSS feed μπορεί κάποιος να επισκεφτεί τη σχετική ιστοσελίδα και να έχει πρόσβαση σε ολόκληρο το περιεχόμενο της είδησης. Τεχνικά, το RSS είναι ένα πρωτόκολλο, ένα πρότυπο κειμένου συνήθως σε μια μορφή της γλώσσας XML, που επιτρέπει σε ιστοσελίδες να διαθέσουν είτε ολόκληρο είτε μέρος του περιεχομένου τους σε άλλες ιστοσελίδες ή εφαρμογές.

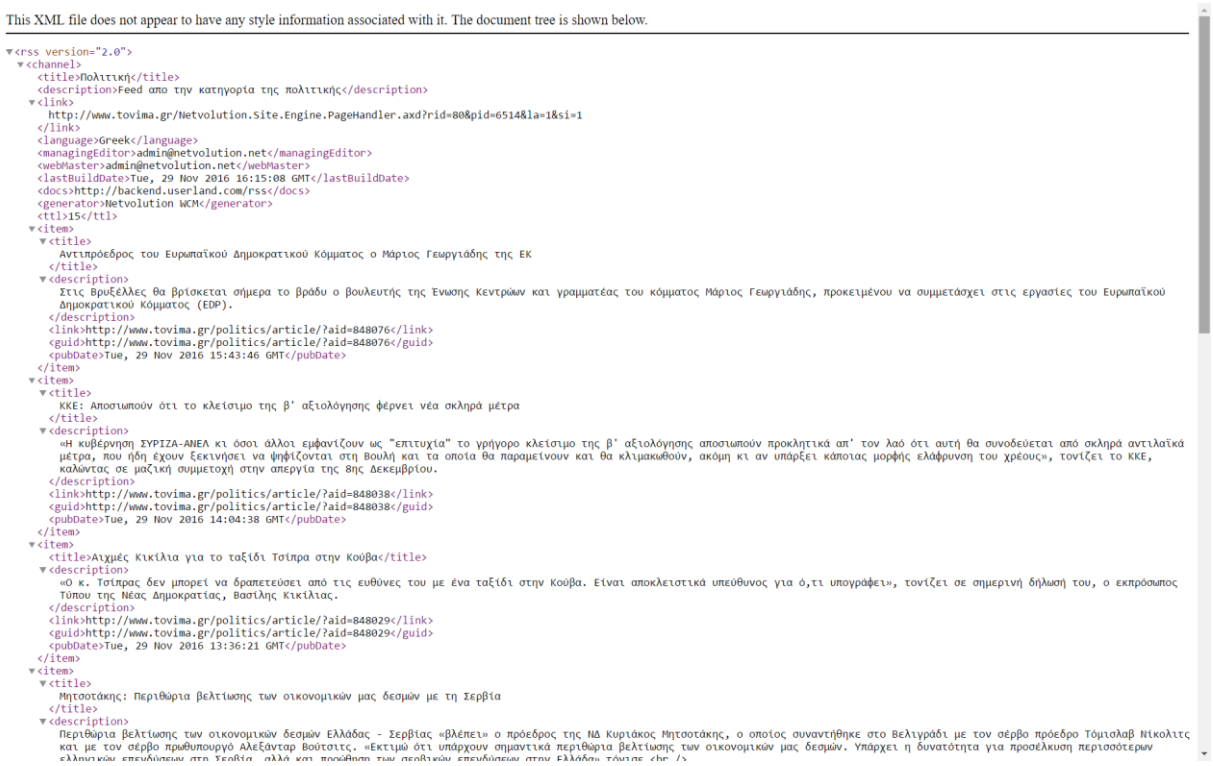
5.1.2 Απόσπαση Πληροφορίας από RSS

Με δεδομένα τα παραπάνω, ο σκοπός του πρώτου βήματος της εργασίας ήταν η δημιουργία μιας αυτοματοποιημένης διαδικασίας συλλογής δεδομένων, ενός RSS aggregator/scrapper. Τα δεδομένα που συλλέχθηκαν ,επιλέξαμε να είναι δημοσιογραφικού χαρακτήρα και για αυτό τα πήραμε από τα RSS Feeds των εφημερίδων:

- Καθημερινή
- Βήμα
- news247
- Ναυτεμπορική

στις κατηγορίες Πολιτική και Ελληνική Οικονομία.

Δημιουργήθηκε ένας συλλέκτης (scrapper),ο οποίος κάθε 90 λεπτά έπαιρνε τα άρθρα από αυτά τα RSS Feeds και ήταν σε συνεχή λειτουργία στο διάστημα 01/04/2016 – 01/06/2016 , με αποτέλεσμα , να έχουν συσσωρευτεί συνολικά, περίπου 6000 διαφορετικά άρθρα. Πιο αναλυτικά, η διαδικασία ξεκινάει κάνοντας synchronous request(σύγχρονη κλήση) στο κάθε RSS ξεχωριστά. Η απάντηση που λαμβάνεται , χρειάζεται να επεξεργαστεί με κατάλληλο τρόπο ,ώστε να εξαχθούν μόνο οι χρήσιμες πληροφορίες από τον όγκο των δεδομένων που επιστρέφει. Τα RSS έχουν μια συγκεκριμένη δομή (XML), στην οποία χρησιμοποιούνται tags (ετικέτες) ,που σηματοδοτούν την αρχή και το τέλος συγκεκριμένου κομματιού πληροφορίας , π.χ. <title> Τίτλος </title> . Έτσι ,θέλουμε να κρατηθεί μόνο ο «Τίτλος» και όχι οι ετικέτες που τον περιβάλλουν. Ένα παράδειγμα από RSS παρουσιάζεται στην παρακάτω εικόνα.



RSS Feed από την πολιτική στήλη της εφημερίδας Το Βήμα

Η ανάκτηση μόνο των χρήσιμων πληροφοριών, λοιπόν, επιτυγχάνεται με τη χρήση των Regular Expressions. Regular Expressions (κανονικές εκφράσεις) είναι μία ακολουθία χαρακτήρων, η οποία προσδιορίζει ένα μοτίβο έρευνας (search pattern) και πρακτικά χρησιμοποιείται για την εύρεση (ή αντικατάσταση) συγκεκριμένων συμβολοσειρών σε ένα κείμενο[18]. Εφαρμόζοντας λοιπόν, κανονικές εκφράσεις στην απάντηση από τα RSS ,

δίνεται η δυνατότητα να διαχωριστούν οι Τίτλοι, Περιγραφές και Ημερομηνίες του εκάστοτε κειμένου που εξετάζεται. Αφαιρούνται όλοι οι ειδικοί χαρακτήρες που μπορεί να αντιμετωπιστούν σε ένα XML και τελικά συλλέγονται μόνο τα επιθυμητά δεδομένα κάθε άρθρου.

5.2 Επικοινωνία με Firebase

5.2.1 FireBase

Αρχικά , μια σύντομη εισαγωγή για τη FireBase.
Η Firebase είναι μια διαδικτυακή real time NoSQL βάση δεδομένων της Google ,που φιλοξενείται στο Cloud και είναι σχεδιασμένη για την εξυπηρέτηση υψηλής ποιότητας εφαρμογών και εξοπλισμένη με κατάλληλα εργαλεία και δομές για να επιτύχει αυτή την ποιότητα. Η βάση είναι Real Time, διότι διαμοιράζεται στου clients, με αποτέλεσμα ό,τι αλλαγές συμβαίνουν, να διαδίδονται άμεσα στους υπόλοιπους clients. Η FireBase δίνει παράλληλα, τη δυνατότητα για cross-platform εφαρμογές με APIs φτιαγμένα για SDK σε iOS, Android , JavaScript και C++.[19]
Η επικοινωνία με τη FireBase στη συγκεκριμένη περίπτωση επιτυγχάνεται με χρήση Node.Js και JavaScript. Η FireBase στη διαδικασία χρησιμοποιείται μόνο σαν ένας χώρος αποθήκευσης των δεδομένων.

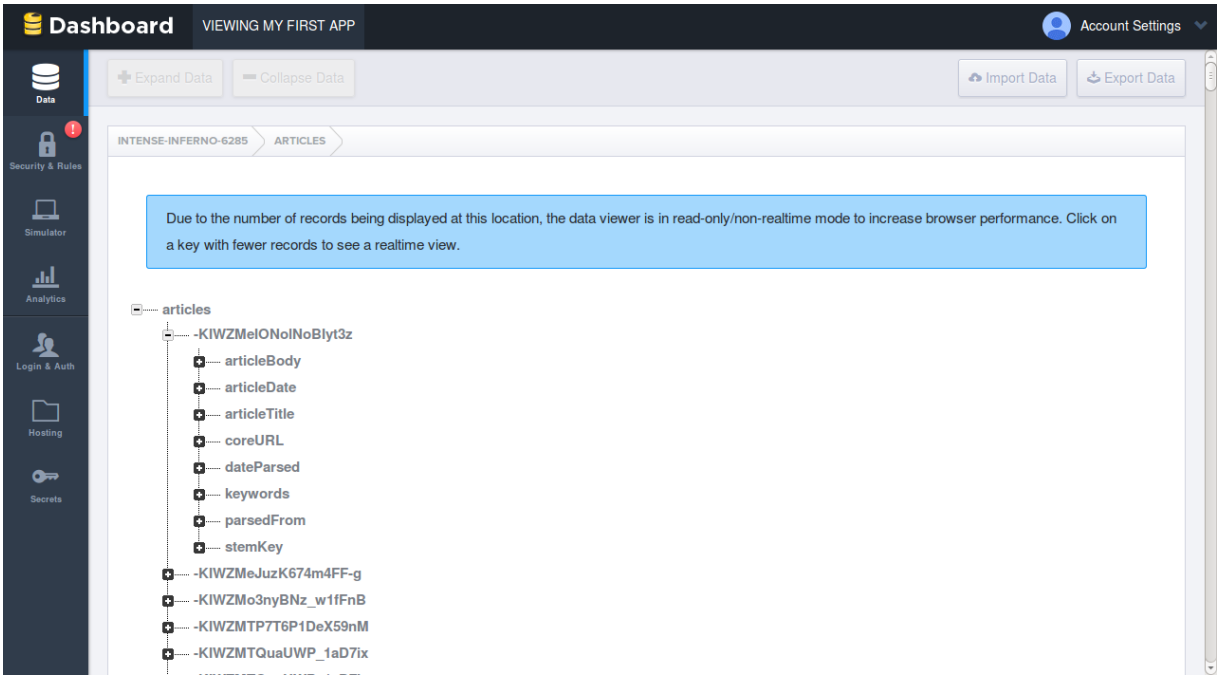
5.2.2 Αποθήκευση και έλεγχος δεδομένων

Όλα τα άρθρα που συλλέγονται, λοιπόν, αποθηκεύονται στη FireBase κάτω από την ετικέτα articles. Κάθε άρθρο στη βάση έχει ένα μοναδικό κλειδί, το οποίο του δίνεται από την ίδια τη FireBase, τη στιγμή που το εγγράφει στη βάση. Για κάθε άρθρο στη βάση υπάρχουν αποθηκευμένα τα παρακάτω στοιχεία του :

- Ο Τίτλος
- Η Περιγραφή
- Η Ημερομηνία Δημοσίευσης
- Η Ημερομηνία Συλλογής του
- Η Εφημερίδα
- Ο Σύνδεσμος (link)

Έτσι, ανά πάσα στιγμή, όλες οι πληροφορίες όλων των άρθρων είναι εύκολα προσβάσιμες και εκμεταλλεύσιμες, ανακτώντας αυτές από τη FireBase. Στην παρακάτω εικόνα φαίνεται ένα στιγμιότυπο από τη μορφή που παρουσιάζονται τα δεδομένα στη FireBase. Διακρίνεται η ετικέτα articles, κάτω από την οποία βρίσκονται όλα τα συλλεγμένα άρθρα και για κάθε

ένα από αυτά παρατηρείται το μοναδικό κλειδί του, το οποίο περιέχει με τη σειρά του, όλες τις προαναφερθείσες πληροφορίες. Μια παρατήρηση είναι ότι στην εικόνα φαίνονται και 2 παραπάνω πεδία, τα keywords (λέξεις κλειδιά) και stemKey (keywords μετά από stemming). Στα αρχικά στάδια της εφαρμογής η εξόρυξη των keywords γινόταν κατά τη συλλογή των κειμένων , αλλά μετά αποφασίσουμε να διαφοροποιήσουμε αυτές τις 2 λειτουργίες και η εξόρυξη λέξεων-κλειδιών γίνεται μετέπειτα ,όπως περιγράφεται στην επόμενη ενότητα. Έτσι, μόνο μερικές από τις πρώτες εγγραφές στη FireBase έχουν αυτά τα επιπλέον πεδία, όλες οι υπόλοιπες έχουν μόνο αυτά που αναφέρονται στα παραπάνω bullets.



FireBase/articles

Στην διαδικασία που περιγράφεται προηγουμένως, πριν το ανέβασμα των νέων άρθρων στη FireBase γίνεται έλεγχος μοναδικότητας. Η ύπαρξη του ίδιου άρθρου παραπάνω από μία φορά στη βάση θα οδηγούσε σε λάθος συμπεράσματα στα επόμενα βήματα της εργασίας , όποτε γίνεται έλεγχος πριν ανεβεί οποιοδήποτε άρθρο αν υπάρχει ήδη στη βάση. Αυτό γίνεται με έλεγχο του Τίτλου ενός νέου άρθρου ,αν υπάρχει ήδη στη FireBase και αν υπάρχει αγνοείται ,διαφορετικά αποθηκεύεται στη βάση. Έχοντας πλέον όλα τα άρθρα στη βάση, υπάρχει η δυνατότητα αναπαραγωγής των διαδικασιών για την επίτευξη του τελικού αποτελέσματος , διαχωρίζοντας τα δεδομένα από τη διαδικασία.

