ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ & ΥΠΗΡΕΣΙΕΣ

ΚΑΤΕΥΘΥΝΣΗ ΔΙΚΤΥΟΚΕΝΤΡΙΚΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΕΞΑΓΩΓΗ ΧΩΡΟΧΡΟΝΙΚΩΝ ΔΕΔΟΜΕΝΩΝ ΑΠΟ ΤΟ TWITTER

Μεταπτυχιακή Διπλωματική Εργασία

Σίδερης Νικόλαος

ΜΕ/ 11067

Επιβλέπων: Δουλκερίδης Χρήστος

Λέκτορας Πανεπιστημίου Πειραιώς

Πειραιάς, Σεπτέμβριος 2015

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Δουλκερίδη Χρήστο για την πολύτιμη βοήθεια και καθοδήγηση του καθ' όλη τη διάρκεια εκπόνησης της διπλωμάτης αυτής. Επίσης οφείλω ένα μεγάλο ευχαριστώ στην οικογένεια μου για την υποστήριξη που μου παρείχαν όλα αυτά τα χρόνια.

ΕΞΑΓΩΓΗ ΧΩΡΟΧΡΟΝΙΚΩΝ ΔΕΔΟΜΕΝΩΝ ΑΠΟ ΤΟ TWITTER

Σημαντικοί Όροι: Twitter, incident detection and tracking, information retrieval

Περίληψη

Πως μπορούν οι πληροφορίες που διατίθενται στο Twitter να αξιοποιηθούν ώστε να εξάγουμε σημαντικά δεδομένα που σχετίζονται με μια συγκεκριμένη γεωγραφική περιοχή; Σε αυτή την εργασία αυτή η ερώτηση θα διερευνηθεί και θα παρουσιαστεί μια στρατηγική για τη συλλογή μηνυμάτων Tweets τα οποία εμπεριέχουν τη γεωγραφική θέση δημιουργίας τους και βρίσκονται εντός μιας συγκεκριμένης γεωγραφικής περιοχής. Πρώτον, προτείνεται αρχιτεκτονική καταγραφής των μηνυμάτων Tweets πραγματικού χρόνου χρησιμοποιώντας το Twitter streaming API και η δημιουργία ενός inverted index με τη χρησιμοποίηση της βιβλιοθήκης του Apache Lucene. Επιπλέον κατά τη διάρκεια της αναζήτησης στο Index χρησιμοποιούνται τεχνικές αναγνώρισης συναισθήματος για την περαιτέρω ανάλυση των αποτελεσμάτων. Τέλος, αξιολογούμε την απόδοση όλου του συστήματος σε σχέση με την αποτελεσματικότητα της αρχιτεκτονικής που ακολουθήθηκε, για την καταγραφή και ανάλυση των Tweets, καθώς και την ορθότητα των αποτελεσμάτων που επιστρέφει το σύστημα.

SPATIOTEMPORAL DATA MANAGEMENT FROM TWITTER

Keywords: Twitter, incident detection and tracking, information retrieval

Abstract

How can information available on Twitter exploited to extract important information occurring in a specific geographical area? In this paper this question will be explored and a strategy will presented to identify tweets which contain their geographical position within a specific geographical area. First, real-time recording of tweets architecture is proposed using Twitter streaming API and creating an inverted index using the library Apache Lucene. In addition during search from the Index recognition techniques are used to further analyze the results of the search. Finally we evaluate the performance of the entire system related to the correctness of the architecture adopted for recording and analysis of the tweets and on the correctness of the results returned by the system.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1: Εισαγωγή

Με την έλευση του Internet όλο και περισσότεροι άνθρωποι είναι συνεχώς δικτυωμένοι σε αυτό. Δημοφιλείς υπηρεσίες, όπως το Facebook και το Twitter, προσελκύουν πολλούς χρήστες, οι οποίοι μοιράζονται με τους φίλους τους το τι συμβαίνει στην καθημερινότητά τους. Ένας μέσος χρήστης του Twitter τροφοδοτεί το δίκτυο με μηνύματα πολλών και διαφορετικών μεταξύ τους θεματικών κατηγοριών. Τα δεδομένα αυτά, τα οποία ονομάζονται και user-generated, αυξάνονται ραγδαία στο παγκόσμιο ιστό και λόγω του δημόσιου χαρακτήρα τους παρουσιάζονται ακόμη και στα αποτελέσματα των μηχανών αναζήτησης, όπως το Google και το Bing.

Η χρησιμοποίηση έξυπνων κινητών συσκευών, καθώς και η δυνατότητα που δίνεται σε έναν χρήστη μέσω της εφαρμογής του Twitter να μοιραστεί τις σκέψεις του σε συνδυασμό με την ακριβή γεωγραφική του θέση, αναμένεται να οδηγήσει σε ραγδαία αύξηση των δεδομένων τα όποια είναι συνδεδεμένα μαζί με μια γεωγραφική περιοχή. Από την πλευρά των γεωγραφικών συστημάτων πληροφοριών αυτή η δυνατότητα έχει προσελκύσει την προσοχή των ερευνητών γεωγραφικών δεδομένων. Η τάση αυτή των location based μηνυμάτων τα οποία είναι διαθέσιμα δημόσια στο δίκτυο υπόσχεται ένα αχαρτογράφητο και κερδοφόρο περιβάλλον όπου μπορεί εύκολα να διεξαχθούν διάφορες κοινωνικό - γεωγραφικές αναλύσεις, χρησιμοποιώντας τις σκέψεις, εμπειρίες ή ακόμη και τα συναισθήματα ενός μεγάλου αριθμού ανθρώπων.

Η παρούσα διπλωματική εργασία εστιάζει στη συλλογή δεδομένων tweets, τα οποία είναι geo-tagged και βρίσκονται εντός μιας συγκεκριμένης γεωγραφικής περιοχής. Η μετέπειτα αποθήκευση και επεξεργασία αυτών εφαρμόζεται χρησιμοποιώντας τη βιβλιοθήκη Lucene. Σκοπός της εργασίας είναι η δημιουργία ενός λειτουργικού συστήματος που θα έχει τη δυνατότητα να εντοπίζει τα tweets που περιέχουν τις λέξεις κλειδιά της επερώτησης που έχει τεθεί, καθώς και τη δυνατότητα να αναλύει και να παρουσιάζει το συνολικό συναίσθημα (αρνητικό, θετικό) των αποτελεσμάτων της αναζήτησης σε πραγματικό χρόνο.

Αναγνώριση προβλήματος

Το Twitter, το οποίο είναι μια micro-blogging υπηρεσία, επιτρέπει στους χρήστες του να δημοσιοποιούν μηνύματα με μέγιστο μέγεθος 140 χαρακτήρων (tweets), αλλά και να ακολουθούν (follow) τα tweets των χρηστών με τους οποίους είναι συνδεδεμένοι. Αυτή τη στιγμή σχεδόν πεντακόσια εκατομμύρια tweets δημιουργούνται κάθε μέρα [14] . Από το 2010 το twitter πρόσφερε τη δυνατότητα της δημοσιοποίησης ενός μηνύματος tweet μαζί με τη γεωγραφική θέση του χρήστη, δίνοντάς του την επιλογή αν ήθελε να εμφανίζεται το γεωγραφικό στίγμα του κάθε φορά που θα δημιουργεί μια. Στο Twitter δε χρησιμοποιείται το σύστημα των check-in, όπως χρησιμοποιείται σε άλλες υπηρεσίες, όπως το Facebook, αλλά η χωροχρονική πληροφορία η όποια προσαρτάται μαζί με το tweet, και παρέχει ένα επιπλέον εννοιολογικό περιεχόμενο. Για παράδειγμα με ένα check-in στο Facebook σε ένα σημείο της Αθήνας μπορούμε να ξέρουμε πως ένας χρήστης πιθανόν να πίνει καφέ εκεί. Ένα tweet όμως με κείμενο <<Έχει τρομερή κίνηση σήμερα>> που αναρτάται σε ένα ύψος του ίδιου δρόμου μας δίνει την επιπλέον πληροφορία της κατάστασης (<<τρομερή κίνηση>>) που επικρατεί στο σημείο αυτό.
Εκμεταλλευόμενοι αυτό το επιπλέον εννοιολογικό περιεχόμενο που προσφέρεται από το Twitter προσπαθούμε να απαντήσουμε στο ερώτημα, ποιο είναι το συναίσθημα ενός συνόλου ανθρώπων, για ένα συγκεκριμένο θέμα, σε μια συγκεκριμένη γεωγραφική περιοχή, σε πραγματικό χρόνο. Για παράδειγμα σε ένα σημαντικό τοπικό γεγονός, όπως ένας μεγάλος αγώνας ποδοσφαίρου, χρησιμοποιώντας το σύστημα που θα προταθεί, θα γνωρίζουμε θέτοντας ως λέξη κλειδί το όνομα μιας ομάδας αν η άποψη του κοινού είναι αρνητική ή θετική σε πραγματικό χρόνο, εξετάζοντας έτσι και τις διακυμάνσεις στο συναίσθημα του κοινού.

Ερευνητικοί στόχοι

- **Στατιστική απεικόνιση σημαντικών πληροφοριών μέσω γραφικών παραστάσεων.**
  Το σύστημα το οποίο προτείνεται κατά τη διαδικασία της καταγραφής των μηνυμάτων Tweets, από το Streaming API του twitter, εξάγει συγκεκριμένες

πληροφορίες από ένα tweet (Hashtags, Language) το οποίο έχει σταλεί από μια συγκεκριμένη γεωγραφική περιοχή. Οι πληροφορίες αυτές παρουσιάζονται στον χρηστή μέσω γραφικών παραστάσεων και απεικονίζουν τα δέκα πιο διαδεδομένα Hashtags, καθώς και τις δέκα πιο δεδομένες γλώσσες που έχουν χρησιμοποιηθεί για τη σύνταξη των tweets.

- **Εφαρμογή sentiment analysis και γραφική απεικόνιση των αποτελεσμάτων στα αποτελέσματα της αναζήτησης.**
  Κατά τη διαδικασία της αναζήτησης το σύστημα, αφού εντοπίσει τα tweets που περιέχουν τις λέξεις - κλειδιά της επερώτησης που έχει τεθεί, εφαρμόζει τεχνικές ανάλυσης συναισθήματος χρησιμοποιώντας τα API των sentiment140 και UClassify. Τα API αυτά επιστρέφουν τιμές οι οποίες δείχνουν το πόσο αρνητικό ή θετικό είναι το συναίσθημα των συνολικών tweets που επιστράφηκαν από την αναζήτηση. Στο τέλος τα αποτελέσματα αυτών απεικονίζονται σε γραφική παράσταση.

- **Απαίτηση όλα τα παραπάνω να συμβαίνουν σε πραγματικό χρόνο.**
  Ένα από τα κυρίως στοιχεία αυτής της εφαρμογής είναι όλες οι διαδικασίες, όπως συλλογή tweets από το Twitter, ανάλυση, δημιουργία Index, αναζήτηση και sentiment analysis να συμβαίνουν σε πραγματικό χρόνο. Έτσι προτείνεται ένα σύστημα που η αρχιτεκτονική του επιτρέπει να συμβαίνουν όλα τα παραπάνω.

Δομή παρούσας εργασίας

Η παρούσα εργασία αποτελείται από οκτώ κεφάλαια τα οποία ταξινομούνται ως εξής:

- **Εισαγωγή** : Το κεφάλαιο ένα παρουσιάζει αρχικά την ερευνητική μεθοδολογία που θα ακολουθηθεί, καθώς και την αναγνώριση του προβλήματος. Αυτό το κεφάλαιο επιπλέον περιέχει και τους ερευνητικούς στόχους της παρούσας εργασίας.