5.3 Λέξεις-Κλειδιά

5.3.1 Αυτόματη ανακεφαλαιοποίηση

Η αυτόματη ανακεφαλαιοποίηση είναι η διαδικασία της δημιουργίας της περίληψης ενός κειμένου, η οποία διατηρεί όλες τις σημαντικές πληροφορίες του κειμένου αυτού και έχει πολύ μικρότερη έκταση από το αρχικό. Η ανακεφαλαιοποίηση παίζει πολύ σημαντικό ρόλο σήμερα όπου ο όγκος των δεδομένων έχει αυξηθεί σημαντικά αφού δεν είναι δυνατή πλέον η προσπέλαση όλων των δεδομένων σε ικανοποιητικό χρόνο. Έτσι χρειαζόμαστε μια περίληψη των δεδομένων διατηρώντας όμως όλες τις σημαντικές πληροφορίες. Υπάρχουν δύο σημαντικές τεχνικές για την ανακεφαλαιοποίηση: η εξόρυξη και η άντληση. Κατά την εξόρυξη επιλέγεται ένα μέρος από τις λέξεις, φράσεις ή και προτάσεις του αρχικού κειμένου και σχηματίζεται η περίληψη, ενώ αντίθετα κατά την άντληση δημιουργείται μια εσωτερική σημασιολογική αναπαράσταση του κειμένου και μετά χρησιμοποιούνται τεχνικές παραγωγής φυσικής γλώσσας έτσι ώστε να δημιουργηθεί η περίληψη η οποία είναι πιο κοντά στην περίληψη που θα δημιουργούσε ένας άνθρωπος. Κατά την τεχνική αυτή μπορεί να χρησιμοποιηθούν λέξεις και φράσεις οι οποίες δεν αναφέρονται στο αρχικό κείμενο. [20]

5.3.2 Εξόρυξη Λέξεων-Κλειδιών

Έτσι , από τις 2 μεθόδους που αναφέρονται στην προηγούμενη υποενότητα , επιλέχθηκε η εξόρυξη των λέξεων κλειδιών , που συνοψίζουν το αντίστοιχο άρθρο.
Η διαδικασία που ακολουθείται ξεκινάει κατεβάζοντας όσα άρθρα έχουν αποθηκευτεί μέχρι στιγμής, στη Firebase και στη συνέχεια προσαρμόζοντας τα δεδομένα στη μορφή που θέλουμε, για να τα περάσουμε στο Neo4j (ώστε να τα έχουμε σε ένα γράφο).Τα δεδομένα που χρειαζόμαστε από κάθε άρθρο για τη δημιουργία του γράφου είναι τα εξής :

- Οι Λέξεις-Κλειδιά κάθε άρθρου
- Ο Τίτλος ,για να φαίνεται κάθε Λέξη-Κλειδί από ποιο άρθρο προέρχεται
- Το core URL (π.χ. tovima.gr),το οποίο πρακτικά δείχνει κάθε άρθρο από ποια εφημερίδα είναι

Τώρα, θα αναλύσουμε την έννοια που έχουν οι Λέξεις-Κλειδιά για την εφαρμογή και πως προκύπτουν αυτές. Όπως είπαμε, θέλουμε οι λέξεις-κλειδιά που θα πάρουμε να περιέχουν όση περισσότερη πληροφορία γίνεται για το άρθρο που αναφέρονται και επειδή θα αντιστοιχηθούν με κόμβους στη συνέχεια, θέλουμε όσο το δυνατόν λιγότερες λέξεις για να μειωθεί και το μέγεθος του τελικού γράφου. Καταρχάς, υπάρχει μία λίστα από λέξεις για κάθε γλώσσα , η οποία περιέχει όλες τις λέξεις που δεν προσφέρουν χρήσιμη πληροφορία στη γλωσσική ανάλυση ενός κειμένου στο οποίο αυτές περιέχονται (π.χ. τα άρθρα) [21]. Οι λέξεις αυτές λέγονται **stop words** και το πρώτο βήμα είναι να τις αφαιρέσουμε. Τώρα, υπάρχουν δύο τρόποι με τους οποίους μπορούμε να καταλήξουμε στις λέξεις-κλειδιά. Ο πρώτος είναι να θεωρήσουμε, ότι η λογική μιας επιτυχούς ανακεφαλαιοποίησης έχει ακολουθηθεί από τους αρθρογράφους κατά τη δημιουργία των τίτλων τους και άρα να πάρουμε σαν λέξεις-κλειδιά , τις λέξεις του τίτλου (χωρίς τις stop words). Η δεύτερη

μέθοδος είναι να χρησιμοποιήσουμε τον αλγόριθμο **textrank** στην περιγραφή που παίρνουμε από κάθε άρθρο.[22]
Κατά τη συλλογή των λέξεων-κλειδιών από τα άρθρα χρησιμοποιήσαμε μία εκδοχή του textrank, σχεδιασμένο για την ελληνική γλώσσα, αλλά στην τελική εκδοχή της εφαρμογής, τόσο για τα άρθρα όσο και για τα tweets κρατάμε απλά τις λέξεις-κλειδιά του τίτλου/tweet, ώστε να είναι δυνατή η επεξεργασία ξενόγλωσσων πηγών.
Και στις 2 περιπτώσεις , πριν την εξόρυξη των λέξεων πρέπει να έχουμε ορίσει σε ποια γλώσσα θα είναι τα κείμενα ,ώστε να έχουμε επιτυχή αναγνώριση των stop words και των λέξεων, εξαιτίας ειδικών χαρακτήρων που μπορεί να εμφανίζονται στην κάθε γλώσσα .
Τέλος, στις λέξεις-κλειδιά που προκύπτουν εφαρμόζεται μια διαδικασία που λέγεται stemming. Με το stemming αφαιρούνται από τις λέξεις οι καταλήξεις τους, με αποτέλεσμα από κάθε λέξη να βρίσκουμε τη ρίζα της. Αυτό το κάνουμε γιατί θέλουμε να ενώσουμε τους κόμβους από λέξεις με ίδιες ρίζες, πχ. χωρίς stemming οι λέξεις : χρηματιστήριο , χρηματιστηρίου , χρηματιστήριων θα δημιουργούσαν 3 διαφορετικούς κόμβους, ενώ μετά το stemming δημιουργείται ένας κοινός κόμβος και για τις 3 το χρηματιστήριο.[23]
Για τον ίδιο λόγο αφαιρούμε και τους τόνους από όλες τις ελληνικές λέξεις .

5.4 Twitter

Έχουμε, μέχρι στιγμής, μαζέψει δεδομένα από τα RSS των εφημερίδων ,τα έχουμε αποθηκεύσει στη FireBase και τα έχουμε επεξεργαστεί κατάλληλα, ώστε να δημιουργήσουμε ένα γράφο από το σύνολο των δεδομένων, όπως έχει ήδη περιγραφεί προηγουμένως. Τώρα θέλουμε να ελέγξουμε, αν το παραπάνω σύστημα, μπορεί να εφαρμοστεί και να δώσει αποτελέσματα και με διαφορετικά δεδομένα. Έτσι, επιλέξαμε να δοκιμάσουμε το μέσο κοινωνικής δικτύωσης Twitter, από το οποίο μπορούμε να συλλέξουμε μεγάλο όγκο δεδομένων πολύ γρήγορα και που διαφέρουν από το δημοσιογραφικό χαρακτήρα που είχαν τα υπόλοιπα δεδομένα μας. Τα δεδομένα που προέρχονται από το Twitter, εμφανίζουν πολύ μεγάλη ποικιλία τόσο στο ύφος όσο και στο περιεχόμενο. Με αυτό τον τρόπο, εξετάζεται η αποτελεσματικότητα της μεθόδου σε ένα διαφορετικό περιβάλλον από το προηγούμενο.
Για το Twitter χρησιμοποιούνται ακριβώς τα ίδια εργαλεία και μεθοδολογία με τα άρθρα από τα RSS με τη διαφορά ότι τώρα, αντί να έχουμε ένα RSS scrapper, έχουμε ένα Twitter streamer. Όπως αναφέραμε και στην τέταρτη ενότητα, το Stream δίνει live tweets που γράφονται για το θέμα που ακολουθείς, σε αντίθεση με το Search που δίνει τα 100 πιο πρόσφατα ή δημοφιλή (αναλόγως τη ρύθμιση) tweets, με μέγιστο βάθος χρόνου μίας εβδομάδας. Εμείς επιλέξαμε το Stream API, διότι θέλαμε περισσότερα δεδομένα από αυτά που μπορεί να προσφέρει το Search, καθώς τα tweets του Search δεν είναι αρκετά για τη αξιόπιστη άντληση συμπερασμάτων από το γράφο.
Για όσο χρόνο λειτουργεί ο Twitter streamer και παίρνουμε tweets, για κάθε ένα από αυτά κρατάμε μόνο τα text τους (το ίδιο το tweet), αφαιρώντας τα # και τα links. Το Twitter, επίσης, κρατάει για κάθε tweet τον αριθμό , με τον οποίο έχει επανακοινοποιηθεί το ίδιο tweet,έχει γίνει δηλαδή retweet από άλλου χρήστες. Εμείς, μπορούμε είτε να συμπεριλάβουμε αυτά τα retweets (πχ. προσθέτοντας στο βάρος των ακμών, τον αριθμό που έχει γίνει κάτι retweet) είτε να τα αγνοήσουμε πλήρως, δηλαδή να μην συμπεριλαμβάνεται κανένα retweet. Επιλέχθηκε το δεύτερο, καθώς θέλαμε να εστιάσουμε στη νέα πληροφορία που μπορεί να προσφέρει κάθε χρήστης και όχι στην κοινωνική τάση και προδιάθεση που υποδεικνύουν τα retweets (αναλύοντας από που προέρχονται και για ποιο λόγο έγιναν).

Το μόνο πρόβλημα που παρουσιάζεται κατά τη συλλογή των tweets, είναι η ύπαρξη spam μηνυμάτων. Spam μηνύματα είναι αυτοματοποιημένα μαζικά μηνύματα που στέλνονται συνεχώς σε ένα αποδέκτη με διαφημιστικό ή κακόβουλο σκοπό. Προφανώς, οι πληροφορίες που συλλέγονται από αυτά τα μηνύματα είναι ανεπιθύμητες και οδηγούν το γράφο σε λανθασμένα κέντρα βάρους, γύρω από αυτά. Το Twitter δεν έχει κάποια μέθοδο που να τα διαχωρίζει και δεν αποτελούν retweets,ώστε να μειώσουμε τουλάχιστον την εμφάνιση τους στη μία φορά. Ένας τρόπος αντιμετώπισής τους, είναι να κάνουμε έλεγχο μοναδικότητας στα tweets που συλλέγουμε, με βάση το κείμενο του tweet, καθώς τα spam messages περιέχουν ακριβώς το ίδιο κείμενο και να μειώσουμε την εμφάνιση τους.
Έχοντας συλλέξει πλέον τα tweets, στη συνέχεια τα ανεβάζουμε στη Firebase, αυτή τη φορά κάτω από την ετικέτα twitter, αλλά με αντίστοιχη δομή όπως και πριν και ακολουθείται η ίδια διαδικασία που περιγράφεται στο 5.3.

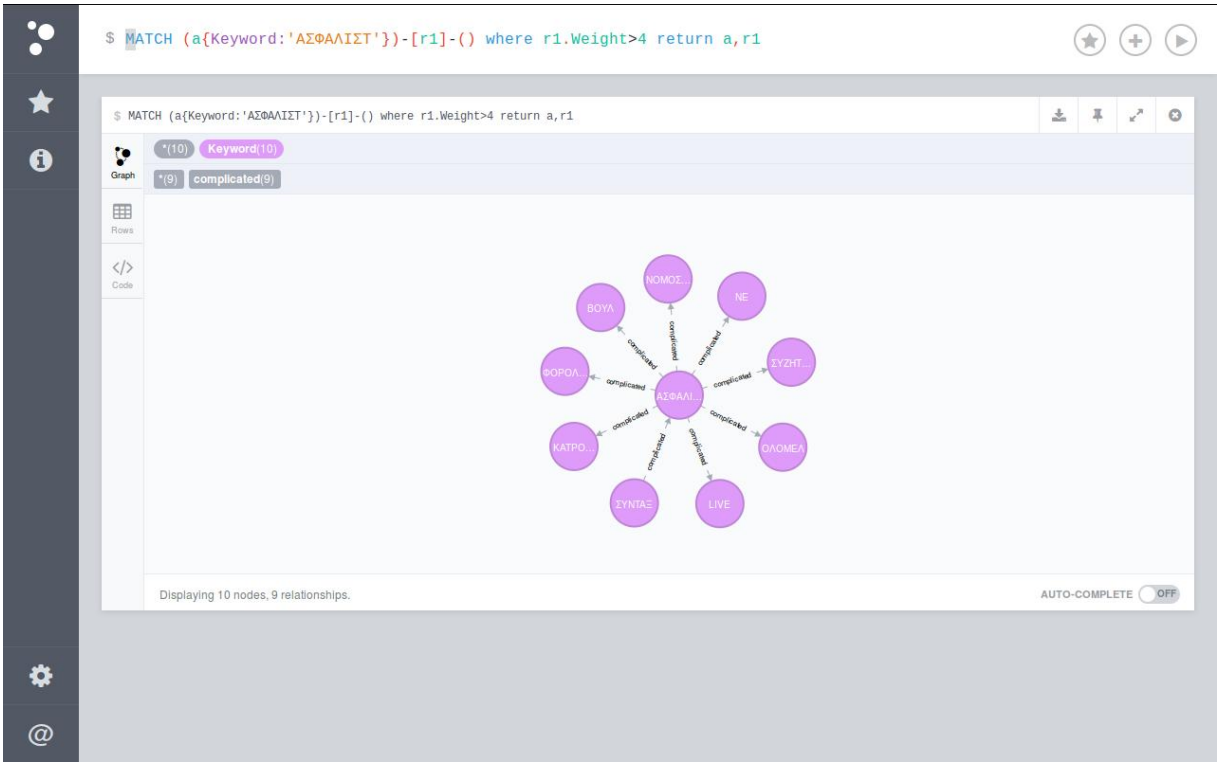
5.5 Neo4j

Έπειτα ,προχωράμε στην δημιουργία του γράφου, στη γραφική βάση δεδομένων Neo4j. Ξεκινάμε με την προσθήκη των keywords(λέξεις-κλειδιά) σαν κόμβους στο Neo4j. Η λογική που ακολουθείται είναι η εξής:
Αρχικά για κάθε άρθρο που έχουμε πάρει ,ελέγχουμε τα keywords του και για κάθε keyword εξετάζουμε, αν έχει ήδη προστεθεί σαν κόμβος στο Neo4j. Αυτό γίνεται με χρήση Cypher Query πάνω στη Neo4j ,ζητώντας το keyword που θέλουμε να προσθέσουμε. Αν υπάρχει στο γράφο ένας κόμβος με λέξη-κλειδί , τη λέξη που ερευνάται, τότε επιστρέφει το Query σαν απάντηση, το id του κόμβου που ταυτίζεται με αυτή τη λέξη-κλειδί και εμείς κρατάμε το id αυτό, ώστε πρακτικά να ενώσουμε τη λέξη-κλειδί, με τον παλιό κόμβο που έχει ήδη δημιουργηθεί. Αντίθετα, αν δεν επιστρέψει απάντηση το Query, προστίθεται στη συνέχεια, ένας νέος κόμβος στο Neo4j και του προσδίδεται ένα μοναδικό id. Αυτή η διαδικασία επαναλαμβάνεται για όλα τα άρθρα (και για όλα τα keywords αυτών). Όταν τελειώσει η διαδικασία στο Neo4j υπάρχουν όσοι κόμβοι, όσες και διαφορετικές λέξεις-κλειδιά και το καθένα έχει ένα ξεχωριστό id. Τώρα, για κάθε ζευγάρι keywords/κόμβων που υπάρχουν στο ίδιο άρθρο/tweet , θέλουμε στη συνέχεια να φτιάξουμε και μία σχέση, που στο γράφο θα αναπαρίσταται ως ακμή. Έτσι, παίρνουμε σε ζεύγη τα ids των κόμβων/keywords κάθε άρθρου και συνεχίζουμε ελέγχοντας αν υπάρχει ακμή που να ενώνει αυτά τα 2 ids (πάλι με Cypher query) . Αν δεν υπάρχει, τότε προστίθεται μία ακμή που να τα ενώνει και σε κάθε νέα ακμή προσδίδονται 3 χαρακτηριστικά :

- Βάρος
- Πίνακας από Τίτλους
- Πίνακας από Πηγές

Ο πίνακας από Τίτλους περιέχει τους τίτλους όλων των κειμένων στα οποία έχουν βρεθεί μαζί οι 2 λέξεις/κόμβοι, που ενώνει η ακμή. Αντίστοιχη πληροφορία περιέχει και ο πίνακας των πηγών απλά δείχνει από ποιες πηγές ήρθαν τα άρθρα (μία από τις εφημερίδες ή από το Twitter). Το βάρος είναι η πιο σημαντική πληροφορία για την εν λόγω εφαρμογή. Με το βάρος επισημαίνεται πόσες φορές έχουν βρεθεί οι 2 λέξεις/κόμβοι μαζί στο ίδιο κείμενο. Έτσι, με την προσθήκη μιας νέας ακμής ορίζουμε βάρος ίσο με 1. Όταν συναντάμε όμως, δύο κόμβους που ήδη είναι ενωμένοι με μία ακμή, αυξάνουμε το βάρος της ακμής κατά 1 και προσθέτουμε τα κατάλληλα στοιχεία στον πίνακα Τίτλων και Πηγών.
Όταν τελειώσει όλη η διαδικασία , αυτό που έχει δημιουργηθεί μέσα στη Neo4j είναι ένας γράφος, ο οποίος αποτελείται από κόμβους/keywords, που είναι συνδεδεμένοι μεταξύ τους, αν έχουν βρεθεί έστω και μία φορά στο ίδιο άρθρο. Έτσι, κοιτώντας τους κόμβους που έχουν ακμές με μεγάλα βάρη, βρίσκουμε τα νοηματικά κέντρα βάρους του γράφου, αφού γύρω από αυτές τις λέξεις/κλειδιά έχει ασχοληθεί το μεγαλύτερο πλήθος της αρθρογραφίας (ή του Twitter). Μία άλλη σημαντική σημείωση στη δομή του γράφου που δημιουργείται από το Neo4j είναι ότι είναι ενιαίος. Αυτό σημαίνει ότι όσο συνεχίζουμε να προσθέτουμε δεδομένα , δεν δημιουργούμε νέους υπο-γράφους, αλλά μεγαλώνουμε τον αρχικό γράφο. Αυτό σημαίνει με τη σειρά του ότι κάθε Query γίνεται σε όλο το γράφο ανεξαρτήτως από που προέρχονται τα δεδομένα (άρθρα ή Twitter). Επειδή, όμως τα δεδομένα που λαμβάνουμε ,μπορεί να προέρχονται από δύο πηγές το Twitter και τα RSS Feeds δημιουργούνται δύο προβλήματα. Το πρώτο είναι ότι αυξάνεται ο χρόνος των queries, τα οποία ελέγχουν στο σύνολο του γράφου, ενώ πχ. να θέλαμε μόνο να ελέγξουμε ένα κομμάτι του γράφου του Twitter. Το δεύτερο είναι ότι, παρόλο που τα δεδομένα και οι λέξεις κλειδιά

που παίρνουμε από το Twitter είναι στα Αγγλικά, ενώ των RSS Feeds είναι κυρίως στα Ελληνικά, μπορεί για κάποια λέξη-κλειδί να ενωθούν 2 κόμβοι μεταξύ τους και να δημιουργηθεί μεγάλη σύγχυση στα αποτελέσματα. Η λύση σε αυτό είναι, από τη στιγμή που έχουμε τα δεδομένα των άρθρων αποθηκευμένα στη FireBase, πριν τη δημιουργία του γράφου από το Twitter να διαγράφουμε στη Neo4j το γράφο από τα RSS, ώστε να έχουμε μόνο στο γράφο δεδομένα από το Twitter. Προφανώς, θα κάνουμε το αντίστοιχο όταν θέλουμε να κατασκευάσουμε το γράφο από τα άρθρα, εκτός και αν επιθυμούμε συνειδητά να ενωθούν τα δεδομένα και από τις 2 πηγές. Για να δούμε όμως όλα αυτά τα αποτελέσματα χρειαζόμαστε ένα εργαλείο οπτικοποίησης.
Το Neo4j μας δίνει τη δυνατότητα για visualization του γράφου μέσα από το Neo4j/browser, το οποίο και φαίνεται παρακάτω. Σε αυτό υπάρχει ένα παράθυρο εντολών όπως φαίνεται στο πάνω μέρος της εικόνας, όπου ο χρήστης εισάγει Queries στη γλώσσα Cypher και του επιστρέφεται ο γράφος σαν απάντηση. Οι δυνατότητες, όμως που δίνει ο default Neo4j browser είναι περιορισμένες και επειδή θέλαμε να διευκολύνουμε τη διαχείρηση της εφαρμογής από ένα χρήστη και να του επιτρέψουμε να βλέπει τους γράφους που επιθυμεί, χωρίς να χρειάζεται να κάνει Cypher Queries και άρα να γνωρίζει τη γλώσσα αυτή. Αυτά μας οδήγησαν λοιπόν στη δημιουργία ενός User Interface όπως περιγράφεται στη συνέχεια.



Neo4j Browser

5.6 Αναπαράσταση Γράφων

5.6.1 Εργαλεία

Όταν τελειώσει η δημιουργία του γράφου έχουμε τη δυνατότητα να δούμε το γράφο με 2 εργαλεία. Το πρώτο είναι το Neo4j browser(εργαλείο που μας δίνεται από το ίδιο το Neo4j) που συνδέεται αυτόματα στη βάση μας και εμφανίζει ,με βάση κάποιο Cypher Query, τον υπο-γράφο που επιθυμούμε να δούμε. Ο δεύτερος τρόπος και ο τρόπος που χρησιμοποιείται για την επίδειξη της συγκεκριμένης εφαρμογής ,γίνεται μέσω ενός διαδικτυακού γραφικού περιβάλλοντος.
Οι γλώσσες που χρησιμοποιήθηκαν για την οπτική διεπαφή με τον χρήστη είναι οι ακόλουθες :

• ***HTML*** : Η *HTML* (ακρωνύμιο του *HyperText Markup Language* ή Γλώσσα Σήμανσης Υπερκειμένου) είναι μια γλώσσα που χρησιμοποιείται για την περιγραφή της δομής των ιστοσελίδων. Η περιγραφή αυτή της δομής, γίνεται με χρήση σήμανσης (*markup*). Ο κύριος τρόπος που επιτυγχάνεται η σήμανση στην *HTML* είναι τα στοιχεία *HTML* (*HTML elements*). Τα στοιχεία *HTML* αποτελούνται από τις ετικέτες (*tags*) εκκίνησης και τερματισμού· οι λέξεις ανάμεσα στις γωνιώδης παρενθέσεις (*angle-brackets* ή "< >"), τις ιδιότητες μέσα στην ετικέτα εκκίνησης και το κείμενο ή γραφικό περιεχόμενο μεταξύ των δύο ετικετών. Οι ετικέτες είναι λοιπόν οι κρυφές λέξεις (δεν εμφανίζονται στον χρήστη), που ορίζουν πως θα σχηματιστεί και θα εμφανιστεί το περιεχόμενο των σελίδων. Οι φυλλομετρητές (*browsers*) μπορούν να διαβάσουν αρχεία αυτής της γλώσσας (έχουν κατάληξη "*.html"*), και να τα καταστήσουν οπτικά ή ακουστικά προσιτά στον χρήστη. [24]

• ***CSS*** : Η *CSS* (ακρωνύμιο του *Cascading Style Sheets* ή Διαδοχικά Φύλλα Στυλ) είναι μια γλώσσα που χρησιμοποιείται για την περιγραφή της παρουσίασης των ιστοσελίδων. Ουσιαστικά ελέγχει την εμφάνιση ενός εγγράφου που έχει γραφτεί σε μια γλώσσα σήμανσης (όπως η *HTML*), μέσω της στιλιστικής του ανάπτυξης, επεμβαίνοντας στο χρώμα, στο σχέδιο και τις γραμματοσειρές του. Επίσης, μας προσφέρει την δυνατότητα να προσαρμόσουμε το παρουσιαστικό μιας ιστοσελίδας στους διάφορους τύπους συσκευών όπως για παράδειγμα μεγάλες ή μικρές οθόνες και εκτυπωτές. Ο διαχωρισμός της *CSS* από την *HTML* μας διευκολύνει αφενός στην συντήρηση των ιστοσελίδων και αφετέρου στο να είναι προσιτό το ίδιο περιεχόμενο σε διαφορετικά περιβάλλοντα.[25]

• ***JavaScript*** : Για τη JavaScipt έχει ήδη γίνει εκτεταμένη περιγραφή στην 3^{η} Ενότητα.

• ***JQuery*** : Η *JQuery* είναι η δημοφιλέστερη βιβλιοθήκη της *JavaScript*, φτιαγμένη για να λειτουργεί σε όλους τους γνωστούς φυλλομετρητές (*browsers*). Σχεδιάστηκε για να απλοποιήσει την υλοποίηση σεναρίων στην πλευρά του πελάτη. Παρέχει μεθόδους στους προγραμματιστές ώστε να κάνει ευκολότερη και γρηγορότερη την δημιουργία ενός δυναμικού περιβάλλοντος καθώς και την επικοινωνία μεταξύ εξυπηρετούμενου – εξυπηρετητή (*client - server*).[26]

Όλες οι παραπάνω τεχνολογίες, τρέχουν κανονικά σε όλους τους γνωστούς σύγχρονους φυλλομετρητές. Αρκεί λοιπόν ο χρήστης να ανοίξει κάποιον από αυτούς όπως ο *Google Chrome* ή ο *Mozilla Firefox*, για να μπορέσει να τρέξει την εφαρμογή. Για την εργασία αυτή επιλέχθηκε ο *Mozilla Firefox*.

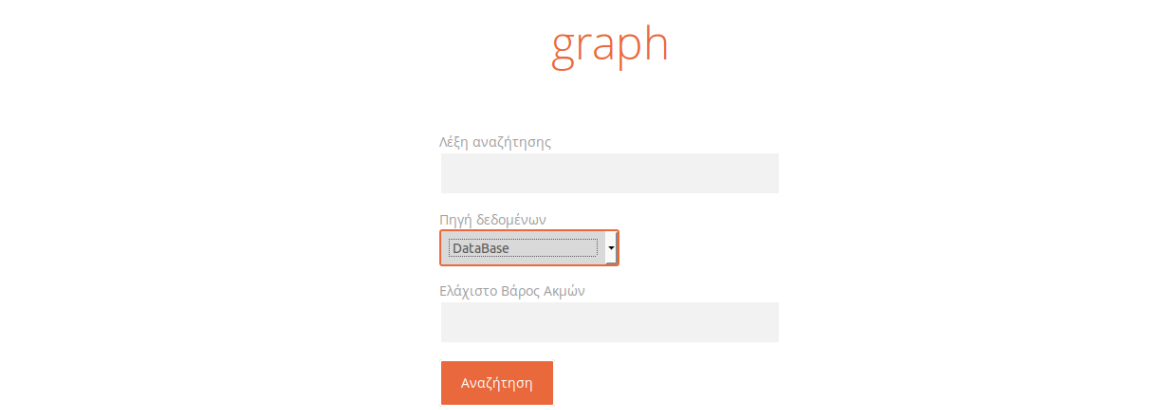
Επίσης, χρησιμοποιείται η βιβλιοθήκη ***Sigma.js,*** η οποία είναι μια open source βιβλιοθήκη *JavaScript* που μας επιτρέπει να σχεδιάσουμε γράφους σε ιστοσελίδες και άρα κατάλληλη για τη συγκεκριμένη εφαρμογή. Η βιβλιοθήκη αυτή έχει σχεδιαστεί κυρίως για οπτικοποίηση διαδραστικά στατικών γράφων, με βάση δεδομένα που έχουν εξαχθεί από μια βάση δεδομένων, όπως η Neo4j, αλλά και για οπτικοποίησης δυναμικών γράφων που παράγονται καθώς ο γράφος κινείται.[27] Είναι εύκολη στην χρήση και υψηλά παραμετροποιήσιμη. Παρόλο που υπάρχουν διάφορες αντίστοιχες βιβλιοθήκες, όπως οι D3.Js[28] και Linkurious [29], επιλέχθηκε η Sigma.Js διότι πρόσφερε τα περισσότερα παραδείγματα και δυνατότητες , σε συνδυασμό με μεγάλη ευκολία εκμάθησης.
Πιο συγκεκριμένα, χρησιμοποιήσαμε ένα plugin που διαθέτει η sigma.js, το οποίο επικοινωνεί απευθείας με τη Neo4j και δημιουργεί το γράφο με βάση κάποιο query , αντίστοιχο στη λογική με αυτό που χρησιμοποιούμε στο Neo4j Browser. Έχει χρησιμοποιηθεί ένα ακόμα plugin,το οποίο πατώντας με το ποντίκι πάνω σε ένα κόμβο εμφανίζει μόνο τους άμεσα γειτονικούς κόμβους (αυτούς που ενώνονται απευθειάς με μια ακμή με τον επιλεγμένο κόμβο). Τα screenshots που παρατίθενται παρακάτω είναι από το sigma.js.