- **Twitter** : Το κεφάλαιο δύο εστιάζει στην ανάλυση της πλατφόρμας της διαδικτυακής εφαρμογής Twitter, η οποία είναι και η πηγή δεδομένων που θα χρησιμοποιηθεί.
- **Lucene** : Το κεφάλαιο τρία περιέχει την παρουσίαση της βιβλιοθήκης Lucene που χρησιμοποιείται στη εργασία αυτή για το indexing, αλλά και την αναζήτηση των καταγραμμένων tweets.
- **Σχετικές μελέτες** : Το κεφάλαιο τέσσερα εξετάζει σχετικές μελέτες που χρησιμοποιούν το Twitter και τη βιβλιοθήκη Lucene για την ανάλυση των δεδομένων.
- **Αρχιτεκτονική συστήματος:** Το πέμπτο κεφάλαιο αναλύει την αρχιτεκτονική του συστήματος.
- **Οδηγός χρήσης εφαρμογής :** Στο έκτο κεφάλαιο παρουσιάζεται ο οδηγός χρήσης της εφαρμογής.
- **Πειραματικές μετρήσεις:** Το έβδομο κεφάλαιο παρουσιάζει τα αποτελέσματα της έρευνας του προτεινόμενου συστήματος, καθώς και την αξιολόγηση αυτών.
- **Συμπεράσματα :**Το κεφάλαιο οκτώ παρέχει απαντήσεις σχετικά με το πρόβλημα, επιπλέον παρουσιάζονται γενικά επιτεύγματα της παρούσας εργασίας και προτάσεις για μελλοντικές εργασίες.

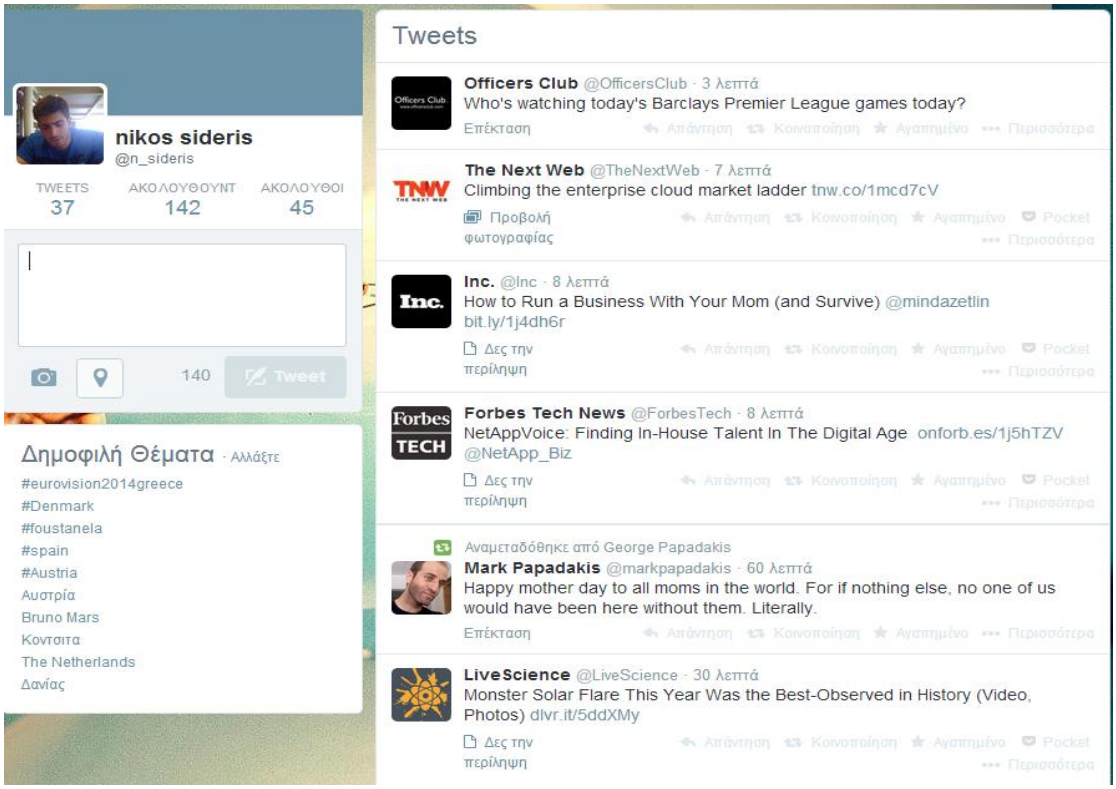
Κεφάλαιο 2: Twitter

Το Twitter είναι ένα online κοινωνικό δίκτυο και μια υπηρεσία που επιτρέπει στους χρήστες να στέλνουν και να διαβάζουν σύντομα μηνύματα κειμένου έως 140 χαρακτήρων, που ονομάζονται tweets. Η πρόσβαση στο Twitter διατίθεται μέσω της ιστοσελίδας του ή το mobile application μιας κινητής συσκευής, μέσω των οποίων οι εγγεγραμμένοι χρήστες μπορούν να διαβάσουν και να στείλουν tweets.

Για τη διατύπωση των παρακάτω θέσεων έχουν χρησιμοποιηθεί πληροφορίες από το www.twitter.com.

Η πλατφόρμα του Twitter αποτελείται από αντικείμενα. Αυτά διακρίνονται σε :

- Tweets
- Users
- Entities
- Places



Εικόνα 1 User profile στοTwitter

Tweets

Τα tweets είναι η βασική μονάδα γύρω από την οποία έχει κατασκευαστεί η πλατφόρμα του Twitter. Χρησιμοποιώντας τα tweets ο χρήστης δημιουργεί ενημερώσεις (status updates). Επιπλέον, τα tweets μπορούν να διαγραφούν, να απαντηθούν, να επισημανθούν σαν αγαπημένα και το αντίθετο.

Users

Οι χρήστες είναι άνθρωποι ή τεχνητής νοημοσύνης αντικείμενα (Robots) οι οποίοι δημιουργούν tweets, μπορούν να ακολουθήσουν άλλους χρήστες (Follow), έχουν το δικό τους home timeline, στο οποίο διακρίνονται τα δικά τους ή άλλα κοινοποιημένα tweets από τους ίδιους, μπορούν να αναφερθούν από άλλους χρήστες και τελικώς μπορούν να εξευρεθούν στην αναζήτηση.

Entities

Οι οντότητες (Entities) βρίσκονται ενσωματωμένες στα tweets. Περιέχουν μεταδεδομένα (metadata), δηλαδή δεδομένα τα οποία περιγράφουν άλλα δεδομένα τα οποία εμπεριέχονται στο twitter και διακρίνονται σε:

1. Hashtags : Λέξεις - κλειδιά που στιγματίζουν ένα tweet. Αποτελούνται από μία λέξη ή φράση και αρχίζουν με ένα "#", χωρίς κενά ή σημεία στίξης.
2. Media : Αντιπροσωπεύουν στοιχεία πολυμέσων, όπως εικόνες και βίντεο.
3. Urls : Αντιπροσωπεύουν URL που περιλαμβάνονται στο κείμενο ενός Tweet.
4. User_mentions : Αντιπροσωπεύουν άλλους χρήστες που αναφέρονται σε ένα tweet κάποιου άλλου user.

Places

Οι τοποθεσίες καταγραφούν γεωγραφικές θέσεις σε ένα tweet. Είναι μεταδεδομένα και περιέχουν στοιχεία όπως πόλη, κωδικό, χώρα, ταχυδρομικό κώδικα, τηλέφωνο, κλπ. Συνήθως καταγράφουν τη θέση από την οποία προήλθε ένα tweet, αλλά αυτό δε είναι πάντα αληθές, αφού ο χρήστης μπορεί να καθορίσει μόνος του την τοποθεσία που θα συνδέσει με ένα tweet καθορίζοντας ένα place_id. Οι πληροφορίες θέσης στο Twitter είναι διαθέσιμες από δύο πηγές :

• **Πληροφορίες γεωγραφικής θέσης**: Οι χρήστες μπορούν προαιρετικά να επιλέξουν να παρέχουν πληροφορίες για την τοποθεσία των Tweets που δημοσιεύουν. Αυτή η πληροφορία μπορεί να είναι ιδιαίτερα ακριβής αν το Tweet δημοσιεύθηκε χρησιμοποιώντας ένα έξυπνο κινητό με δυνατότητες GPS.

• **Προφίλ του χρήστη** : Η τοποθεσία χρήστη μπορεί να εξαχθεί από το πεδίο, τοποθεσία στο προφίλ του χρήστη.

Το Twitter API

Το API του twitter μπορεί να διαχωριστεί σε 3 κομμάτια. Τα 2 από αυτά αποτελούν τα REST APIs και το υπόλοιπο το Streaming API. Παλιότερα τα rest APIs δε παρέχονταν ενιαία από το Twitter, καθώς το search API το διαχειριζόταν μια τρίτη εταιρία. Μετά την εξαγορά της από το twitter όμως, στον τελικό χρήστη παρέχεται μια πλήρης διασύνδεση μεταξύ αυτών των API. Το streaming API είναι διαφορετικό από τα άλλα 2, διότι υποστηρίζει long-lived connections και η αρχιτεκτονική του είναι διαφορετική.

Rate Limiting

Το API του twitter επιτρέπει στους developers να έχουν πρόσβαση στα δεδομένα σύμφωνα με ένα περιορισμένο αριθμό κλήσεων. Στην έκδοση 1.1 η οποία μελετάται σε αυτή την εργασία όλες οι κλήσεις χρειάζονται πιστοποίηση

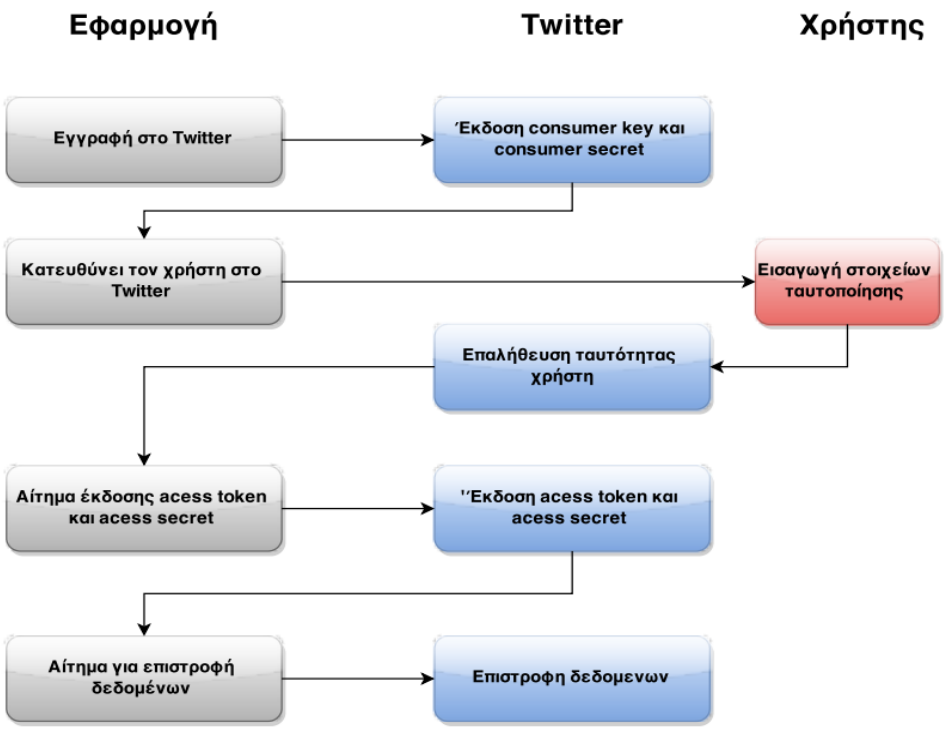
(authentication). Για την πιστοποίηση χρησιμοποιείται το Oauth το οποίο σύμφωνα με το www.oauth.net είναι :

«Ένα ανοιχτό πρωτόκολλο το οποίο επιτρέπει την ασφαλή χορήγηση άδειας (authorization) χρησιμοποιώντας μια απλή και τυποποιημένη μέθοδο από το διαδίκτυο, τα κινητά και τις desktop εφαρμογές ».

Εισαγωγή στο Open Authentication (OAuth)

Το open Authentication (OAuth) είναι ένα ανοιχτό πρότυπο για έλεγχο ταυτότητας, το οποίο εγκρίθηκε από το Twitter για να παρέχει πρόσβαση σε ιδιωτικές πληροφορίες ενός χρήστη. Οι κωδικοί πρόσβασης είναι ευάλωτοι σε κλοπή και το OAuth παρέχει μια ασφαλέστερη εναλλακτική λύση στις παραδοσιακές προσεγγίσεις πιστοποίησης ταυτότητας με τη χρήση χειραψίας τριών δρόμων . Βελτιώνει, επίσης, και την εμπιστοσύνη του χρήστη στην εφαρμογή , αφού ο κωδικός πρόσβασης του χρήστη ποτέ δεν μοιράζεται με εφαρμογές τρίτων. Παρακάτω παρουσιάζονται λεπτομερειακά τα βήματα για την αποστολή μιας κλήσης στοTwitter API χρησιμοποιώντας το OAuth :

1. Όλες οι εφαρμογές υποχρεούνται να εγγραφούν στο Twitter. Μέσω αυτής της διαδικασίας παρέχετε στην εφαρμογή ένα κλειδί (consumer key) και το μυστικό (consumer secret) που η εφαρμογή πρέπει να χρησιμοποιήσει για να πιστοποιηθεί στο Twitter.
2. Η εφαρμογή χρησιμοποιεί το κλειδί και το μυστικό για να δημιουργήσει μια μοναδική σύνδεση με το Twitter με την οποία ένας χρήστης κατευθύνεται για έλεγχο ταυτότητας. Ο χρήστης εξουσιοδοτεί την εφαρμογή ταυτοποιώντας τον εαυτό του στο Twitter. Το Twitter επαληθεύει την ταυτότητα του χρήστη και εκδίδει έναν κωδικό PIN.
3. Ο χρήστης παρέχει αυτό το PIN στην εφαρμογή. Η εφαρμογή χρησιμοποιεί το PIN για να ζητήσει ένα " Access Token " και " Access Secret " μοναδικά για το χρήστη.



Εικόνα 2 Διάγραμμα ροής OAUTH

API v1.1's Authentication Model

Η αυθεντικοποίηση ενός χρήστη μπορεί να γίνει με δυο μεθόδους.

- Εφαρμογή-Χρήστη αυθεντικοποίηση (Application-user authentication)
- Αυθεντικοποίηση εφαρμογής (Application-only authentication)

Στην πρώτη περίπτωση το υπογεγραμμένο request που στέλνει ένας χρήστης περιέχει και την ταυτότητα της εφαρμογής, αλλά και την ταυτότητα του χρήστη, ενώ στη δεύτερη περίπτωση το υπογεγραμμένο request περιέχει μόνο την ταυτότητα της εφαρμογής. Οι διαφορετικές αυτές κλήσεις αποσκοπούν στην αυθεντικοποίηση της εφαρμογής και του χρήστη, ώστε να εφαρμοστεί το rate limiting.

Για να δημιουργηθεί ένα HTTPrequest στο twitter χρειάζεται να δοθούν key/values δεδομένα από την πλατφόρμα του Twitter. Αυτά είναι:

1. **Consumer key** : Αναγνωριστικό για το ποια εφαρμογή κάνει το request.
2. **Nonce** : Είναι μια μεταβλητή την όποια θα πρέπει να αναπαράγει η εφαρμογή για κάθε http request. Το twitter χρησιμοποιεί αυτή την τιμή, ώστε να αναγνωρίσει εάν το συγκεκριμένο request έχει υποβληθεί πολλαπλές φορές.
3. **Signature :** Περιέχει μια τιμή η οποία είναι το αποτέλεσμα ενός αλγορίθμου που έχει ως είσοδο όλες τις παραμέτρους μαζί με δύο μυστικές τιμές.
4. **Signature method :** Είναι ο αλγόριθμος HMAC-SHA1
5. **Timestamp :** Δηλώνει τη χρονική περίοδο που δημιουργήθηκε το request
6. **Token :** Περιέχει μια τιμή η οποία επιτρέπει την πρόσβαση της εφαρμογής σε ένα συγκεκριμένο twitter account.
7. **Version** : Default τιμή 1.0

Παράδειγμα HTTP POST request

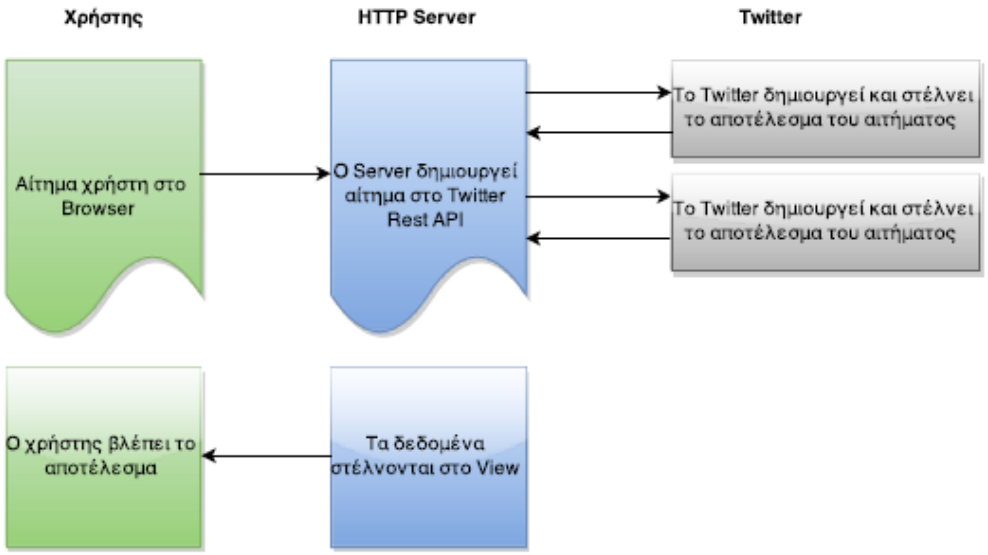
```
POST /1/statuses/update.json?include_entities=true HTTP/1.1
Accept: */*
Connection: close
User-Agent: OAuth gem v0.4.4
Content-Type: application/x-www-form-urlencoded
Authorization:
      OAuth oauth_consumer_key="xvz1evFS4wEEPTGEFPHBog",
           oauth_nonce="kYjzVBB8Y0ZFabxSWbWovY3uYSQ2pTgmZeNu2VS4cg",
           oauth_signature="tnnArxj06cWHq44gCs1OSKk%2FjLY%3D",
           oauth_signature_method="HMAC-SHA1",
           oauth_timestamp="1318622958",
           oauth_token="370773112-
GmHxMAgYyLbNEtIKZeRNFsMKPR9EyMZeS9weJAEb",
           oauth_version="1.0"
Content-Length: 76
```

Host: api.twitter.com

status=Hello%20Ladies%20%2b%20Gentlemen%2c%20a%20signed%20OAuth%20request%21

Το Rest API

Οι μέθοδοι του Rest API επιτρέπουν στους προγραμματιστές να έχουν πρόσβαση σε βασικά δεδομένα του Twitter. Αυτά περιλαμβάνουν πληροφορίες για το χρήστη, ενημερώσεις κτλ. Το μεγαλύτερο αρνητικό για τη χρήση του είναι ο χρονικός περιορισμός που υπόκεινται οι μέθοδοί του. Στο παρακάτω διάγραμμα παρουσιάζεται η αρχιτεκτονική λειτουργιάς του Rest API.



Εικόνα 3 Twitter rest API αρχιτεκτονική

Στην παρούσα διπλωματική δε θα χρησιμοποιηθεί το Rest API,καθώς ο χρονικός περιορισμός και το όριο κλήσεων σε μια μέθοδο μας αποτρέπουν από την ανάλυση των δεδομένων που χρειαζόμαστε.

Τα Streaming APIs

Αντιθέτως από το Rest API το Streaming API προσφέρει μια συνεχή ανοιχτή σύνδεση με την πλατφόρμα του twitter παρέχοντας έτσι μια ζωντανή ροή των tweets. Υπάρχουν τριών ειδών streams, αυτά είναι :

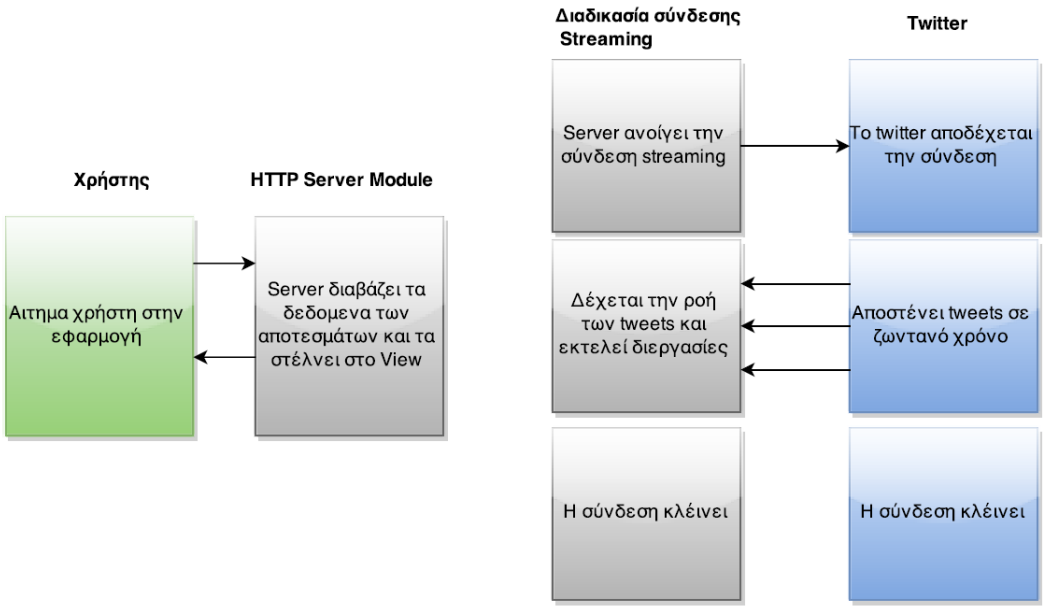
- Public Streams
- User Streams
- Site Streams

Τα Public Streams συμπεριλαμβάνουν όλα τα tweets τα οποία είναι διαθέσιμα στο twitter, μια υποθετική χρήση τους είναι το Data Mining.

Τα User Streams συμπεριλαμβάνουν σχεδόν όλα τα στοιχεία που διαθέτει το twitter για κάποιον συγκεκριμένο πιστοποιημένο user.

Τελευταία τα Site Streams προσφέρουν ότι και τα public streams με τη διαφορά ότι μπορούν να λαμβάνουν ζωντανά δεδομένα από το twitter για μεγάλο αριθμό χρηστών.

Όπως φαίνεται και στην παρακάτω εικόνα, μια εφαρμογή για να συνδεθεί στο Streaming API του twitter χρειάζεται 2 διαφορετικά μέρη. Ένα μέρος της εφαρμογής θα πρέπει να διατηρεί μια ανοιχτή σύνδεση στους servers του twitter, έτσι ώστε να λαμβάνει, να φιλτράρει, αλλά και να αποθηκεύει τα αποτελέσματα, ενώ ένα άλλο μέρος της εφαρμογής να δέχεται τα αιτήματα των χρηστών και από τα είδη αποθηκευμένα δεδομένα να ανακτά τις πληροφορίες και να τις αποστέλλει στον χρήστη.