5.6.2 Πλευρά του Server

Όπως αναφέραμε, το *Node* μας επιτρέπει να γράψουμε *JavaScript* στην πλευρά του εξυπηρετητή (*server-side JavaScript*). Στα πλαίσια της εφαρμογής αυτής, για την περίπτωση αναζήτησης λέξεων στο Twitter, θα δημιουργήσουμε έναν *HTTP* εξυπηρετητή. Για τη δημιουργία του χρησιμοποιείται το, ήδη, ενσωματωμένο πακέτο 'http' του Node.Js. Αυτό το πακέτο μας παρέχει μια διεπαφή για λειτουργίες *HTTP* τόσο στον εξυπηρετητή και όσο στον εξυπηρετούμενο. Ο *HTTP* εξυπηρετητής που δημιουργούμε, εξυπηρετεί κάθε εισερχόμενο *HTTP* αίτημα (*HTTP request*).Ο server μας λοιπόν ξεκινάει και ακούει σε μία τοπική θύρα TCP/IP και περιμένει HTTP requests. Όταν ο χρήστης δώσει τα στοιχεία στη σελίδα της διεπαφής (όπως αυτή περιγράφεται στην ενότητα 5.6.3) και πατήσει το κουμπί της αναζήτησης, δημιουργείται μια κλήση στην τοπική θύρα που ακούει ο server και περνάνε παράλληλα τα δεδομένα που έδωσε ο χρήστης. Τώρα που ο server έχει λάβει το request από την εφαρμογή, πρέπει να επεξεργαστεί τα δεδομένα που του δίνονται. Καταρχάς τα δεδομένα που λαμβάνει ο server από το HTTP request πρέπει να τα μετατρέψει από δυαδικά σε string, ώστε να είναι δυνατή η περαιτέρω επεξεργασία τους. Έχοντας κωδικοποιήσει σωστά (utf-8) ο server τα δεδομένα ,με χρήση regular expressions βρίσκει τη λέξη κλειδί και το χρόνο αναζήτησης του Twitter. Στη συνέχεια, ο server δημιουργεί ένα Twitter stream (όπως περιγράφεται στην ενότητα5.4) και όταν περάσει ο χρόνος αναζήτησης προσθέτει στο γράφο του Neo4j τα νέα δεδομένα. Όταν τελειώσουν όλα αυτά επιστρέφει την απάντηση στην εφαρμογή και αυτή εμφανίζει το γράφο που επιθυμεί ο χρήστης. Όπως είναι κατανοητό όλη αυτή η διαδικασία μπορεί να πάρει αρκετό χρόνο για να ολοκληρωθεί, αλλά ο server στο πέρασμα των 2 λεπτών θα δεχθεί, δεύτερο request, καθώς θεωρείται ότι το πρώτο έκανε timeout. Αυτό οδηγεί στην επανεκκίνηση της όλης διαδικασίας που προφανώς θα δημιουργήσει και λάθος αποτελέσματα. Έτσι, έπρεπε είτε να θέσουμε το χρόνο για τον οποίο κάνει το socket timeout, σε ένα αυθαίρετο πολύ μεγάλο χρόνο είτε να αφαιρέσουμε τελείως τον Listener για το Timeout, η οποία είναι και η λύση που επιλέξαμε.

5.6.3 UI (User Interface)

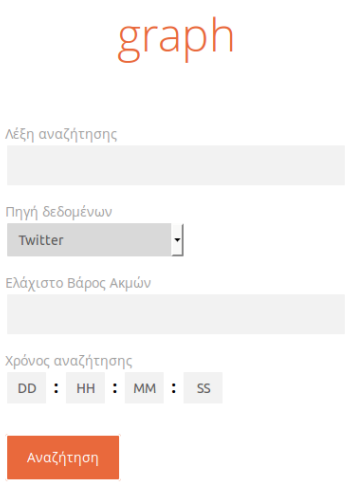
Το γραφικό περιβάλλον της εφαρμογής δημιουργήθηκε με σκοπό να δώσουμε τη δυνατότητα σε οποιοδήποτε χρήστη, να χρησιμοποιεί την εφαρμογή, χωρίς να χρειάζεται να γνωρίζει τη γλώσσα ερωτήσεων Cypher. Παράλληλα, ο χρήστης δεν χρειάζεται να ασχοληθεί καθόλου με τον κώδικα, για να τρέξει το κομμάτι που αφορά το Twitter. Η δομή και η μορφή των σελίδων είναι πολύ απλή και κατανοητή, για την άμεση παρουσίαση των αποτελεσμάτων. Καταρχάς, πρέπει να σημειωθεί ότι τη στιγμή που χειρίζεται για πρώτη φορά ο χρήστης τη διεπαφή μας, στο Neo4j είναι ήδη φτιαγμένος ο γράφος από τα δεδομένα των RSS και ότι δεν έχουν συλλεχθεί καθόλου δεδομένα από το Twitter. Έτσι, οι επιλογές που του δίνονται σε αυτό το σημείο είναι 2.
Πρώτη επιλογή είναι να ελεγχθεί ο γράφος στη Neo4j, που έχει δημιουργηθεί από τα RSS, για μια λέξη που θα εισάγει ο χρήστης. Αυτό το επιλέγει ο χρήστης από την επιλογή του drop down κουτιού «Database». Ο χρήστης εισάγει τη λέξη που επιθυμεί στο πεδίο «Λέξη Αναζήτησης» και το κατώφλι του βάρους των ακμών που ενώνονται με αυτό τον κόμβο/λέξη στο πεδίο «Ελάχιστο Βάρος Ακμών». Με το πάτημα του κομβίου «Αναζήτηση» γίνεται έλεγχος των δύο πεδίων που έχει συμπληρώσει ο χρήστης, ώστε να έχει δώσει μία οποιαδήποτε συμβολοσειρά στο πρώτο και ένα θετικό ακέραιο αριθμό στο δεύτερο πεδίο. Στην περίπτωση που δεν πληρούνται οι παραπάνω συνθήκες εμφανίζεται αντίστοιχο προειδοποιητικό μήνυμα στο χρήστη, για να συμπληρώσει κατάλληλα τα πεδία. Όταν συμπληρωθούν τα πεδία λοιπόν, εφαρμόζουμε το stemming που περιγράψαμε στην 5.3.2 ενότητα , στη λέξη που έδωσε ο χρήστης καθώς στη Neo4j όλες οι λέξεις είναι ήδη χωρίς καταλήξεις και διαφορετικά, ακόμα και να υπήρχε η λέξη που επιθυμούσε δεν θα τη βρίσκαμε στο γράφο. Οπότε με τη λέξη και το βάρος που έχουμε πάρει, βάζουμε το Sigma.js να κάνει Query στη Neo4j. Αν δεν υπάρχει η λέξη/κλειδί που ζήτησε ο χρήστης τότε η απάντηση από το Neo4j είναι το κενό και δεν θα εμφανιστεί κάποιος γράφος. Οπότε εμφανίζεται κενή σελίδα με το κουμπί «graph» που οδηγεί πίσω στην αρχική. Διαφορετικά, θα εμφανιστεί ένας γράφος που θα έχει σαν κέντρο τη λέξη κλειδί που έδωσε ο χρήστης και όλους τους γειτονικούς κόμβους που ικανοποιούν το όριο βάρους που έθεσε. Πάλι υπάρχει πάνω αριστερά το κουμπί «graph» που οδηγεί στην προηγούμενη σελίδα. Στην περίπτωση, τέλος, που ο χρήστης δώσει μια υπαρκτή λέξη κλειδί, αλλά πολύ μεγάλο βάρος, θα εμφανιστεί μόνο ο κόμβος της λέξης. Παρακάτω παρουσιάζονται η αρχική σελίδα και η μορφή της κενής σελίδας.



Σελίδα Πρώτης Επιλογής από RSS

Κενή Σελίδα

Η δεύτερη επιλογή που δίνεται από το drop down κουτί είναι το «Twitter» . Σε αυτή την περίπτωση ο χρήστης δίνει μία λέξη-κλειδίπου επιθυμεί, πάλι στο πεδίο «Λέξη Αναζήτησης», ώστε να την αναζητήσουμε στο Twitter και να συλλέξουμε σχετικά tweets. Ο χρήστης δίνει επίσης την ώρα την οποία θέλει να διαρκέσει η συλλογή των tweets συμπληρώνοντας το πεδίο «Χρόνος Αναζήτησης» και τα αντίστοιχα πεδία των Ημερών , Ωρών , Λεπτών , Δευτερολέπτων τα «DD» «HH» «MM» «SS». Τέλος , δίνει το βάρος των ακμών που θέλει να του εμφανιστούν όταν τελειώσει η συλλογή και επεξεργασία των tweets και εμφανιστεί ο γράφος. Όπως και στην πρώτη περίπτωση, πριν ξεκινήσει η αναζήτηση γίνεται έλεγχος των δοσμένων δεδομένων από το χρήστη. Στο επιπλέον πεδίο του χρόνου γίνεται έλεγχος , ο οποίος ζητάει να έχει συμπληρωθεί με θετικό αριθμό τουλάχιστον ένα από τα τέσσερα πεδία του χρόνου. Όσα δεν έχουν συμπληρωθεί, θεωρούνται μηδενικά και μετατρέπουμε το συνολικό χρόνο σε χιλιοστά του δευτερολέπτου, ώστε να το δώσουμε στη σωστή μορφή σαν παράμετρο στο Twitter. Παρακάτω, παρουσιάζεται η δεύτερη μορφή της αρχικής σελίδας με τα επιπλέον πεδία.



Σελίδα Επιλογής Twitter

Τώρα, με το πάτημα του κομβίου «Αναζήτηση», δεν οδηγούμαστε στη σελίδα αποτελεσμάτων, αφού πρέπει να περάσει πρώτα ο χρόνος συλλογής από το Twitter και μετά ένας επιπλέον χρόνος για τη δημιουργία του γράφου από τα δεδομένα που μαζέψαμε. Η διαδικασία που ακολουθείται πιο συγκεκριμένα είναι η ακόλουθη. Όπως, έχουμε αναφέρει, έχει δημιουργηθεί ένας server στο Node.JS και ο οποίος “ακούει” στη θύρα 8080. Παίρνοντας, λοιπόν, τη λέξη-κλειδί και το χρόνο που επιθυμεί ο χρήστης τα στέλνουμε με POST REQUEST στη θύρα 8080, όπου τα λαμβάνει ο server. Από το REQUEST που λαμβάνει, κρατάει μόνο το “σώμα” και με Regular Expressions, βρίσκει τη λέξη-κλειδί και το χρόνο. Στη συνέχεια, ο server καλεί το Twitter με τα δεδομένα αυτά και περιμένει να τελειώσει η διαδικασία συλλογής δεδομένων και τα ανεβάζει στη FireBase. Μετά, γίνεται η μέθοδος που περιγράφεται αναλυτικά στις προηγούμενες ενότητες για τη δημιουργία του γράφου και όταν ολοκληρωθεί και αυτή, επιστρέφει ο server απάντηση ότι τελείωσε για να οδηγηθούμε στην παρουσίαση των αποτελεσμάτων. Κατά τη διάρκεια που δουλεύει ο server εμφανίζεται η παρακάτω σελίδα αναμονής.

---

graph

Δημιουργία Γράφου...

Σελίδα Αναμονής

6. Αποτελέσματα και Αξιολόγηση

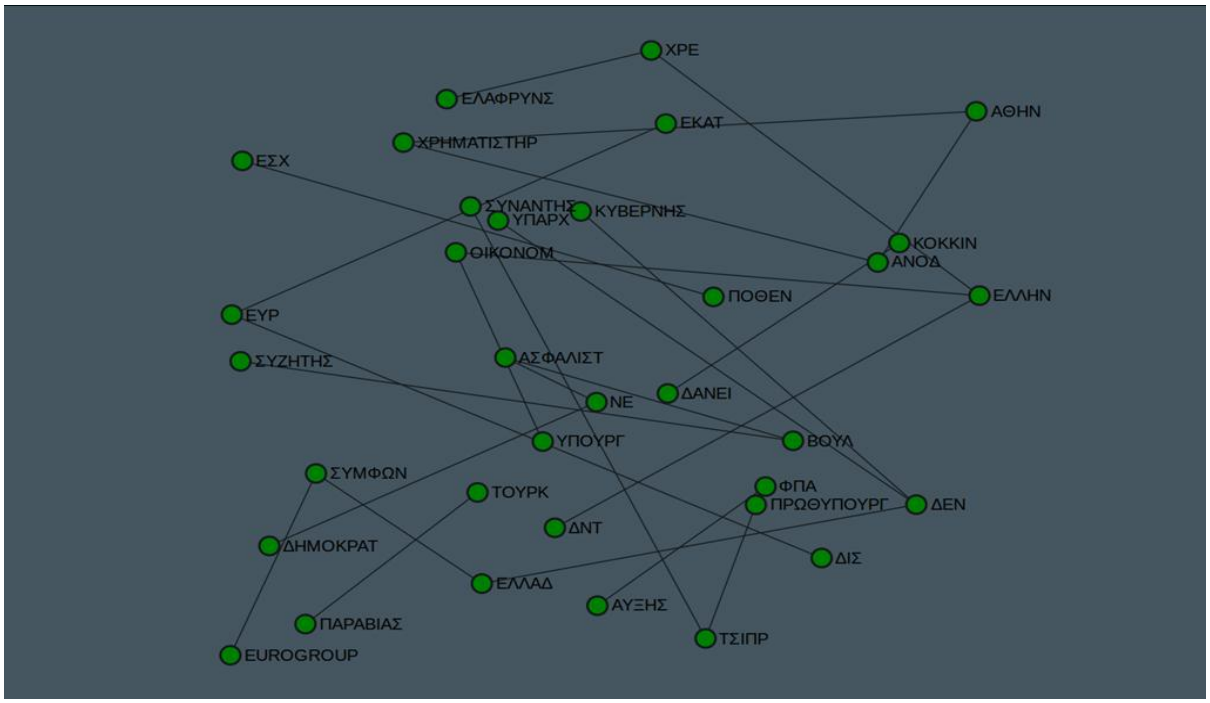
6.1 Αποτελέσματα της μεθόδου

Σε αυτή την ενότητα θα περιγραφούν κάποια από τα αποτελέσματα τα οποία είδαμε από τη μεθοδό μας. Γενικά ο σκοπός της παραπάνω μεθοδολογίας είναι να δημιουργήσουμε ένα γράφο με την οπτικοποίηση του οποίου θα μπορέσει ο χρήστης να βγάλει κάποια λογικά πορίσματα. Για αυτό είναι πολύ σημαντική η σωστή συλλογή των δεδομένων μας και η κατάλληλη μετατροπή τους σε μορφή κατάλληλη για να τα παρουσιάσουμε σαν γράφο. Πάντα αυτό που εξετάζουμε στο γράφο είναι οι συσχετίσεις μεταξύ των κόμβων/λέξεων-κλειδιών (πρακτικά δηλαδή τις ακμές που ενώνουν τους κόμβους) και βγάζουμε συμπεράσματα μόνο από αυτές.