Εικόνα4 Twitter streaming API αρχιτεκτονική

Public Streams

Τα public streams μας δίνουν πρόσβαση στη ζωντανή ροή δεδομένων του twitter φιλτράροντάς τα σύμφωνα με παραμέτρους . Ένα θετικό χαρακτηριστικό είναι ότι δεν περιοριζόμαστε πλέον από rate limiting, όπως ισχύει στο rest api.

Endpoints

Για να συνδεθούμε με το Streaming API θα πρέπει να καλέσουμε μερικά από τα παρακάτω endpoints σχηματίζοντας ένα HTTP αίτημα.

1. POST statuses/filter

POST https://*stream.twitter.com/1.1/statuses/filter.json*

Επιστρέφει μία ζωντανή ροή δεδομένων από public statuses, τα οποία ταιριάζουν σε μία ή περισσότερες παραμέτρους. Οι παράμετροι που δέχεται είναι :

- **Follow**: Καθορίζονται οι χρήστες από τους οποίους λαμβάνονται δεδομένα. Ως παράμετρο εισάγεται μια λίστα από user ids.
- **Track**: Για αναζήτηση keywords ή φράσεων. Ως παράμετρο δέχεται μια λίστα από φράσεις ή λέξεις. Το φιλτράρισμα γίνεται στο πεδίο text του tweet, αλλά και σε μερικά entities πεδία όπως display_ur, text και hashtags.
- **Locations**: Καθορίζεται το γεωγραφικό μήκος και πλάτος της περιοχής. Το twitter χρησιμοποιεί Bounding Box (π.χ -180,-90,180,90), ώστε να γίνει η αναζήτηση.
- **Delimited**: Αν θέλουμε τα μηνύματα να περιορίζονται λόγω μεγέθους
- **Stall_warnings**: Είναι μηνύματα που στέλνονται σε περίπτωση κινδύνου διακοπής της σύνδεσης με τους servers του twitter.

2. *GET statuses/sample*

Επιστρέφει ένα μικρό τυχαίο δείγμα από όλα τα public statuses. Ως παραμέτρους δέχεται delimited και stall_warnings

3. *GET statuses/firehose*

Το firehose επιστρέφει όλα τα public statuses του twitter. Η πρόσβαση σε αυτό επιτρέπεται μόνο σε εταιρίες - συνεργάτες του twitter.

Κεφάλαιο 3 : Lucene

Εισαγωγή

Το twitter είναι ένα τεράστιο κοινωνικό δίκτυο από το οποίο έχουν σταλεί δισεκατομμύρια μηνύματα στα χρόνια λειτουργίας του, και δίνει τη δυνατότητα μέσω του API που διαθέτει την πρόσβαση σε αυτά τα tweets. Θα μπορούσαμε να κατηγοριοποιήσουμε τα δεδομένα αυτά σε μια σχεσιακή βάση δεδομένων, ωστόσο η ανάκτηση αυτής της πληροφορίας μέσω εκατοντάδων ή χιλιάδων κατηγοριών και υποκατηγοριών δεν είναι πλέον μια αποτελεσματική μέθοδος. Το ζητούμενο στην σημερινή εποχή είναι να μπορούμε να κάνουμε αναζητήσεις δεδομένων με τον ευκολότερο δυνατό τρόπο, και την ακριβή εύρεση της πληροφορίας που χρειαζόμαστε στο μικρότερο δυνατό χρόνο.

Ένα καλό παράδειγμα είναι οι αναζητήσεις που μπορούμε να κάνουμε από το www.google.com. Εν συντομία για να μπορέσει η μηχανή αναζήτησης του Google να παρέχει ορθά αποτελέσματα σε μια αναζήτηση ακολουθεί διαδικασίες. Αυτές συμπεριλαμβάνουν web crawling παραπάνω από 60 τρισεκατομμυρίων ιστοσελίδων και αποθήκευση των αποτελεσμάτων σε Index. Έπειτα σύμφωνα με τα κριτήρια αναζήτησης και αφού αυτά φιλτραριστούν μέσα από τους αλγορίθμους της μηχανής αναζήτησης, επιλέγονται από το Index αρχεία τα οποία είναι σχετικά με την αναζήτηση. Στο τέλος τα αρχεία ταξινομούνται και παρουσιάζονται στον χρήστη σύμφωνα με 200 διαφορετικούς παράγοντες. [16]

Παρόμοιας λειτουργιάς με τη μηχανή αναζήτησης του Google είναι και το Lucene. Το Lucene είναι μια βιβλιοθήκη ανάκτησης πληροφοριών (information retrieval). Η βιβλιοθήκη χαρακτηρίζεται από τη σταθερότητά της, αλλά και το ότι περιέχει άδεια ανοιχτού κώδικα (open source) σύμφωνα με τη άδεια Apache Software License. Αυτή τη στιγμή το Lucene θεωρείται η πιο διάσημη βιβλιοθήκη IR και χρησιμοποιείται από τις μεγαλύτερες εταιρίες όπως : NetFlix, Digg, MySpace, LinkedIn, Fedex, Apple, SalesForce.com, theEclipseIDE [3].

Δομή του Lucene

Τα κυριότερα στοιχεία του Lucene περιγράφονται παρακάτω [9] :

IndexWriter : Η Index Writer κλάση δημιουργεί ένα ευρετήριο (index) και καθορίζει εάν ένα νέο index θα δημιουργηθεί ή εάν ένα από τα υπάρχοντα indexes θα επαναχρησιμοποιηθεί για την προσθήκη νέων εγγράφων.

Directory : Αντιπροσωπεύει τη θέση στην όποια αποθηκεύεται ένα Lucene Index.

Document : Είναι η κύρια μονάδα του Indexing και Searching. Ένα Document αποτελείται από ένα σύνολο πεδίων (fields). Ένα πεδίο μπορεί να αποθηκευτεί με το Document, και επιστρέφεται με την αναζήτηση ενός Document. Έτσι, κάθε Document πρέπει να περιέχει συνήθως ένα ή περισσότερα αποθηκευμένα πεδία που το προσδιορίζουν.

Field : Κάθε πεδίο έχει τρία μέρη: το όνομα, το type και το Value. Ο τύπος μπορεί να είναι κείμενο (String, TokenStream), δυαδικός (byte []), ή αριθμητικός (int, double). Τα πεδία αποθηκεύονται στο Index, ώστε να μπορούν να επιστραφούν στις αναζητήσεις.

Analyzer : Χρησιμοποιείται για να φιλτράρει τα δεδομένα που ευρετηριοποιούνται. Ένας analyzer δημιουργεί TokenStreams αναλύοντας το text.

IndexSearcher : Χρησιμοποιείται για την αναζήτηση από το Index.

Term : Αντιπροσωπεύει μια λέξη από το κείμενο. Αποτελείται από δύο στοιχεία, το κείμενο, σαν ένα string, και το όνομα του πεδίου στο οποίο εμφανίζεται το κείμενο.

QueryParser : Χρησιμοποιείται για την κατασκευή ενός parser που μπορεί να κάνει αναζητήσεις σε ένα Index. Το query μπορεί να είναι είτε ένας όρος, που αναζητά όλα τα έγγραφα, τα οποία περιέχουν ένα συγκεκριμένο όρο ή ένα ερώτημα που περικλείεται σε παρενθέσεις.

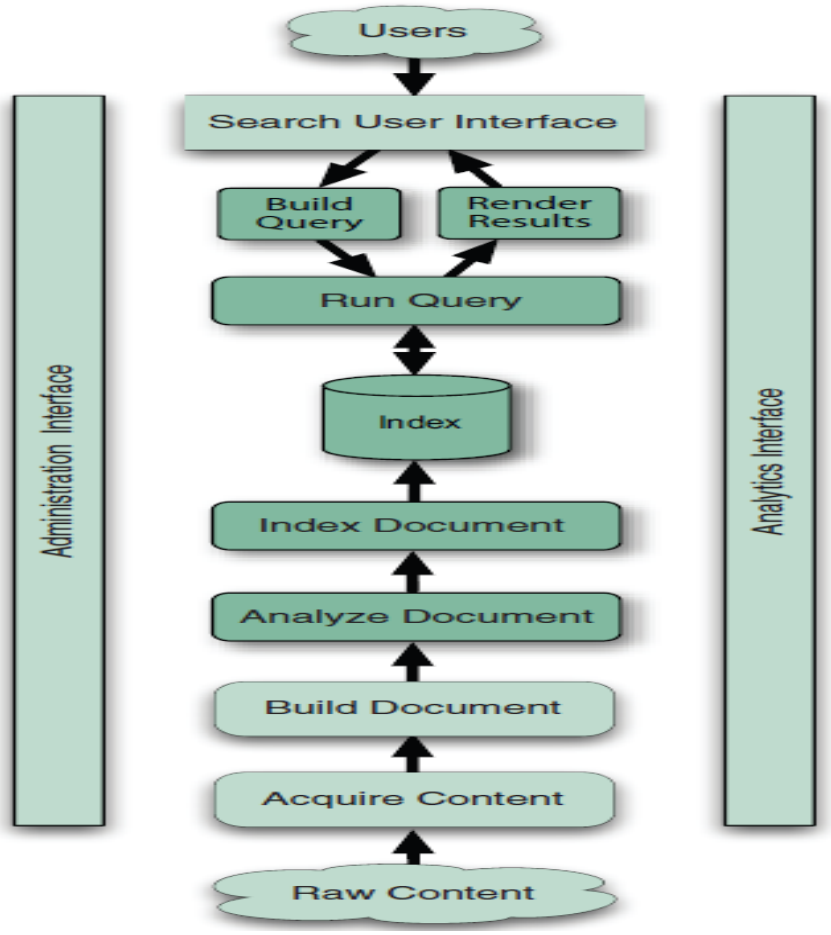
Query : Περιέχει τα κριτήρια αναζήτησης που δημιουργήθηκαν από το QueryParser.

Hits : Περιέχει τα αντικείμενα του εγγράφου που επιστρέφονται από την εκτέλεση του ερωτήματος.

Indexing

Το Lucene δεν είναι μια ολοκληρωμένη εφαρμογή αναζήτησης, αυτό που μας παρέχει είναι το core indexing και searching. Στην παρακάτω εικόνα απεικονίζονται τα τυπικά στοιχεία μιας εφαρμογής αναζήτησης. Η βιβλιοθήκη του Lucene μπορεί να μας προσφέρει τα παρακάτω στοιχεία :

1. Analyze Document
2. Index Document
3. Index
4. Run Query
5. Build Query-Render Results



Εικόνα5 Lucene αρχιτεκτονική [9]

Acquire Content

Η ανάκτηση πληροφορίας είναι η διαδικασία κατά την οποία συγκεντρώνουμε τα δεδομένα που χρειάζονται για να κάνουμε indexing. Στην παρούσα εργασία η πληροφορία ανακτήθηκε από το API του twitter σε json μορφή.

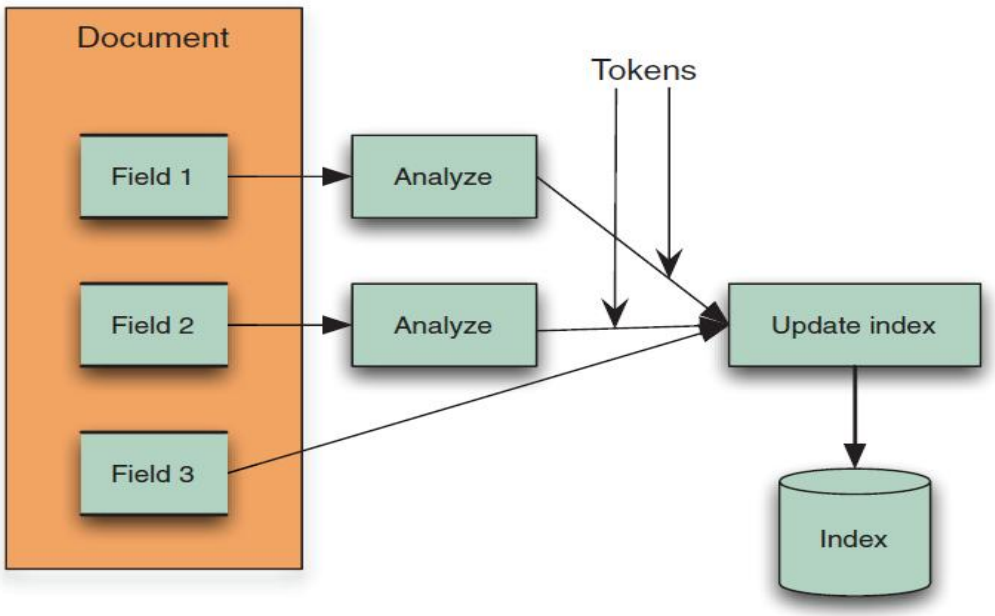
Build Document

Τα Document είναι η κύρια μονάδα του Indexing και της αναζήτησης. Για να δημιουργηθεί το Index, θα πρέπει πρώτα να μετατραπούν τα δεδομένα σε Documents και Fields, στη συνέχεια τα πεδία αυτά χρησιμοποιούνται για τις αναζητήσεις. Η λογική της εφαρμογής της εργασίας χρησιμοποίει ένα document για κάθε ξεχωριστό tweet.

Analyze Document

Κατά τη διαδικασία του Indexing εφαρμόζεται το tokenization. Tokenization είναι η διαδικασία κατά την οποία το lucene διαχωρίζει το κείμενο σε μικρά στοιχεία. Ο τρόπος που γίνεται αυτός ο διαχωρισμός σε tokens είναι στην επιλογή του προγραμματιστή χρησιμοποιώντας το κατάλληλο analyzer. Το Lucene παρέχει μια σειρά από analyzers που δίνουν ακριβή έλεγχο πάνω σε αυτή τη διαδικασία. Η εφαρμογή της παρούσας διπλωματικής χρησιμοποιεί το StandarAlyzer.

- StandarAnalyzer : Είναι ο πιο εξελιγμένος analyzer του Lucene. Έχει την ικανότητα να εντοπίσει ορισμένα είδη μαρκών, όπως ονόματα εταιρειών και e-mail. Επίσης μετατρέπει σε lowercase όλους τους χαρακτήρες και αφαιρεί stopwords και σημεία στίξης.



Εικόνα 6 Document analyzer [9]

Στο διάγραμμα φαίνεται η διαδικασία ανάλυσης κατά τη διάρκεια του indexing. Τα πεδία 1 και 2 αναλύονται, παράγοντας μία αλληλουχία από token. Το Πεδίο 3 είναι unanalyzed, προκαλώντας ολόκληρη την τιμή του να αποθηκεύεται ενιαία στο index.

Η έξοδος της διαδικασίας ανάλυσης είναι μια ροή από token τα οποία γράφονται στο Index και περιγράφουν μια λέξη από ένα text. Το token εκτός από την ίδια την λέξη εμπεριέχει και metadata, τα οποία αναφέρονται σε σημεία έναρξης και τέλους ενός χαρακτήρα στο αρχικό κείμενο, τον τύπο του π.χ. String, και ένας integer που περιγράφει την υπάρχουσα θέση του token.

Index

Σε αυτό το στάδιο τα Documents προσθέτονται στο Index. Το Lucene αποθηκεύει τα δεδομένα σε μια δομή δεδομένων γνωστή ως inverted index. Αυτή η δομή είναι πολύ αποδοτική, αφού δημιουργεί ένα mapping από τα αρχικά αρχεία. Η

δημιουργία αυτού του mapping μας επιτρέπει τη γρήγορη αναζήτηση keywords. Οι περισσότερες μηχανές αναζήτησης στο Web χρησιμοποιούν τη δομή inverted index.

Κάθε Index αποτελείται από ένα ή περισσότερα Segments. Κάθε Segment είναι ένα υποσύνολο του Index. Για παράδειγμα στην παρούσα διπλωματική για κάθε Tweet δημιουργούμε ένα Document το οποίο εισάγεται στο Index.

BuildQuery

Όταν ο χρήστης εισάγει στη μηχανή αναζήτησης ένα αίτημα αναζήτησης αυτό μεταφράζεται στην εφαρμογή σε ένα Query object, το οποίο στη βιβλιοθήκη του Lucene ονομάζεται QueryParser. Το QueryParser εσωτερικά μετατρέπει το κείμενο που εισάγει ο χρήστης σε ένα από τα παρακάτω query types:

Querytypes

- Term Query : Αναζήτηση σύμφωνα με ένα Term. Τα Term Querys είναι ιδιαίτερα χρήσιμα για την ανάκτηση εγγράφων με ένα κλειδί.
- TermRange Query : Τα Terms ταξινομούνται με αλφαβητική σειρά στο Index δίνοντάς μας τη δυνατότητα να κάνουμε αναζητήσεις όρων του κειμένου μέσα σε ένα εύρος τιμών.
- NumericRange Query : Όπως στο TermRange Query, και σε αυτό το query η αναζήτηση γίνεται μεταξύ ενός εύρους αριθμών.
- Prefix Query : Το Prefix Query ταιριάζει με Documents που περιέχουν τους όρους που αρχίζουν με ένα συγκεκριμένο string.
- Boolean Query : Μπορούμε να χρησιμοποιήσουμε το Boolean query με ένα συνδυασμό από τα query types επιτρέποντας έτσι τη χρησιμοποίηση των λογικών συντελεστών AND, OR, NOT .
- Phrase Query : Ένα Query που ταιριάζει με τα Document που περιέχουν μια συγκεκριμένη ακολουθία όρων. Εάν για παράδειγμα ο χρήστης εισάγει σε queryParse το string "new york", θα μετατραπεί εσωτερικά σε phrase Query.
- Wildcard-Query : Αναζήτηση με τη χρήση των χαρακτήρων '*' και '?' για terms στα οποία λείπουν στοιχεία.

- Fuzzy Query :Συγκρίνει όρους σύμφωνα με την ομοιότητα που έχουν μεταξύ τους. Η μέτρηση της ομοιότητας βασίζεται στον αλγόριθμο Damerau-Levenshtein .
- MatchAllDocs Query : Ένα Query που ταιριάζει όλα τα έγγραφα.

SearchQuery

Το Search Query είναι το τελικό στάδιο πριν την εμφάνιση των αποτελεσμάτων στον χρήστη. Σε αυτό το σημείο το Lucene αναζητά και επιστρέφει τα Documents τα οποία περιέχουν αποτελέσματα για μια συγκεκριμένη αναζήτηση. Το Lucene επιστρέφει δεδομένα σύμφωνα με ένα συνδυασμό από τους ακόλουθους αλγόριθμους αναζήτησης.

- **VectorSpace** : Είναι ένας αλγόριθμος ο οποίος αναπαριστά ως διανύσματα Documents και Queries.

  $$d_j = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$$
  $$q = (w_{1,q}, w_{2,q}, \ldots, w_{t,q})$$

  Κάθε διάσταση αντιστοιχεί σε ένα ξεχωριστό term. Η συνάφεια μεταξύ ενός query και ενός Document υπολογίζεται από την απόσταση ενός διανύσματος μεταξύ αυτών των διανυσμάτων.

- **StandardBooleanModel** : Έχουμε Documents D = {D1, ..., Di, ..., Dn} και Terms T = {t1, t2, ..., tj, ..., tm} οπού το Di εμπεριέχει στοιχεία του Ti. Δίνοντας ένα Query Q = (Wi OR Wk OR ...) AND ... AND (Wj OR Ws OR ...)….. βρίσκουμε αν το Di εμπεριέχει η όχι το Query.

Αρχικά το Lucene χρησιμοποιεί το BooleanModel, έτσι ώστε να ελαττώσει τα Documents που πρέπει να βαθμολογηθούν (Scored) και έπειτα χρησιμοποιεί το VectorSpace για να επιστραφούν ταξινομημένα τα αποτελέσματα.

Lucene scoring

Κάθε φορά που ένα document ταιριάζει με τα κριτήρια μιας αναζήτησης, το Lucene υπολογίζει ένα βαθμό (τιμή συνάφειας) και εκχωρεί το σκορ στο document . Το σκορ αντικατοπτρίζει το πόσο ταιριάζει ένα document σε ένα query. Υψηλότερες βαθμολογίες δείχνουν ισχυρότερη ομοιότητα . Το Lucene για να ταξινομήσει τα αποτελέσματα της αναζήτησης χρησιμοποιεί την ακόλουθη φόρμουλα. Η βαθμολογία υπολογίζεται για κάθε Document (d) που αντιστοιχεί σε κάθε Term (t) σε ένα Query (q)

$$\sum tinq\ ((tf\ (tind\ )\ idf\ (t)\ boost(t.fieldind\ )\ lengthNorm(t.fieldind\ ))\ coord(q,d\ )\ queryNorm(q))$$

tf(t in d)	Παράγοντας συχνότητας για το t στο d.Ο αριθμός της συχνότητας που ο όρος t εμφανίζεται στο τρέχον έγγραφο. Έγγραφα που έχουν περισσότερες εμφανίσεις ενός όρου λαμβάνουν υψηλότερη βαθμολογία .
idf(t)	Inverse document frequency. Αντιπροσωπεύει τη μοναδικότητα ενός Term. Σπανιότεροι όροι δίνουν μεγαλύτερη συνεισφορά στο σύνολο της βαθμολογίας .
boost(t.field in d)	Field και document boost. Η <<ώθηση>> που προσδιορίζεται στο ερώτημα αναζήτησης από τον χρήστη. Μια ώθηση πάνω από 1,0 θα αυξήσει τη σημασία αυτού του όρου ενώ μια ώθηση κάτω από 1.0 θα μειώσει τη σημασία του. Μια ώθηση που ισούται με 1.0 (η προεπιλεγμένη ώθηση) δεν έχει καμία

	επίδραση .
lengthNorm(t.field in d)	Παράγοντας ο οποίος εφαρμόζεται για διαφορετικά μήκη στα πεδία που πραγματοποιήθηκε η αναζήτηση. Συνήθως μακρύτερα σε μήκος πεδία επιστρέφουν μικρότερη τιμή. Αυτό σημαίνει αναζητήσεις που ταιριάζουν με μικρότερα πεδία λαμβάνουν υψηλότερη βαθμολογία από αυτά με μακρύτερα πεδία .
coord(q, d)	Coordination factor. Παράγοντας βαθμολογίας με βάση πόσους από κοινού όρους στο συγκεκριμένο document υπάρχουν με το ερώτημα . Συνήθως ένα έγγραφο που περιέχει περισσότερους από τους όρους του ερωτήματος θα λάβει υψηλότερη βαθμολογία από ότι ένα άλλο έγγραφο με λιγότερους όρους αναζήτησης
queryNorm(q)	Παράγοντας που χρησιμοποιείται για να κάνει τα αποτελέσματα των ερωτημάτων συγκρίσιμα .

Κεφάλαιο 4: Σχετικές μελέτες

Υπάρχουν δύο κύριες περιοχές έρευνας που σχετίζονται με την εργασία μας. Πρώτον, είναι η λήψη και καταγραφή των μηνυμάτων tweets, και δεύτερον είναι η επεξεργασία αυτών χρησιμοποιώντας τη βιβλιοθήκη Lucene. Στην ενότητα αυτή δίνουμε μια επισκόπηση των σχετικών μελετών που χρησιμοποιούν το Twitter και τη βιβλιοθήκη Lucene για την ανάλυση των δεδομένων.