6.1.1 Παραδείγματα υπογράφων από RSS

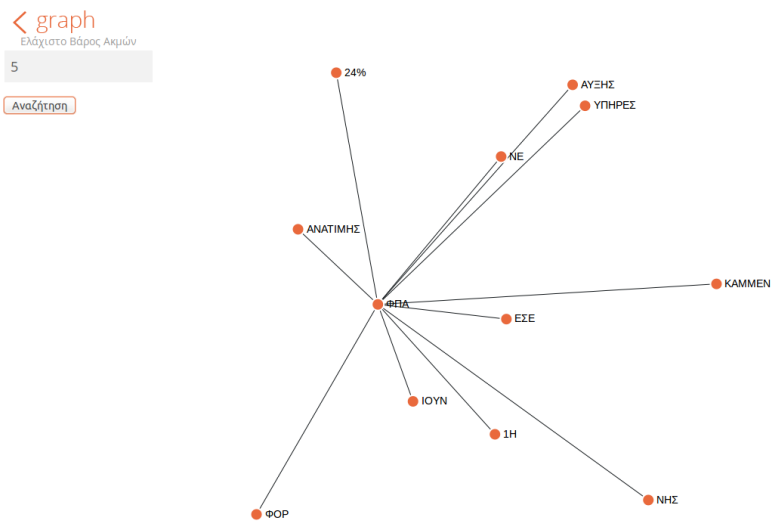
Όλα τα αποτελέσματα προέκυψαν από μελέτη του γράφου και εφαρμογή queries, τέτοιων ώστε οι υπο-γράφοι που μας επιστρέφουν, να δίνουν κάποιες σαφείς πληροφορίες. Η λογική πίσω από την εύρεση των ισχυρών συσχετίσεων είναι ότι εξετάζουμε αρχικά τους κόμβους που έχουν ακμές με μεγάλα βάρη (φιλτράρουμε το γράφο με ένα query edge.weight>threshold), χωρίς κάποια κεντρική λέξη κλειδί (Εικόνα 1). Αυτό πρακτικά σημαίνει ότι παίρνουμε όλες τις ακμές που συνδέουν οποιουσδήποτε 2 κόμβους με βάρος μεγαλύτερο του 18 (για την Εικόνα 1). Αυτό που μας δίνουν ποιοτικά αυτές οι ακμές και το οποίο εξ αρχής ζητούσαμε, είναι ένα σύνολο λέξεων που συναντήθηκε συχνά στην αρθρογραφία (ή στο Twitter) και μάλιστα συναντήθηκαν τουλάχιστον τόσες φορές με τη λέξη που συνδέονται. Μέσα από αυτό το πρίσμα, τη μαθηματική βαρύτητα των συσχετίσεων, χωρίς καμία επεξεργασία γλώσσας σε κάποιο βήμα της μεθοδολογίας, ο χρήστης πρέπει να βγάλει όποια συμπεράσματα είναι δυνατά. Έτσι, το βασικό πράγμα που γνωρίζει ο χρήστης με το πέρας της παραπάνω διαδικασίας είναι για ποιες λέξεις υπήρχαν κοινές και πυκνές αναφορές σε άρθρα. Για τη διευκόλυνση της εποπτείας του χρήστη επί του γράφου, χρησιμοποιείται ένα plugin του SigmaJS, το neighborhoods. Αυτό το plugin δίνει τη δυνατότητα με το πάτημα ενός κόμβου με το ποντίκι στο γράφο, να κάνει διάφανους όλους τους κόμβους, πέρα από τους γειτονικούς κόμβους του επιλεγμένου. Ο χρήστης είναι σε θέση πλέον εύκολα να εξετάσει το γράφο και να εντοπίσει ποια λέξη-κλειδί επιθυμεί, ώστε στη συνέχεια να ερευνήσει το γράφο με βάση αυτή τη λέξη. Εφαρμόζεται, τελικά query με αυτή λέξη-κλειδί που δρα ως κέντρο στον υπογράφο που θα παρουσιαστεί και μεταβάλλοντας το όριο του επιτρεπτού βάρους των ακμών διαμορφώνονται οι υπογράφοι από τη σύγκριση των οποίων μπορούν να εξαχθούν τα πιθανά συμπεράσματα.

Εικόνα 1(Γράφος με ακμές βάρους >18)

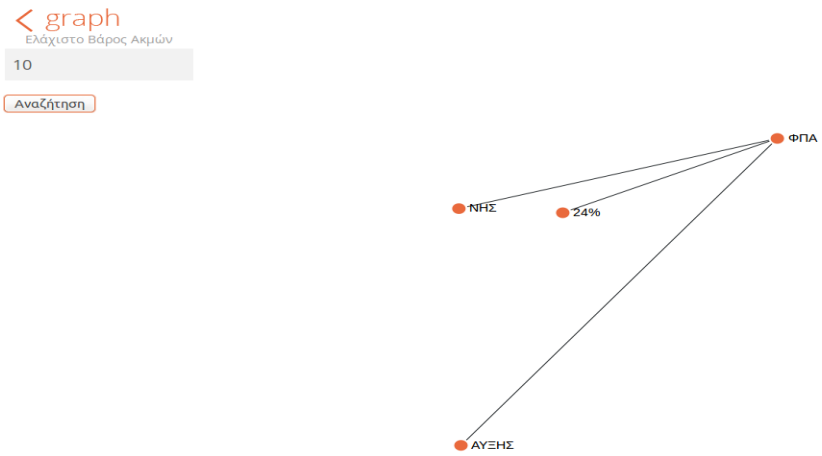


Μερικά αποτελέσματα που προέκυψαν με βάση την παραπάνω λογική παρατίθενται στη συνέχεια. Ξεκινώντας από το γράφο στην Εικόνα 1, εντοπίζουμε τον όρο “ΦΠΑ” και παρατηρούμε ότι έχει ισχυρές συνδέσεις με τον κόμβο “ΑΥΞΗΣ”. Έτσι, εφαρμόζοντας query με λέξη-κλειδί το “ΦΠΑ” και ένα μικρό βάρος για τις ακμές παίρνουμε την εικόνα 2α. Στην εικόνα αυτή, φαίνεται ο κεντρικός κόμβος “ΦΠΑ” και οι συνδέσεις που έχει με 9 διαφορετικούς κόμβους, με διάφορα βάρη. Από αυτό το στιγμιότυπο δεν μπορούμε να εξάγουμε σαφές συμπέρασμα και για αυτό το λόγο ανεβάζουμε το threshold του βάρους και παίρνουμε την εικόνα 2β. Σε αυτή την εικόνα φαίνεται ξεκάθαρα η ισχυρή σύνδεση με τους δύο άλλους κόμβους “ΑΥΞΗΣ(Η)” , ”24%” και ”ΝΗΣ(ΙΑ)” και άρα ο χρήστης μπορεί να συμπεράνει ότι κατά τη διάρκεια 01/04/2016 – 01/06/2016 υπήρξε σαν θέμα η αύξηση του ΦΠΑ στα νησιά.

Εικόνα 2α(ΦΠΑ μικρό βάρος)

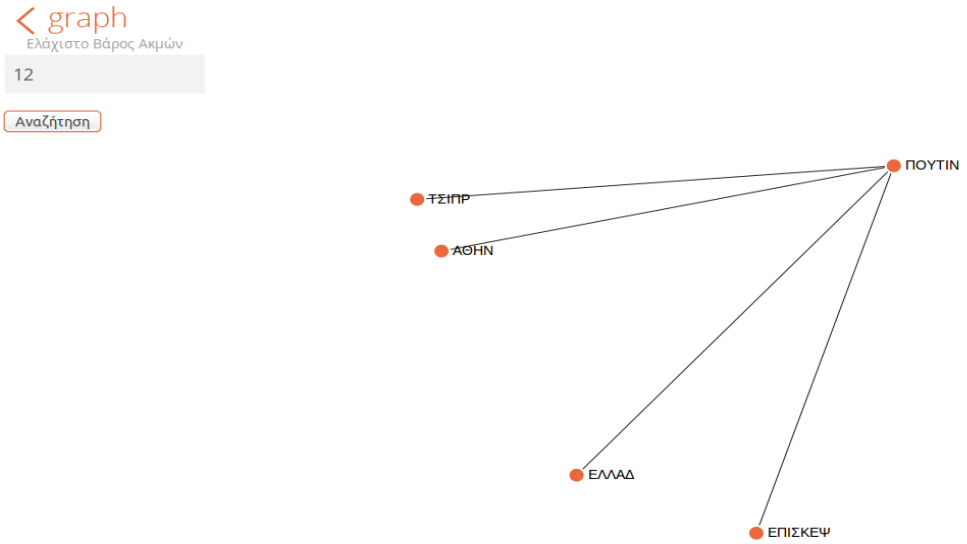


Εικόνα 2β(ΦΠΑ)



Ένα άλλο παράδειγμα που προέκυψε από την ίδια διαδικασία με την παραπάνω για το γεγονός της επίσκεψης του Βλαντιμίρ Πούτιν στην Ελλάδα και τη συνάντηση του με τον πρωθυπουργό Αλ. Τσίπρα. Όπως βλέπουμε στην εικόνα 3 έχουμε σαν κεντρικό κόμβο το "ΠΟΥΤΙΝ" το οποίο συνδέεται με τους κόμβους "ΤΣΙΠΡ(Α)", "ΕΠΙΣΚΕΨ(Η)" ,"ΕΛΛΑΔ(Α)" και "ΑΘΗΝ(Α)" και το σύνολο αυτών οδηγεί στην παραπάνω πρόταση και συμπέρασμα.

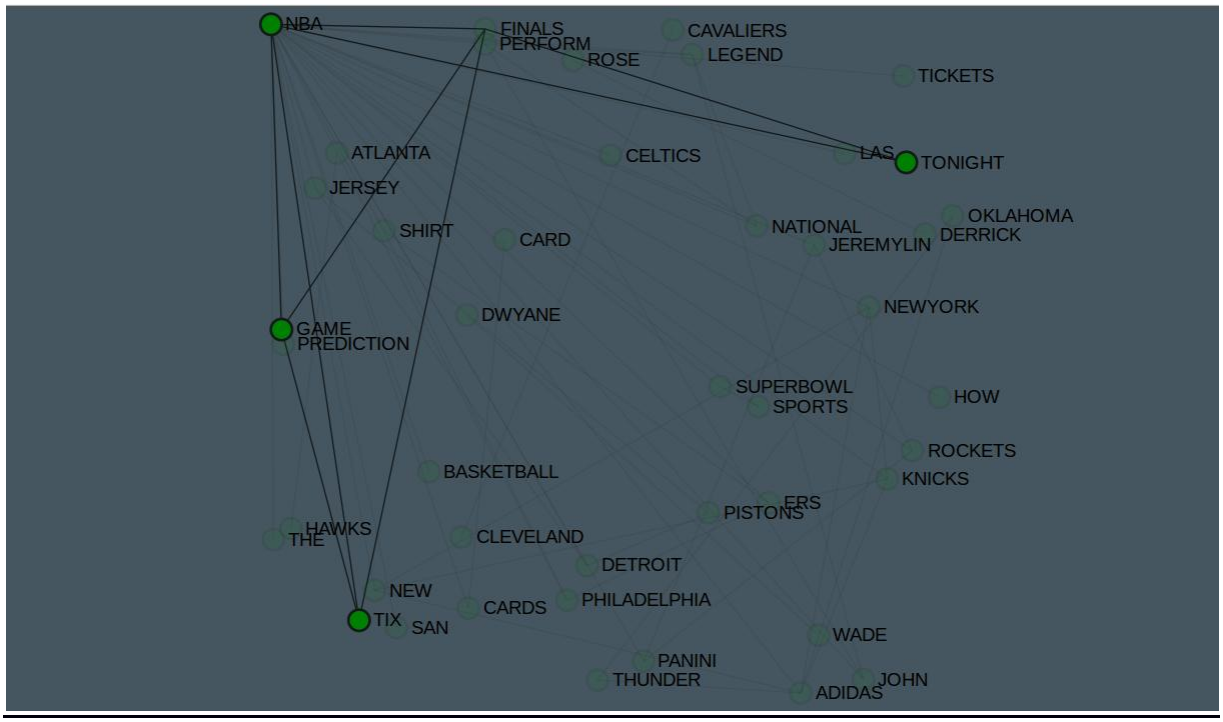
Εικόνα 3 (Επίσκεψη Πούτιν)