Τα τελευταία χρόνια, τα κοινωνικά δίκτυα και οι micro-blogging υπηρεσίες έγιναν πολύ δημοφιλή ερευνητικά θέματα. Το Twitter είναι ιδιαίτερα ενδιαφέρον για τους ερευνητές λόγω της προσβασιμότητας των δεδομένων, καθώς σε αντίθεση με τα περισσότερα κοινωνικά δίκτυα, τα μηνύματα των χρηστών μπορούν να ανακτηθούν σε πραγματικό χρόνο μέσω του Twitter API .

Η [1] είναι μια έρευνα η οποία ειδικεύεται στην καταγραφή συγκεκριμένων tweets τα οποία αναφέρονται σε αρρώστιες. Η αρχιτεκτονική του συστήματος παρουσιάζει ομοιότητες με την εφαρμογή της παρούσας εργασίας. Για την καταγραφή των tweets σε πραγματικό χρόνο η έρευνα χρησιμοποιεί το Streaming API του twitter, δίνοντας ως φίλτρα λέξεις οι οποίες αναφέρονται σε νοσήματα. Αφού μετατραπεί η ροή από tweets σε json μορφή, αυτά αποθηκεύονται σε αρχεία που το κάθε αρχείο μπορεί να περιέχει έως 1000 tweets. Έπειτα για το indexing και την αναζήτηση χρησιμοποιείται η βιβλιοθήκη Lucene. Πιο συγκεκριμένα η έρευνα χρησιμοποιεί την αναζήτηση που προσφέρει το Lucene ως scanner για την εύρεση μηνυμάτων τα όποια αναφέρονται σε νοσήματα, αλλά δε είναι αξία προσοχής και θα πρέπει να αγνοηθούν. Για παράδειγμα μηνύματα που περιέχουν Bieber fever (π.χ. Getting Bieber fever), μηνύματα που περιέχουν τη λέξη Love και μηνύματα τα οποία περιέχουν τις λέξεις Shot και Vaccine. Τα υπόλοιπα αποτελέσματα της αναζήτησης αποθηκεύονται προσωρινά σε μια σχεσιακή βάση δεδομένων MySql. Για την επιπλέον ορθότητα των αποτελεσμάτων και προτού εφαρμοστεί η κατηγοριοποίηση (Classification), ελέγχονται τα δεδομένα που βρίσκονται στην προσωρινή βάση δεδομένων για tokens, τα οποία δε είναι σωστά ορθογραφημένα, αυτά τα token απορρίπτονται. Στο τέλος πριν την αποθήκευση των τελικών αποτελεσμάτων σε μια νέα βάση δεδομένων, για την επεξεργασία και κατηγοριοποίηση των tweets χρησιμοποιείται το μοντέλο των Μηχανών Διανυσμάτων Υποστήριξης (Support

Vector Machines – SVM). Οι πειραματικές μετρήσεις έδειξαν ότι το μοντέλο κατηγοριοποίησης που εφαρμόστηκε μπορούσε να αναγνωρίσει μηνύματα Tweets τα οποία αναφέρονταν σε νοσήματα με επιτυχία 85.5%. Επιπλέον παρατηρήθηκε ότι ένα tweet κατά μέσο όρο χρειάζεται 12.46 Milliseconds από την είσοδο έως την έξοδο του από το σύστημα και ότι συνολικά το σύστημα μπορεί να διαχειριστεί καθημερινά 6 εκατομμύρια tweets. Συγκρίνοντας την αρχιτεκτονική της έρευνας με αυτή της παρούσας εργασίας παρατηρείται ότι η χρήση βάσεων - δεδομένων, καθώς και η χρήση αρχείων για την προσωρινή αποθήκευση των Tweets δε είναι απαραίτητη. Στην εργασία αυτή προτείνεται μέσω της δομής - ουράς η χρησιμοποίηση της εσωτερικής μνήμης RAM του υπολογιστή για την αντικατάσταση των αρχείων που αποθηκεύουν υπό μορφή json τα tweets. Επιπλέον προτείνεται η on the fly επεξεργασία του Index χρησιμοποιώντας την βιβλιοθήκη Lucene, καθώς αυτό καθιστά μη απαραίτητη τη χρησιμοποίηση σχεσιακών βάσεων - δεδομένων.

Μια άλλη έρευνα [4] χρησιμοποιεί εκτενώς τη δυνατότητα που δίνει το twitter για την καταγραφή της γεωγραφικής θέσης των tweets, έτσι ώστε να εντοπίσει σημαντικά γεγονότα σε μια συγκεκριμένη γεωγραφική περιοχή. Τα tweets περιέχουν ένα μεγάλο ποσοστό θορύβου και είναι δύσκολο να εξαχθούν πολύτιμες πληροφορίες από αυτά. Αυτή η πρόκληση έχει οδηγήσει τους ερευνητές στην εύρεση γεγονότων τα όποια συμβαίνουν σε μεγαλύτερη κλίμακα. Η παρούσα έρευνα εστιάζει στον εντοπισμό γεγονότων σε μικρότερη κλίμακα. Για την εύρεση αυτών των γεγονότων παρουσιάζεται ένα μοντέλο που εξάγει την πληροφορία της τοποθεσίας εκτός από τα tweets τα οποία είναι geotagged και από το κείμενο, χρησιμοποιώντας το Solr, το οποίο ως βάση χρησιμοποιεί τη βιβλιοθήκη του Lucene. Σύμφωνα με τη μελέτη, αυτή η διεργασία αυξάνει τον αριθμό των geotagged μηνυμάτων του Twitter κατά περίπου 2,5 φορές. Για να εξάγει από το κείμενο την πληροφορία της τοποθεσίας χρησιμοποιεί αρχικά τα μηνύματα τα οποία είναι geotagged, αγνοώντας τα retweets, και περιέχουν στο κείμενο τους μια αναφορά σε μια τοποθεσία. Έτσι χρησιμοποιεί τρείς βάσεις δεδομένων. Η πρώτη περιέχει τα μηνύματα που είναι geotagged, καθώς η δεύτερη περιέχει την τοποθεσία που έχει εξαχτεί από το κείμενο μαζί με την ακριβή γεωγραφική θέση του μηνύματος δημιουργώντας ένα πρότυπο της μορφής(“I’m at Times Square (Broadway& 7th Ave, btn 42nd & 47th, New York) w/ 3 others”). Επίσης υπολογίζεται η γεωγραφική διακύμανση μεταξύ της τοποθεσίας και της γεωγραφικής θέσης, όσο πιο μικρή η διακύμανση τόσο πιο ακριβής η περιγραφή της

τοποθεσίας με αυτή της γεωγραφικής θέσης. Τέλος η τρίτη βάση περιέχει μηνύματα τα οποία δε είναι geotagged. Το σύστημα έχει τη δυνατότητα με αυτό το τρόπο να εντοπίζει χρησιμοποιώντας τη δεύτερη βάση δεδομένων τη σχετική γεωγραφική θέση ενός μη geotagged μηνύματος. Έπειτα για να εξάγει τις πιο διαδεδομένες τοποθεσίες χρησιμοποιεί έναν αλγόριθμο, ο όποιος μετατρέπει τις πληροφορίες για την τοποθεσία σε hash τιμές. Οι hash τιμές είναι ίδιες για τοποθεσίες οι οποίες βρίσκονται γεωγραφικά κοντά. Για τις τοποθεσίες που θα αναγνωριστούν ως πιο διαδεδομένες το σύστημα αναζητά λέξεις -κλειδιά τα οποία χρησιμοποιούνται περισσότερες από 3 φόρες. Αυτές οι λέξεις -κλειδιά ορίζουν το τοπικό γεγονός. Πειραματικά αποδείχτηκε ότι η ερευνά είχε επιτυχία εντοπισμού ενός σημαντικού γεγονότος στο 25.5 % . Ωστόσο αξιοσημείωτο είναι ότι με τη χρήση της εξαγωγής της τοποθεσίας από το κείμενο το σύστημα κατάφερε να εξάγει 689 διαδεδομένες τοποθεσίες, ενώ χωρίς αυτό μόλις 6, αύξηση 115%.

Η [5] είναι μια παρόμοια μελέτη στη χρησιμοποίηση του twitter, αλλά και των geotagged μηνυμάτων, για την ανίχνευση και την ανάλυση γεγονότων από τα Social Media που σκοπό έχει τη δημόσια ασφάλεια. Συμφώνα με την έρευνα 52% των εγκλημάτων που συνέβησαν, δεν καταγράφηκαν από την αστυνομία . Η μελέτη αυτή ως πηγή δεδομένων χρησιμοποιεί το twitter, αλλά και δεδομένα που διατίθενται στο internet όπως της αστυνομίας, πυροσβεστικής κτλ. Για το indexing και την αναζήτηση χρησιμοποιείται η βιβλιοθήκη Lucene. Δημιουργούνται δυο διαφορετικά indexes, το ένα περιέχει τα αρχικά tweets και το άλλο το σύνολο των tweets τα οποία έχουν αναγνωριστεί ότι αναφέρονται σε ένα σημαντικό γεγονός. Το πρώτο index χρησιμοποιείται για την ανάλυση και εξαγωγή των σημαντικών γεγονότων, ενώ το δεύτερο index το χρησιμοποιεί η εφαρμογή για την αποθήκευση των μηνυμάτων αντικαθιστώντας έτσι τη χρησιμοποίηση μιας βάσης δεδομένων. Επιπλέον, για την περαιτέρω διεύρυνση της έννοιας μιας λέξης που υπάρχει σε ένα μήνυμα tweet, η παρούσα μελέτη χρησιμοποιεί ένα offline λειτουργικό το IPSV [18] το οποίο δημιουργεί metada και κατηγορίες για κάθε λέξη, έτσι ώστε τα αποτελέσματα της αναζήτησης να είναι πιο αξιόπιστα. Το σύστημα για τον εντοπισμό σημαντικών γεγονότων σε μια συγκεκριμένη γεωγραφική περιοχή και για μια συγκεκριμένη χρονική στιγμή χρησιμοποιεί τη μέθοδο Louvain [17], η οποία βασίζεται στη μέθοδο του local modularity maximization. Όπως και στην προηγούμενη έρευνα, [4] για να αυξηθεί ο αριθμός των μηνυμάτων που αναφέρονται σε σημαντικά γεγονότα,

εξάγονται πληροφορίες σχετικές με την τοποθεσία ενός μηνύματος και από το κείμενο του tweet. Έπειτα υλοποιείται ένας ταξινομητής ο όποιος επιλέγει σε ποια κατηγορία ανήκει το κάθε μήνυμα tweet ή και ένα σύνολο tweets. Οι διαθέσιμες κατηγορίες είναι αυτές οι οποίες έχουν επιλεχτεί με το υψηλότερο ποσοστό σχετικότητας για το γεγονός που συμβαίνει. Στο τέλος χρησιμοποιείται το Sentiment140 API [6], το οποίο αξιολογείται στην παρούσα εργασία, για την υλοποίηση της ανάλυσης συναισθήματος έτσι ώστε το σύστημα να εξάγει το συναίσθημα των χρηστών ως προς ένα γεγονός. Εάν και στατιστικά αυτά τα δεδομένα είναι χρήσιμα η μελέτη αυτή δεν χρησιμοποιεί μια αρχιτεκτονική καταγραφής μηνυμάτων σε πραγματικό χρόνο. Η χρησιμοποίηση μιας αρχιτεκτονικής πραγματικού χρόνου η οποία παρουσιάζεται και προτείνεται από την παρούσα διπλωματική εργασία θα μπορούσε να προσφέρει στην συγκεκριμένη έρευνα άμεσα στοιχεία για τη δημόσια ασφάλεια.

Τελικώς, παρατηρούμε ότι οι περισσότερες μελέτες οι οποίες αντλούν τα δεδομένα τους από το Twitter επικεντρώνονται πρωτίστως στο να καταγράφουν τα εισερχόμενα δεδομένα από το twitter και έπειτα να τα επεξεργάζονται. Για παράδειγμα στις μελέτες [4] [5] η επεξεργασία των δεδομένων εφαρμόζεται αφού έχουν καταγραφεί τα δεδομένα. Στην παρούσα εργασία προτείνεται μια διαφορετική αρχιτεκτονική ενός συστήματος, η όποια επιτρέπει την καταγραφή και ανάλυση των δεδομένων από το Twitter σε πραγματικό χρόνο. Επιπλέον η αρχιτεκτονική της εργασίας θεωρώντας ότι η εφαρμογή σχεσιακών βάσεων – δεδομένων, όπως χρησιμοποιείται στην ερευνά [1] δεν είναι πλέον μια αποτελεσματική μέθοδος, προτείνει τη χρησιμοποίηση ενός συστήματος NoSQL αξιοποιώντας τη βιβλιοθήκη Lucene.

Κεφάλαιο5: Αρχιτεκτονική συστήματος

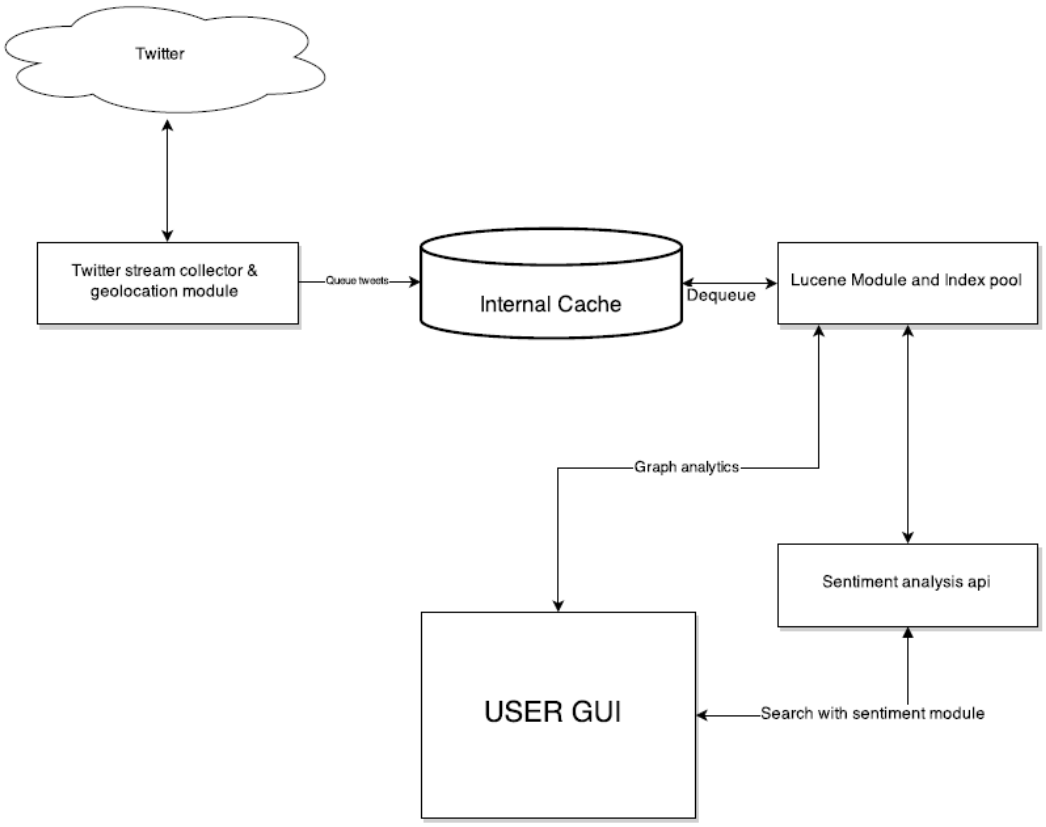
Εισαγωγή

Χρησιμοποιείται το streaming API του twitter φιλτραρισμένο από ένα σύνολο γεωγραφικών τιμών που ορίζουν μια ευρύτερη περιοχή ως πηγή των δεδομένων μας. Τα tweets έπειτα επεξεργάζονται και χρησιμοποιείται η βιβλιοθήκη Lucene για την δημιουργία, αλλά και την αναζήτηση ενός index. Ο χρήστης χρησιμοποιώντας αυτό το Index μπορεί να κάνει σύνθετες αναζητήσεις στο σύνολο των tweets, καθώς και να αναλύσει το συναίσθημα που προκύπτει από τα αποτελέσματα της αναζήτησης μέσα από τεχνικές ανάλυσης συναισθήματος. Παρακάτω θα αναλυθούν λεπτομερώς οι διαδικασίες αυτές του συστήματος.

Αρχιτεκτονική συστήματος.

Για την υλοποίηση της παρούσας εργασίας χρειάστηκε να δημιουργηθεί η αρχιτεκτονική ενός συστήματος το οποίο θα καταγράφει, αλλά και θα επεξεργάζεται τα tweets για μια συγκεκριμένη γεωγραφική περιοχή σε πραγματικό χρόνο. Το σύστημα αποτελείται από τις παρακάτω ενότητες.

- Λήψη Tweets
- Επεξεργασία και Indexing
- Ανάκτηση πληροφοριών από το Index
- Παρουσίαση αποτελεσμάτων στον τελικό χρήστη.



Εικόνα 7 Αρχιτεκτονική συστήματος.

Παρακάτω θα αναλυθούν πιο αναλυτικά οι ενότητες του συστήματος.

Λήψη Tweets

Για την καταγραφή των tweets χρησιμοποιείται το Streaming API. Πιο συγκεκριμένα, για να καταγράφουν τα tweets εντός μιας συγκεκριμένης γεωγραφικής περιοχής στέλνεται ένα POST HttpWebRequest αίτημα στη διεύθυνση https://stream.twitter.com/1.1/statuses/filter.json. Το twitter,για να πιστοποιήσει την ταυτότητα της εφαρμογής, χρησιμοποιεί το OAuth το οποίο αναλύθηκε παραπάνω. Αφού ληφθούν τα απαραίτητα στοιχεία για την πιστοποίηση της εφαρμογή μας από το Twitter, εισάγεται ως Header στο HTTP αίτημα ένα πεδίο που έχει ως κλειδί την τιμή Authorizationκαι η τιμή του είναι η ακόλουθη.

"OAuth oauth_nonce=\"NjM1NzA0OTkxOTgxNTk2OTgz\",
oauth_signature_method=\"HMAC-SHA1\", oauth_timestamp=\"1434891598\",
oauth_consumer_key=\"szOqeMR6YoqeVkEQ0RnUQA\",
oauth_token=\"27613498-AflUzoAeKJQD2xY8sc0SwGtz4rg3tIHut8TENYhpc\",
oauth_signature=\"lx6gzSwieFEreVeq2C7i9i8CFcY%3D\", oauth_version=\"1.0\""

Εικόνα 8 OAuth Authorization Value

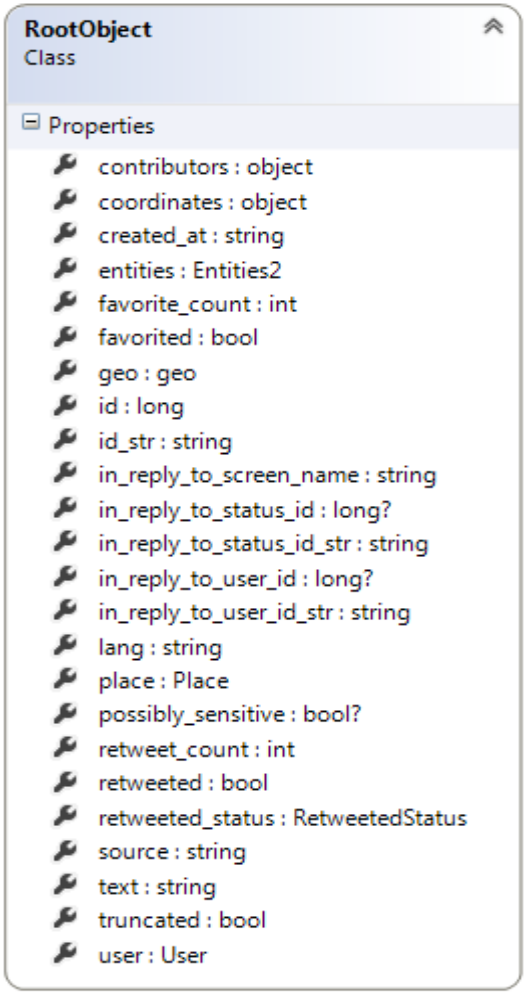
Για το Body του HTTP αιτήματος χρησιμοποιείται η παράμετρος Locations. Η τιμή της συγκριμένης παραμέτρου, που μας δίνεται από το streaming API, αποτελείται από μια λίστα διαχωρισμένων με κόμμα τιμών που περιγράφουν μια γεωγραφική περιοχή (BoundingBox). Οι τιμές αυτές παρουσιάζονται ως ζεύγος γεωγραφικού πλάτους και μήκους. Ένα παράδειγμα μια τιμής που περιγράφει μια γεωγραφική περιοχή είναι η 23.686986,37.948818,23.789693,38.032856 που προσδιορίζει την Αθήνα. Επομένως το αποτέλεσμα αυτή της διαδικασίας θα μας επιστρέψει όλα τα tweets τα οποία περιέχουν τοποθεσία και βρίσκονται εντός της γεωγραφικής περιοχής που επιθυμούμε. Αντιθέτως με το Rest API, το Streaming API δεν χρησιμοποιεί την τοποθεσία που έχει εισάγει ο χρήστης στο προφίλ του για να φιλτράρει το γεωγραφική περιοχή. Υπάρχουν δυο προϋποθέσεις που πρέπει να ισχύουν για να καταγράψουμε τα μηνύματα από το twitter.

1. Το πεδίο <<coordinates>> να μην είναι κενό, και η τιμή αυτού του πεδίου, που είναι της μορφής γεωγραφικό πλάτος/μήκος, να συμπίπτει εντός του ορισμένου bounding box.
2. Το πεδίο <<coordinates>> να είναι κενό, αλλά το πεδίο <<place>> να έχει τιμή. Και εδώ ισχύει ότι και προηγουμένως, δηλαδή η τιμή αυτή πρέπει να συμπίπτει εντός του bounding box.

Βλέποντας το παραπάνω διάγραμμα η περιγραφή που μόλις είδαμε αντιστοιχεί στο <<Twitter stream collector & geolocation module>>

Επεξεργασία και Indexing

Το Streaming API επιστρέφει μια ροή από tweets σε μορφή JSON που βρίσκονται εντός μιας επιλεγμένης γεωγραφικής περιοχής σε πραγματικό χρόνο. Τα μηνύματα αυτά μετατρέπονται σε ειδικά προσαρμοσμένα αντικείμενα, τα οποία και ονομάζουμε RootObject. Η κλάση, η οποία έχει δημιουργηθεί για αυτό το σκοπό, μοντελοποιεί πλήρως τα πεδία ενός μηνύματος από το Twitter. Παρακάτω παρουσιάζεται η κλάση RootObject, καθώς και τα properties τα όποια την αποτελούν.