6.1.2 Παραδείγματα από Twitter

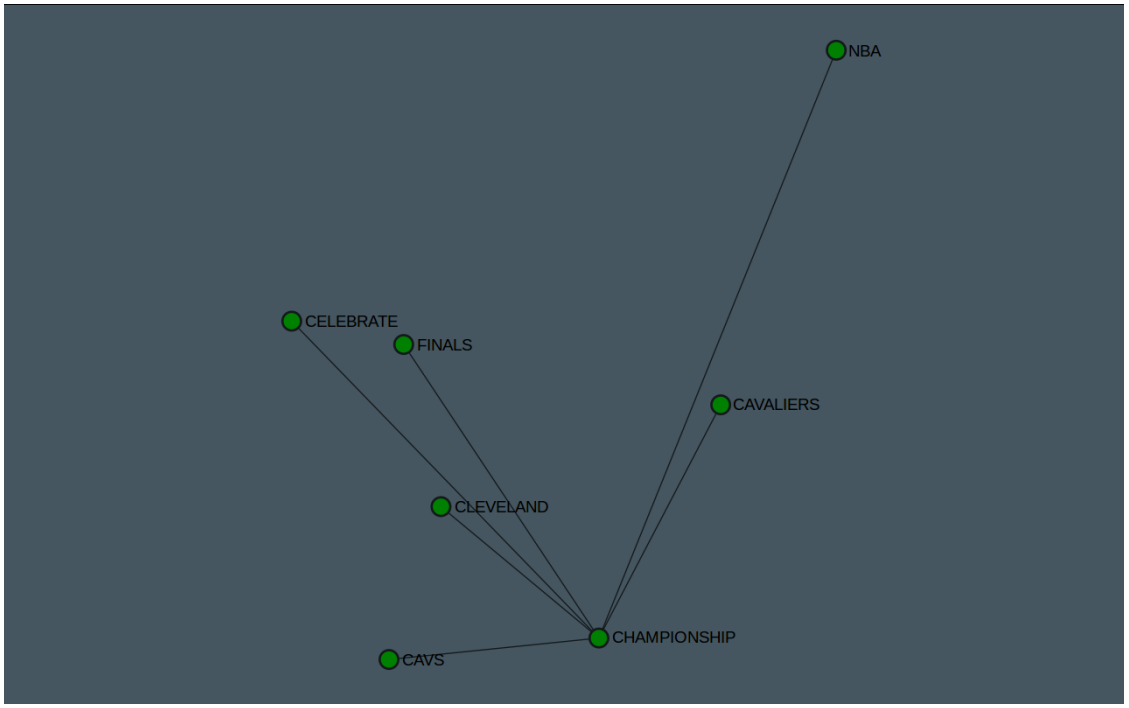
Τα αποτελέσματα που παρουσιάζονται σε αυτή την ενότητα προέρχονται από το Twitter και δημιουργήθηκαν με ένα δείγμα από live tweets, με τη διαδικασία που έχει περιγραφεί στις προηγούμενες ενότητες. Λόγω αυτού τα αποτελέσματα από το Twitter, σε αντίθεση με τα προηγούμενα, είναι αδύνατο να εξαχθούν ξανά, καθώς δεν έχουμε αποθηκεύσει τα tweets. Το Twitter,επίσης, δίνει γενικά καλύτερα αποτελέσματα σε αναζητήσεις κάποιου Αγγλικού όρου στο tweet stream και για αυτό επιλέχθηκαν Αγγλικές λέξεις για αναζήτηση στο Twitter. Αυτό συμβαίνει διότι στα Ελληνικά, πρώτον είναι πολύ μικρότερος ο αριθμός των ενεργών χρηστών που θα κάνουν κάποιο tweet χρησιμοποιώντας ελληνικούς χαρακτήρες (σε αντίθεση είναι συχνό φαινόμενο να κάνουν Έλληνες κάποιο tweet στα Αγγλικά) και δεύτερον το track του stream API, στα Ελληνικά είναι case sensitive (τα κεφαλαία θεωρούνται διαφορετικά από τα πεζά γράμματα), ενώ στα Αγγλικά δεν είναι, πράγμα που μειώνει εμφανώς τα tweets που παίρνουμε. Έτσι, διαλέξαμε πράγματα που να είναι επίκαιρα την περίοδο της αναζήτησης. Πρώτα, έχουμε ένα παράδειγμα που επιλέχθηκε ο όρος 'nba' στις 20/06/2016 για μία ώρα, αρκετή για ένα τόσο δημοφιλή όρο ώστε να μπορέσει να δώσει αρκετά δεδομένα και πήραμε τον υπο-γράφο που φαίνεται στην παρακάτω εικόνα (έχουμε εστιάσει στον κόμβο 'finals'). Στις 19/06/2016 διεξήχθη ο 7^{ος} και τελευταίος αγώνας των τελικών του NBA και όπως φαίνεται από το συγκεκριμένο υπο-γράφο της Εικόνας 4, μπορούμε να συμπεράνουμε από αυτόν, ότι εκείνο το βράδυ ήταν οι τελικοί του nba.

Εικόνα 4α (nba finals)



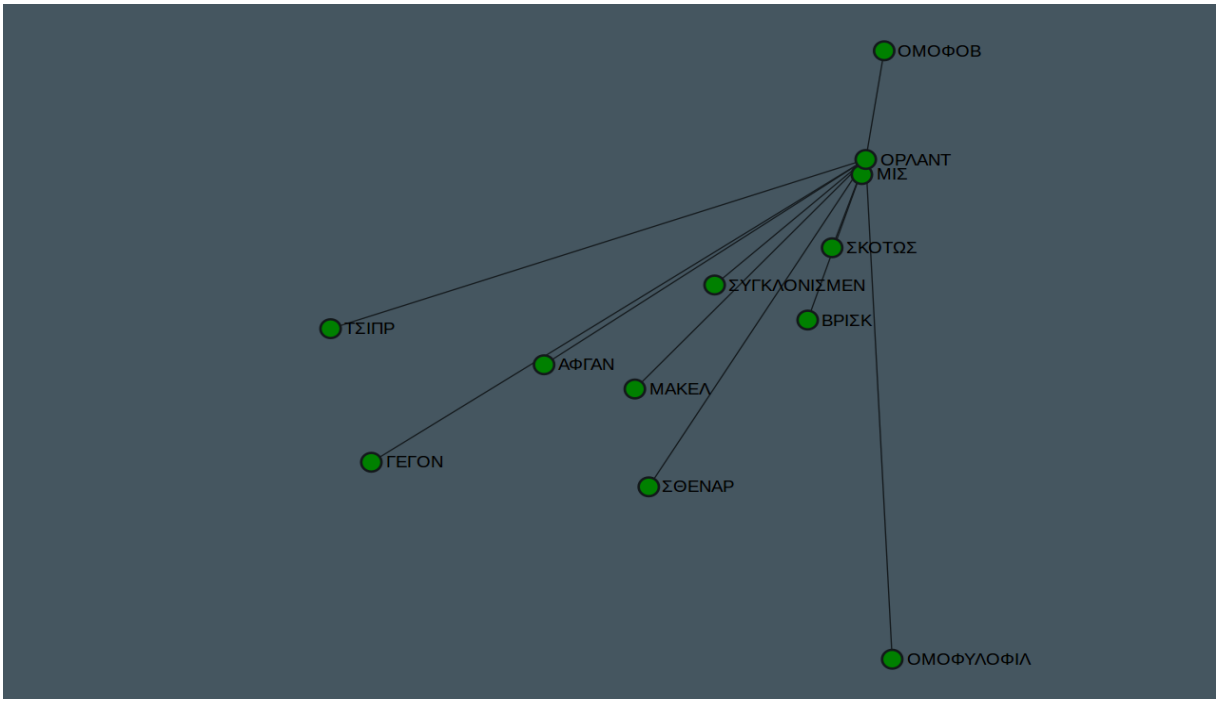
Και στη συνέχεια από την παρακάτω εικόνα, μπορούμε να συμπεράνουμε ότι η ομάδα που επικράτησε στους τελικούς είναι οι Cleveland Cavaliers, καθώς ξεκάθαρα βλέπουμε ότι ο κόμβος “Championship” συνδέεται με τους κόμβους “Cleveland” “Cavaliers” και “Celebrate” (που σημαίνει γιορτάζουν).

Εικόνα 4β (Cavaliers Champions)



Ένα δεύτερο παράδειγμα , είναι από tweets στις 13/06/2016 κάνοντας track τη λέξη-κλειδί “Ορλαντο” ('ΟΡΛΑΝΤ' μετά το stemming). Έγινε search με τον ελληνικό όρο αυτή τη φορά στο Twitter και το αποτέλεσμα φαίνεται στο συγκεκριμένο, παρακάτω υπο-γράφο, όπου παρουσιάζονται όλες οι λέξεις-κλειδία,πληροφορίες για το συμβάν που έγινε.

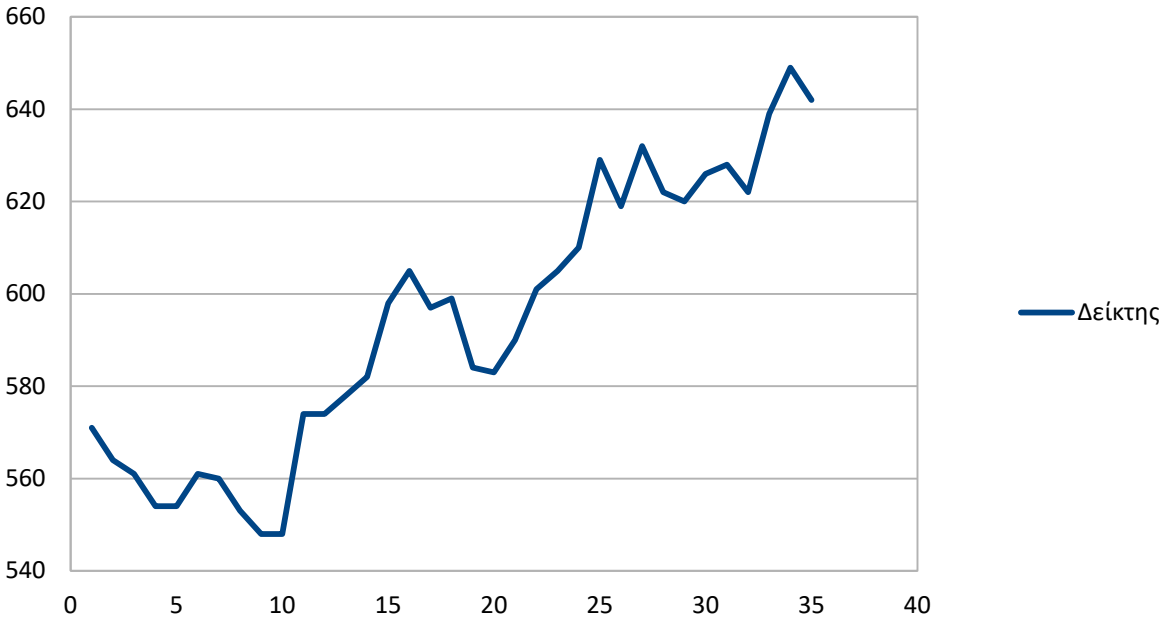
Εικόνα 5 (Orlando)



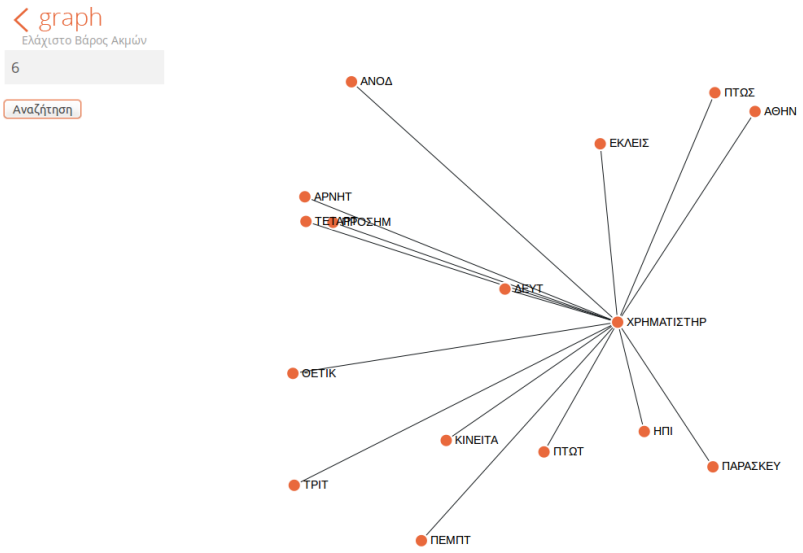
6.1.3 Περίπτωση ‘Χρηματιστηρίου’

Από τα άρθρα που πήραμε από το RSS, είδαμε ότι ένας βασικός υπο-γράφος που εμφανίζεται είναι με κέντρο τη λέξη-κλειδί “ΧΡΗΜΑΤΙΣΗΡ(ΙΟ)” (stemmed) , το οποίο συνδέεται με λέξεις όπως το “ΑΝΟΔ(ΟΣ)” , ”ΠΤΩΣ(Η)” , ”ΠΡΟΣΙΜ(Ο)” , ”ΘΕΤΙΚ(Ο)” και “ΑΡΝΗΤΙΚ(Ο)”. Έτσι, αυξάνοντας σταδιακά το κατώφλι (threshold) για το βάρος προκύπτουν οι 3 γράφοι των εικόνων 7-9. Σε αυτούς παρατηρείται να φεύγουν πρώτα οι όροι ”ΠΤΩΣ(Η)” και “ΑΡΝΗΤΙΚ(Ο)” (μεταξύ άλλων) και να μένει τελικά ο κόμβος “ΑΝΟΔ(ΟΣ)” (από αυτούς που προσδίδουν κάποιο πρόσημο και είναι και αυτοί που μας ενδιαφέρουν) . Παράλληλα , είδαμε και ποσοτικά, ότι έχουμε 25 εμφανίσεις των ”ΠΤΩΣ(Η)” και “ΑΡΝΗΤΙΚ(Ο)” , ενώ 45 των “ΑΝΟΔ(ΟΣ)” και ”ΘΕΤΙΚ(Ο)”. Αυτό το παρατηρήσαμε από τα βάρη μεταξύ του κάθε όρου, από τους παραπάνω, με το “ΧΡΗΜΑΤΙΣΗΡ”. Έτσι συμπεράναμε πως λογικά θα παρατηρηθεί γενική άνοδος του δείκτη για το διάστημα το οποίο μαζεύαμε RSS Feeds (01/04 – 01/06). Παρακάτω παρατίθεται η κίνηση του δείκτη του χρηματιστηρίου για το ίδιο διάστημα και όντως παρατηρείται μια αναμενόμενη κίνηση του με βάση τα πορίσματα που βγάλαμε. Καταλήγουμε λοιπόν ότι τα άρθρα, αναφέροντας σχεδόν καθημερινά την τάση του χρηματιστηρίου, μας έδωσαν αρκετά δεδομένα για να μπορέσουμε μέσω του γράφου να εξάγουμε ασφαλές συμπέρασμα για το δείκτη.

Εικόνα 6 (Δείκτης Χρηματιστηρίου)



Εικόνα 7(Χρηματιστήριο για βάρος ακμών >6)



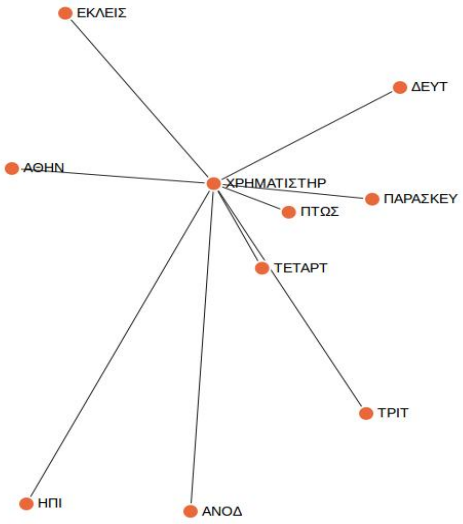
Εικόνα 8(Χρηματιστήριο για βάρος ακμών >9)

graph

Ελάχιστο Βάρος Ακμών

9

Αναζήτηση



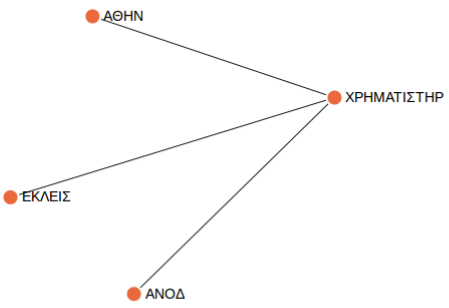
Εικόνα 9(Χρηματιστήριο για βάρος ακμών >13)

graph

Ελάχιστο Βάρος Ακμών

20

Αναζήτηση



6.2 Σύνοψη Συμπερασμάτων και Αξιολόγηση

Θα ξεκινήσουμε με μερικά σχόλια επί της όλης διαδικασίας και των αποτελεσμάτων. Αρχικά, πρέπει να εστιάσουμε στον σκοπό που επιθυμούσαμε να καλύψει η παραπάνω εφαρμογή. Ο αρχικός στόχος μας ήταν, να φτιαχτεί ένα σύστημα το οποίο θα μπορεί να ανταποκρίνεται στον συνεχώς αυξανόμενο όγκο πληροφοριών (είτε από μέσα ενημέρωσης είτε από τα μέσα κοινωνικής δικτύωσης) και να μπορεί να επεξεργαστεί στη συνέχεια αυτά τα δεδομένα. Η επεξεργασία του συστήματος γίνεται με βασικό σκεπτικό την οπτικοποίηση της πληροφορίας, μέσω απλών διαδικασιών και για αυτό οδηγηθήκαμε στην αναπαράσταση των δεδομένων με γράφους. Θέλαμε ο γράφος να προκύπτει χωρίς τη χρήση κάποιας γλωσσικής ανάλυσης, αλλά με απλά μαθηματικό τρόπο όπου λέξεις που βρίσκονται συχνά μαζί καταλήγουν γειτονικοί κόμβοι με μεγάλα βάρη στο γράφο. Έτσι, μέσα από αυτό το σύστημα θέλουμε να εξετάσουμε αν οποιοσδήποτε χρήστης μπορεί να το χρησιμοποιήσει και να εξάγει κάποια χρήσιμα αποτελέσματα. Ακριβώς επειδή τα δεδομένα δεν υφίστανται κάποια ειδική επεξεργασία κατά τη συλλογή τους ή εξαγωγή του γράφου από αυτά, αναμένεται ότι τα συμπεράσματα τα οποία θα παρθούν από αυτά, να είναι αντίστοιχης φύσης. Αυτό σημαίνει, για παράδειγμα παίρνοντας τα άρθρα των εφημερίδων σαν δεδομένα τα αποτελέσματα που μπορεί να εξαχθούν θα είναι κάποια πολιτικά ή οικονομικά γεγονότα που σχολιάστηκαν από αυτά. Αντίστοιχα για το Twitter, ο χρήστης αναλόγως με ποια λέξη-κλειδί ερευνήσει το Twitter θα λάβει και παρόμοια αποτελέσματα. Τα παραπάνω φαίνονται ξεκάθαρα από τα αποτελέσματα που παρουσιάζονται στην ενότητα 6.1, καθώς βλέπουμε ότι από τα άρθρα βγάζουμε συμπεράσματα για το ΦΠΑ, την επίσκεψη Πούτιν, θέματα πολιτικής/οικονομικής φύσεως, ενώ από το Twitter βγάλαμε συμπεράσματα για τους τελικούς του NBA και το μαζική δολοφονία στο Ορλάντο, αθλητικής και κοινωνικής φύσης ζητήματα αντίστοιχα.

Αυτά δείχνουν ότι η εφαρμογή μας μπόρεσε να οδηγήσει σε αξιοποιήσιμα αποτελέσματα σε διαφορετικούς τομείς και μάλιστα και σε δύο διαφορετικές γλώσσες, χωρίς να χρειάζεται ο χρήστης να παραμετροποιήσει κάτι. Άρα, σε πρώτο στάδιο μπορούμε να πούμε ότι η μέθοδος μας, αποφεύγοντας τη χρήση Επεξεργασίας Φυσικής Γλώσσας (NLP) , έχει το πλεονέκτημα ανεξαρτησίας γλώσσας και θέματος που εξετάζουμε. Από όλα αυτά συμπεραίνουμε ότι η εφαρμογή έχει μεγάλη προσαρμοστικότητα (**Adaptability**) που δίνει πολλές δυνατότητες και στην περαιτέρω έρευνα για αποτελέσματα σε άλλους τομείς και τη παρουσίαση διαφόρων προβλημάτων με γράφους. Επίσης, δίνεται η δυνατότητα για εύκολη ενσωμάτωση νέων στοιχείων (**Extensibility**).Τα ήδη υπάρχοντα άρθρα είναι αποθηκευμένα στη FireBase και με τις ίδιες διαδικασίες μπορούμε να συνεχίσουμε να αποθηκεύουμε άρθρα από οποιαδήποτε πηγή RSS επιθυμούμε. Θα είχε ενδιαφέρον η μελέτη ενός γράφου με δεδομένα πολύ περισσότερα ή/και διαφορετικά από τα ήδη συλλεγόμενα.

Παράλληλα, η εργασία έχει δομηθεί κατάλληλα ώστε να διακρίνεται σε επιμέρους στάδια, να υπάρχει δηλαδή **Modularity**. Ο κώδικας μας έχει ξεκάθαρα διαχωρισμένα τα στάδια του,

συγκεκριμένα και διαφορετικά modules (μονάδες) για την εξόρυξη δεδομένων από τις πηγές μας, για την δημιουργία των λέξεων-κλειδιών και τη δημιουργία του γράφου από τα δεδομένα. Αυτή η λογική ανάπτυξης κώδικα βοηθάει ιδιαίτερα σε μεγάλα projects για τη συντήρηση και έλεγχο του κώδικα (**Maintanability**). Αυτό ήταν απαραίτητο και για την εξασφάλιση της αναδημιουργίας του γράφου στη δική μας περίπτωση, καθώς θέλαμε να έχουμε σταθερά τα δεδομένα μας, κατά τη διάρκεια που ερευνούσαμε τον τρόπο με τον οποίο θα τα παρουσιάσουμε μέσω του γράφου. Παράλληλα διευκολύνει και την επαναχρησιμοποίηση κομματιών/modules που έχουμε ήδη φτιάξει σε άλλα projects που χρειάζονται κάποια συγκεκριμένη λειτουργία. Αυτή η επαναχρησιμοποίηση (**Reusability**) και αξιοποίηση των ήδη υπαρχόντων πόρων, μπορεί να ελαττώσει τόσο το χρόνο όσο και το κόστος δημιουργίας τέτοιων project. Αυτή η λογική ενθαρρύνεται γενικότερα από την κοινότητα ανοιχτού λογισμικού και κατεπέκταση του Node.JS. Όπως, ήδη, αναφέραμε το Node.Js παρέχει ένα διαχειριστή πακέτων, το Node Package Manager (NPM), από το οποίο χρησιμοποιήσαμε αντίστοιχα βιβλιοθήκες ανοιχτού κώδικα για την υλοποίηση της συγκεκριμένης εφαρμογής. Αντίστοιχα και η Sigma.Js είναι μια «ανοιχτή» βιβλιοθήκη και γενικότερα ολόκληρη η υλοποίηση της εφαρμογής βασίστηκε στα **Open Standards**[30][31].

Πέραν όμως των παραπάνω τα οποία αφορούν κυρίως τη λογική με την οποία χτίστηκε ο κώδικας της εφαρμογής, προσπαθήσαμε να είναι εύκολα χρησιμοποιήσιμη από όλους τους χρήστες (**Usability**). Αυτό οδήγησε στην δημιουργία της διεπαφής που αναφέραμε, η οποία έχει πολύ απλή και κατανοητή δομή. Ο χρήστης χρειάζεται να συμπληρώσει τα απαραίτητα μόνο πεδία για τη δημιουργία των αποτελεσμάτων και την εμφάνιση του γράφου. Αυτή η οπτικοποίηση των αποτελεσμάτων δίνει τη δυνατότητα να δει συσχετίσεις άμεσα και γρήγορα, χωρίς να χρειάζεται να ανατρέξει στην πληθώρα των αντίστοιχων πηγών. Προφανώς, πρέπει εδώ να αναφέρουμε ότι ο γράφος δεν δίνει πάντα χρήσιμες πληροφορίες και μπορεί να υπάρξουν γράφοι που δίνουν είτε τετριμμένες πληροφορίες είτε και συσχετίσεις που δεν βγάζουν κάποιο νόημα.

Από όλα τα παραπάνω αξιολογούμε ότι η συγκεκριμένη εργασία κάνει ένα βήμα προς τη δημιουργία μιας λογικής και μιας μεθοδολογίας για την αντιμετώπιση της ανάλυσης δεδομένων με τη χρήση των γράφων, έχοντας και υψηλή χρηστικότητα, αλλά και κατάλληλη δομή για πολλές μελλοντικές επεκτάσεις.

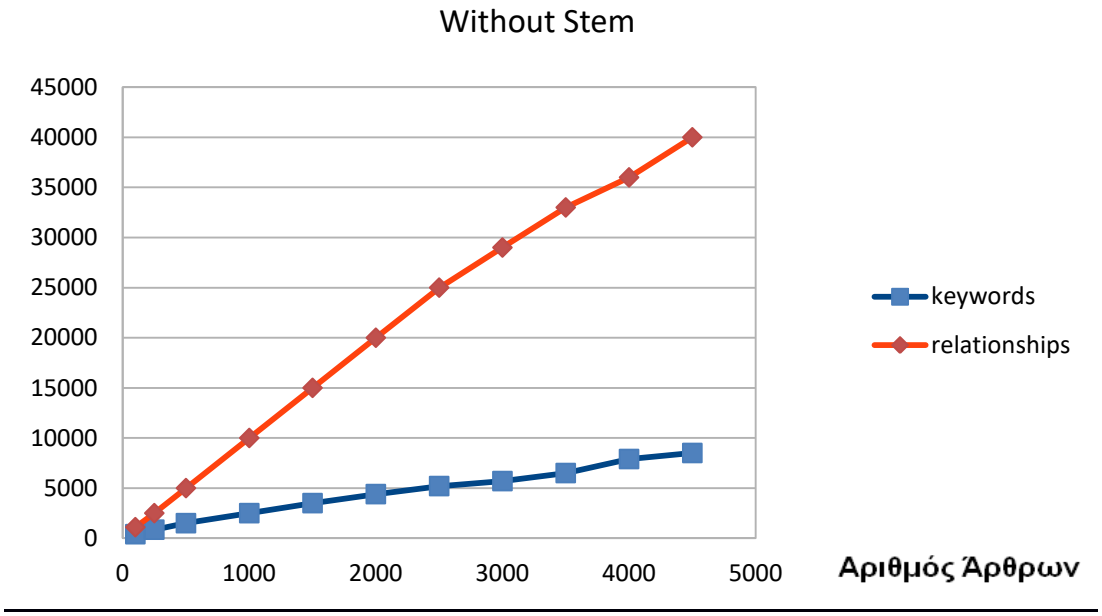
Κάποια στατιστικά συμπεράσματα στην λήψη δεδομένων και στη επεξεργασία των λέξεων κλειδιών. Για το Twitter παρατηρήσαμε τα εξής :

- Ακόμα και για τους πιο δημοφιλείς (από πλευρά αναζήτησης) Έλληνες πολιτικούς , τα tweets που τους αφορούν δεν ξεπερνούν τα 40 την ώρα ,ενώ για αντίστοιχους Αμερικάνους είναι χιλιάδες ανά ώρα.
- Παρατηρούνται πιο πολλά retweets παρά νέα tweets ,για αυτό και είναι απαραίτητος ο έλεγχος για retweets ,γιατί θα μας οδηγήσουν σε γράφους με πολύ διαφορετικά βάρη.

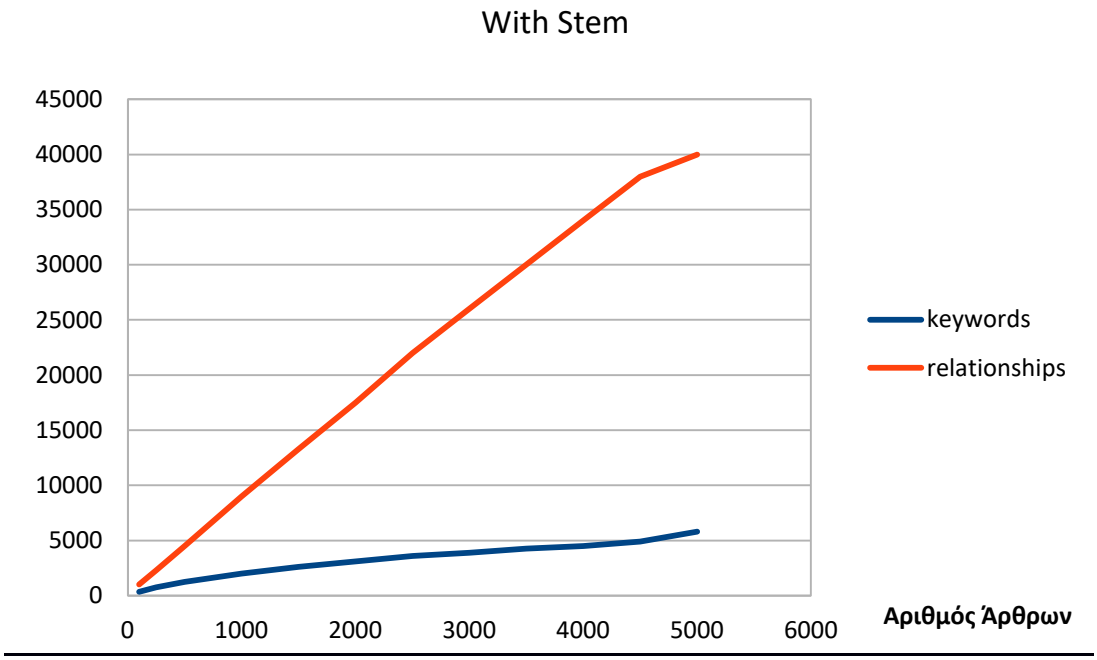
Αντίστοιχα στοιχεία που βγάλαμε από τα RSS είναι :

- Γενικά κάθε RSS μπορεί να έχει μέχρι 20 άρθρα τη φορά. Τα νέα άρθρα που πήραμε σε χρονική περίοδο μίας εβδομάδας από όλα τα RSS μας ήταν περίπου 1000 άρθρα (εξετάζοντας για νέα άρθρα κάθε 90 λεπτά)
- Φαίνεται με το αυξανόμενο πλήθος των άρθρων ότι γενικά οι λέξεις που χρησιμοποιούνται στην ελληνική πολιτική και οικονομική αρθογραφία (μετά το stemming) συγκλίνουν στις 6000 λέξεις. Αυτή η σύγκλιση γίνεται και ακόμα πιο φανερή όταν συγκρίνουμε τον αριθμό λέξεων με stemming και χωρίς . Αυτό μας οδηγεί στο συμπέρασμα ότι με αρκετό πλήθος κειμένων μπορούμε να οδηγηθούμε σε κορεσμό του γράφου όπου όλα συνδέονται με όλα για αυτό και είναι σημαντικός ο έλεγχος των βαρών.

Εικόνα 10 (Άρθρα-Λέξεις χωρίς Stemming)



Εικόνα 11 (Άρθρα-Λέξεις με Stemming)



7. Επίλογος

Τα τελευταία χρόνια παρατηρείται μεγάλη αύξηση των δεδομένων και έτσι γίνεται απαραίτητη η ανάπτυξη εφαρμογών για την επεξεργασία και ανάλυση αυτών. Πιο συγκεκριμένα δημιουργούνται νέοι τρόποι με τους οποίους αντιμετωπίζεται αυτός ο τεράστιος όγκος πληροφοριών, όπως η χρήση γράφων για την ανάλυση και σύνδεση των δεδομένων. Ένα πρόσφατο τέτοιο παράδειγμα είναι η χρήση του Linkurious (παρόμοια βιβλιοθήκη με τη Sigma.JS) και της Neo4j, για την ανάλυση των 2.6 Terabytes δεδομένων των Panama Papers και η αναπαράσταση όλων των συσχετίσεων γίνεται μέσα από γράφους.[32]

Στην εργασία αυτή παρουσιάζεται ένα αντίστοιχο στη λογική σύστημα, για την διαχείριση των πληροφοριών που προέρχονται από τα άρθρα των εφημερίδων η Καθημερινή, το Βήμα, το news247 και η Ναυτεμπορική , αλλά και από την ιστοσελίδα κοινωνικής δικτύωσης Twitter. Το σύστημα μας, αρχικά συλλέγει και αποθηκεύει τα RSS Feeds των παραπάνω εφημεριδών, αλλά δίνει και τη δυνατότητα για να ψάξει τιτιβίσματα (*tweets*). Αυτά επεξεργάζονται και αποθηκεύονται στη συνέχεια σε μια μη-σχεσιακή (*NoSQL*) βάση δεδομένων γράφου (*graph database*), αφού πρώτα έχουν ξεχωριστεί σε λέξεις-κλειδιά. Τελικά, με τη χρήση των κατάλληλων εργαλείων, παρουσιάζονται τα δεδομένα αυτά στον χρήστη σε μορφή γράφων. Μέσα από την εφαρμογή μας και μέσω ενός λειτουργικού γραφικού περιβάλλοντος που δημιουργήθηκε, ο κάθε χρήστης, χωρίς να απαιτείται να γνωρίζει τη γλώσσα ερωτημάτων Cypher, μπορεί να αλληλεπιδρά απευθείας με την βάση δεδομένων.Δίνεται λοιπόν η δυνατότητα στον χρήστη, να ανασύρει από την βάση δεδομένων το γράφο με κέντρο τη λέξη-κλειδί που επιθυμεί και το βάρος των συσχετίσεων που θέλει.

Η εφαρμογή μας έδωσε διάφορα αποτελέσματα σε ποικίλες θεματικές κατηγορίες, τόσο σε Ελληνική αλλά και Αγγλική γλώσσα, έχει όμως αρκετά περιθώρια βελτίωσης. Καταρχάς, στον χρόνο δημιουργίας του γράφου, ο οποίος λόγω του μεγάλου αριθμού κόμβων και κυρίως ακμών που έχει, χρειάζεται αρκετή ώρα για την ολοκλήρωση του. Το ίδιο ισχύει και κατά την προσθήκη νέων κόμβων και ακμών, διότι για κάθε ένα από αυτά γίνεται ελέγχος μοναδικότητας στο σύνολο του γράφου, οπότε ακόμα και μικρός να είναι ο αριθμός που θα προστεθεί, ο χρόνος εισαγωγής τους θα είναι μεγάλος. Παράλληλα, υπάρχουν εργαλεία όπως το Linkurious, τα οποία όμως δεν είναι ανοιχτού λογισμικού και τα οποία δίνουν πολλές νέες δυνατότητες στην αναπαράσταση των γράφων, τόσο στο μέγεθος που μπορούν να υλοποιήσουν χωρίς να δημιουργούν δυσκολίες στο σύστημα, όσο και στις επιλογές που δίνουν στο χρήστη και στους τρόπους που μπορεί να αλληλεπιδράσει με το γράφο(όπως τρισδιάστατοι γράφοι). Σημαντική, τέλος, θα ήταν η εύρεση ένος αποτελεσματικού τρόπου αναγνώρισης και αντιμετώπισης των spam μηνυμάτων στο Twitter. Τα spam μηνύματα, αλλοιώνουν το δείγμα των δεδομένων(tweets) που συλλέγουμε από το Twitter και μπορούν να οδηγήσουν σε αποπροσανατολιστικά αποτελέσματα, καθώς το κέντρο βάρους του γράφου που θα προκύψει, είναι πολύ πιθανό να εστιαστεί γύρω από αυτά.

Πέραν όμως των παραπάνω βελτιώσεων, η εφαρμογή έχει πολλές δυνατότητες και για μελλοντικές επεκτάσεις. Ξεκινώντας με το πιο απλό, μπορεί να ελεγχθεί το ίδιο σύστημα σε άλλες πηγές δεδομένων, πχ. αθλητικές εφημερίδες ή μετεωρολογικά δεδομένα, καθώς όπως ήδη αναφέραμε η εφαρμογή μας χαρακτηρίζεται από προσαρμοστικότητα, αφού η μεθοδολογία παραμένει ίδια ανεξαρτήτως του πεδίου των δεδομένων. Επίσης, θα μπορούσε να επεκταθεί λειτουργικά το υπάρχον γραφικό περιβάλλον που δημιουργήσαμε και οι επιλογές που δίνει στο χρήστη, όπως για παράδειγμα τη δυνατότητα να διαγράφει κόμβους που πιθανά να μην θέλει ή να διαγράφει και να δημιουργεί τον αρχικό γράφο από την αρχή ή να εμφανίζει πέρα από τους γειτονικούς κόμβους.

Μια άλλη πιθανή επέκταση είναι η προσθήκη μιας χρονικής μεταβλητής στο γράφο, δηλαδή στα άρθρα που παίρνουμε να κρατάμε την μέρα συγγραφής τους σαν δεδομένο, με αποτέλεσμα να δίνεται η δυνατότητα στο χρήστη να μην αλλάζει μόνο τα βάρη του γράφου, αλλά να μπορεί να εστιάσει και σε συγκεκριμένες χρονικές περιόδους. Αυτή η επέκταση, είναι πιθανό να υποδείξει κάποιες τάσεις ή επαναλήψεις θεματολογίας της αρθρογραφίας ανά χρονικές περιόδους. Αντίστοιχα, εκτός της μεταβλητής του χρόνου, θα μπορούσε να ερευνηθεί και η πιθανή αξιοποίηση των retweets. Στη παρούσα εργασία, τα retweets (αναδημοσιεύσεις tweets) αγνοούνται, αλλά όπως επισημάναμε στην προηγούμενη ενότητα, η γενική τάση του Twitter δείχνει ότι οι χρήστες του κάνουν περισσότερα retweets παρά νέα tweets, όποτε πιθανά να ήταν σωστή η ενσωμάτωσή τους, με κάποιο παραμετρικό βάρος στο γράφο.

Τέλος, θα μπορούσε η εφαρμογή να οδηγηθεί και σε μια πιο ποιοτική αλλαγή, προσθέτοντας έννοιες/παραμέτρους συναισθήματος. Η ανάλυση συναισθήματος (sentiment analysis), εν συντομία είναι η αναγνώριση συναισθήματος μέσα σε ένα κείμενο/tweet, να μπορεί το σύστημα πρακτικά να διαχωρίσει αν το ύφος του κειμένου είναι θετικό ή αρνητικό.[33] Ειδικά, για το Twitter έχουν γίνει ήδη διάφορες αναλύσεις και εφαρμογές πάνω στην ανάλυση συναισθήματος. [34] Μια ιδέα με βάση την οποία θα μπορούσε να κινηθεί το σύστημα προς αυτήν την κατεύθυνση, διατηρώντας την υπάρχουσα δομή του, είναι να εντοπίσουμε τις λέξεις που θεωρούμε ότι προσδίδουν συναίσθημα και να εστιάσουμε σε αυτούς τους κόμβους και να δούμε τα βάρη με τα οποία συνδέονται με τους υπόλοιπους κόμβους. Έτσι , συμψηφίζοντας τα βάρη των «θετικών» με των «αρνητικών» , αν καταλήξουμε σε θετικό αποτέλεσμα το γενικό ύφος του κειμένου είναι θετικό, διαφορετικά αρνητικό.

Βιβλιογραφία

[1]Abraham Silberschatz, Henry F. Korth, S. Sudarshan, “Database System Concepts - 6th Edition”, McGraw-Hill January 28 2010
[2] Jonas Partner, Aleksa Vukotic, Nicki Watt, “Neo4j in Action”, 28 October 2014,Early Access : June 2012.
[3] The Definitive Guide to Graph Databases for the RDBMS Developer by Michael Hunger,Ryan Boyd and William Lyon
[4] Ian Robinson, Jim Webber, Emil Eifrem, “Graph Databases”, O'Reilly Media,June 2013.
[5] https://neo4j.com/developer/cypher-query-language/
[**6**] JavaScript: The Definitive Guide, 6th Edition - O'Reilly Media, April 2011.
[7]https://nodejs.org/en/
[8] www.npmjs.org
[9] Mike Cantelon, Marc Harter, T. J. Holowaychuk, Nathan Rajlich, “Node.js in Action”, Manning, October 2013
[10] Pedro Teixeira, “Professional Node.js: Building JavaScript Based Scalable Software”, Wiley, October 23 2012.
[11] Seth A. Myers, Aneesh Sharma, Pankaj Gupta, Jimmy Lin, “Information Network or Social Network? The Structure of the Twitter Follow Graph”, April 2014
[12] Seth A. Myers, Jure Leskovec, “The Bursty Dynamics of the Twitter Information Network”, 11 March 2014
[13] https://dev.twitter.com/overview/api
[14] Fred Morstatter, Jurgen Pfeffer, Huan Liu, Kathleen M. Carley, “Is the Sample Good Enough? Comparing Data from Twitter's Streaming API with Twitter's Firehose”,2013.
[15] Fred Morstatter, Jurgen Pfeffer, Huan Liu, “When is it Biased? Assessing the Representativeness of Twitter's Streaming API”, January 2014
[16] Duffy, P. D., & Bruns, A. (2006). The Use of Blogs, Wikis and RSS in Education: A Conversation of Possibilities. Learning on the move a university for the real world (pp. 31-38). Queensland University of Technology.
[17] Anderson, P. (2007). What is Web 2.0? Ideas, technologies and implications for education. (M. Hepworth, B. Kelly, R. Metcalfe, & L. Phipps, Eds.)Technology, Feb(February), 1-64. JISC
[18] http://www.regular-expressions.info/
[19] https://firebase.google.com/
[20] Automatic Summarization By Ani Nenkova and Kathleen McKeown
[21] https://github.com/xtsimpouris/gr-nlp-law/blob/master/Greek%20Stopwords/stopwords.txt
[22] Rada Mihalcea and Paul Tarau , “TextRank: Bringing Order into Texts”
[23] http://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html
[24] http://groups.engin.umd.umich.edu/CIS/course.des/cis400/html/html.html
[25] https://developer.mozilla.org/en-US/docs/Web/CSS
[26] https://jquery.com/
[27] http://sigmajs.org/
[28] https://d3js.org/
[29] http://linkurio.us/
[30] https://msdn.microsoft.com/en-us/library/ee658094.aspx

[31]http://www.itu.int/en/ITU-T/ipr/Pages/open.aspx
[32] https://linkurio.us/panama-papers-how-linkurious-enables-icij-to-investigate-the-massive-mossack-fonseca-leaks/
[33] https://lct-master.org/files/MullenSentimentCourseSlides.pdf
[34]A Pal and P Paroubek. (2010), Twitter as a Corpus for Sentiment Analysis and Opinion Mining