Έχοντας ως σκοπό το σύστημα να μην χάσει κανένα από τα επιστρεφόμενα tweets εισήχθη μια δομή τύπου ουράς FIFO (First-In-First-Out, Πρώτο-Μέσα-Πρώτο-Έξω) ενδιάμεσα της λειτουργιάς της επεξεργασίας των μηνυμάτων και αυτή του Indexing. Σε αυτή τη δομή εισάγονται προσωρινά στη RAM τα αντικείμενα τύπου RootObject, έως ότου να επεξεργαστούν. Αυτή η διαδικασία εκτελείται συνεχώς και

αδιάκοπτα σε ένα ξεχωριστό Thread από το Main Thread. Τώρα μια άλλη διαδικασία σε ένα άλλο διαφορετικό Thread αναλαμβάνει το ρόλο της επεξεργασίας της δομής ουράς και της δημιουργίας του Index. Η συγκεκριμένη διεργασία για κάθε μοναδικό tweet δημιουργεί και από ένα Lucene document. Κάθε document αντιπροσωπεύει και ένα tweet. Το κάθε document περιέχει τα ακόλουθα fields :

Field name	Analyzed	FieldStore
Text	NAI	OXI
Language	NAI	NAI
UserName	NAI	OXI
Country	NAI	OXI
PlaceName	NAI	OXI
Hashtags	NAI	NAI
JsonTweet	OXI	NAI

Όπως παρατηρείται, τα περισσότερα fields δηλώνονται ως Analyzed εκτός του πεδίου JsonTweet. Ο λόγος που συμβαίνει αυτό είναι, επειδή το Lucene χρειάζεται να δημιουργήσει tokens, τα όποια θα είναι αναζητήσιμα μέσω του Inverted Index. Έτσι, για παράδειγμα, εάν ο χρηστής θελήσει να αναζητήσει tweets τα όποια προέρχονται από ένα συγκεκριμένο χρηστή του twitter, θα πρέπει η αναζήτηση του να εφαρμοστεί στο πεδίο UserName το οποίο περιέχει τα ονόματα των χρηστών. Εν αντιθέσει το πεδίο JsonTweet δεν πρέπει να είναι διαθέσιμο για αναζητήσεις. Αυτό το πεδίο αποτελείται από όλα τα πεδία ενός μηνύματος tweet σε json μορφή και ο σκοπός ύπαρξής του είναι η αποθήκευσή του για τη μετέπειτα παρουσίαση του μέσω της εφαρμογής στον χρήστη. Αυτό που θέλουμε είναι το ακριβές κείμενο string, σε μορφή json, του πεδίου JsonTweet να αποθηκευτεί στο Index ως έχει και για αυτό το λόγο δηλώνεται με την επιλογή FieldStore=NAI. Τα υπόλοιπα δυο πεδία τα όποια έχουν τεθεί ως FieldStore= NAI είναι απαραίτητα για την εμφάνιση των τιμών τους στις γραφικές παραστάσεις. Τα πεδία όπως Text, UserName, Country και PlaceName δεν χρειάζεται να αποθηκευτούν, αφού χρησιμοποιούνται μόνο στις αναζητήσεις του Lucene, έτσι έχουν δηλωθεί μόνο ως Analyzed.

Τα Documents και τα Fields αυτά αποτελούν το Index. Η κλάση, η όποια είναι υπεύθυνη για τη δημιουργία του Index,ονομάζεται IndexWriter. Ο constructor του αντικείμενου αυτού προσδιορίζει το Directory που θα δημιουργηθεί το Index, το Lucene Analyzer που θα χρησιμοποιηθεί, καθώς και το επιτρεπτό όριο μεγέθους που θα δέχεται για κάθε Field. Στην παρούσα εργασία για να επεξεργαστούν οι τιμές των fields χρησιμοποιείται ο StandarAnalyzer που παρουσιάστηκε στα προηγούμενα κεφάλαια. Ο StandarAnalyzer χρησιμοποιείται για τη μετατροπή ενός κειμένου σε tokens, αφαιρεί χαρακτήρες όπως σημεία στίξης, μετατρέπει τα κεφαλαία σε lowercase και αφαιρεί stopwords. Επιπλέον τίθεται το επιτρεπτό όριο μεγέθους ενός Field που γίνεται Indexed στους 10.000 όρους. Αφού δημιουργηθεί το αντικείμενο της κλάσης IndexWriter προστίθενται σε αυτό τα documents που έχουν δημιουργηθεί. Ο IndexWriter αποθηκεύει στην προσωρινή μνήμη RAM το σύνολο των documents, πρώτου καταχωρήσει τα Documents στο Index. Για να προστατέψει τη μνήμη του υπολογιστή από υπερχείλιση, καταχωρεί στο Index τα Documents, προτού η χρήση της RAM φτάσει τα 16MB. Αυτή είναι μια Default τιμή και μπορεί να προσαρμοστεί κατάλληλα, αν θελήσουμε. Στην εφαρμογή μας ο buffer καταχωρεί τα Documents στο index κάθε φορά που χρησιμοποιείται το index για αναζητήσεις. Αυτό συμβαίνει κάθε μισό λεπτό, καθώς ανανεώνονται οι γραφικές παραστάσεις που παρουσιάζονται στην εφαρμογή. Η παραπάνω ρουτίνα που περιγράφτηκε, λειτουργεί συνεχώς σαν ένα ατέρμονα βρόγχο συλλέγοντας και εκχωρώντας στο Index τα μηνύματα του Twitter.

Βλέποντας το παραπάνω διάγραμμα, η περιγραφή που μόλις αναλύθηκε αντιστοιχεί στο <<Internal cache>> και αντιπροσωπεύει τη δομή ουράς FIFO και στο <<Lucene module & Index pool>> το οποίο αντιπροσωπεύει τη δημιουργία του index από documents.

Ανάκτηση πληροφοριών από το Index

Στην παρούσα εφαρμογή οι πληροφορίες που καταχωρούνται στο Index χρησιμοποιούνται κατά δυο διαφορετικούς τρόπους.

- Δημιουργία γραφημάτων για την οπτική ανάδειξη χρήσιμων πληροφοριών.
- Σύνθετη αναζήτηση ανάκτησης πληροφοριών.

Για να επιτευχτεί η αναζήτηση πληροφοριών από το Index σε πραγματικό χρόνο χρησιμοποιείται ένα σημαντικό χαρακτηριστικό κομμάτι της βιβλιοθήκης Lucene που ονομάζεται αναζήτηση σε πραγματικό χρόνο (near-real-time search). Η αναζήτηση σε πραγματικό χρόνο επιτρέπει την αναζήτηση σε Documents άμεσα μετά το Indexing. Για την ανάκτηση αυτών των πληροφοριών χρησιμοποιείται η μέθοδος του IndexWriter που ονομάζεται getReader. Η κύρια λειτουργιά της συγκεκριμένης μεθόδου είναι η επιστροφή ενός readonly reader που εμπεριέχει όλα τα καταχωρημένα documents στο Index, καθώς και αυτά που δε έχουν καταχωρηθεί ακόμα και βρίσκονται στη προσωρινή μνήμη RAM. Αξίζει να σημειωθεί ότι με την κλήση αυτή της μεθόδου τα προσωρινά αποθηκευμένα documents καταχωρούνται στο Index. Αυτή η λειτουργία μας προσφέρει αναζήτηση σε πραγματικό χρόνο χωρίς να χρειάζεται να κλείνουμε το indexWriter και να τον ανοίγουμε ξανά. Δηλαδή, ενώ εκτελούνται όλες οι διαδικασίες της εφαρμογής, όπως parsing, επεξεργασία, και indexing, δίνεται η δυνατότητα στο χρήστη να εκτελεί αναζητήσεις στο Index. Έπειτα αφού έχουμε τον reader που περιέχει τα καταχωρημένα indexed documents υλοποιούνται δύο διαφορετικοί τρόποι για την αναζήτηση των δεδομένων που παρουσιάζονται παρακάτω.

Δημιουργία γραφημάτων για την οπτική ανάδειξη χρήσιμων πληροφοριών

Παρουσιάζονται δυο διαφορετικοί τύποι γραφημάτων σε γράφημα πίτας ανάλογα με την επιλογή του χρήστη. Το πρώτο γράφημα παρουσιάζει τα δέκα πιο διαδεδομένα χρησιμοποιούμενα Hashtags στο σύνολο των καταγραμμένων tweets, ενώ το δεύτερο παρουσιάζει τις δέκα πιο διαδεδομένες γλώσσες που έχουν χρησιμοποιηθεί για τη σύνταξη των tweets. Για την εξαγωγή αυτών των πληροφοριών έχει δημιουργηθεί μια μέθοδος, η οποία ονομάζεται FindMostFrequentlyUsedFields. Η συγκεκριμένη μέθοδος δέχεται ως παραμέτρους έναν Lucene reader και το πεδίο το οποίο επιθυμούμε να μελετήσουμε. Έχοντας τον reader ελέγχεται ο αριθμός των μη διαγραμμένων documents καλώντας την μέθοδο NumDocs. Χρησιμοποιώντας έναν επαναληπτικό βρόγχο για κάθε Document λαμβάνουμε τις τιμές για το επιλεγμένο πεδίο (Hashtags ή Language αναλόγως με την επιλογή του χρήστη). Εσωτερικά δημιουργείται ένας πίνακας κατακερματισμού, ο όποιος περιέχει ως κλειδί ένα string, παρουσιάζοντας το hashtag ή τη γλώσσα, και μια τιμή integer, η οποία αυξάνεται για

κάθε εμφάνιση του κλειδιού και προσδιορίζει το πόσες φόρες εμφανίζεται αυτό. Στο τέλος ταξινομούνται κατά φθίνουσα σειρά τα αποτελέσματα και επιλέγονται τα πρώτα δέκα. Τα αποτελέσματα παρουσιάζονται στο τέλος στο χρήστη σε γράφημα πίτας. Επειδή το σύστημα λειτουργεί σε πραγματικό χρόνο, η συγκεκριμένη διαδικασία εκτελείται κάθε μισό λεπτό (0.5sec), έτσι ώστε να ενημερώνονται οι γραφικές απεικονίσεις με νεότερα αποτελέσματα. Για τη μέτρηση των χρονικών διαστημάτων χρησιμοποιείται η κλάση System.Windows.Forms.Timer που παρέχεται από το .Net Framework θέτοντας το property Interval ίσο με 30000 millisecond. Όταν περάσει αυτός ο χρόνος, καλείται μια μέθοδος η οποία με την σειρά της καλεί την μέθοδο createCharts, ώστε να εκτελέσει τις διαδικασίες που μόλις περιγράφτηκαν.

Σύνθετη αναζήτηση ανάκτησης πληροφοριών.

Το σύστημα δίνει τη δυνατότητα στον χρήστη να εκτελέσει σύνθετες αναζητήσεις σε όλα τα Fields τα οποία εμπεριέχονται σε ένα Lucene document. Για παράδειγμα, μπορούν να πραγματοποιηθούν αναζητήσεις στο κείμενο του tweet, στα hashtags ή ακόμα και στη γλώσσα που χρησιμοποιεί ένα tweet. Για την αναζήτηση αυτή χρησιμοποιείται η βιβλιοθήκη του Lucene. Πιο συγκεκριμένα έχοντας τον Lucene reader ελέγχεται καλώντας την μέθοδο NumDocs ο αριθμός των μη διαγραμμένων documents, όπως ακριβώς συνέβαινε και στην αναζήτηση για την δημιουργία των γραφικών διαγραμμάτων. Έπειτα, αν αυτός ο αριθμός των Documents είναι μεγαλύτερος του μηδενός, δημιουργείται ένα αντικείμενο της κλάσης IndexSearcher. Αυτή είναι η κεντρική κλάση που χρησιμοποιείται για την αναζήτηση των εγγράφων σε ένα Index, η οποία προσπελαύνει ένα ευρετήριο και προσφέρει ένα σύνολο από μεθόδους αναζήτησης . Η μέθοδος της κλάσης IndexSearcher που χρησιμοποιείται για την αναζήτηση είναι η Search. Η Search μέθοδος δέχεται ως ορίσματα ένα αντικείμενο τύπου Query και ένα αντικείμενο τύπου TopScoreDocCollector. Το αντικείμενο τύπου Query περιέχει τα κριτήρια αναζήτησης που δημιουργούνται από την QueryParser κλάση. Η λειτουργιά της QueryParser κλάσης είναι η μετατροπή ενός ερωτήματος που έχει εισαχθεί από τον χρήστη, σε ένα instance ενός Query αντικείμενου. Παρακάτω παρουσιάζονται

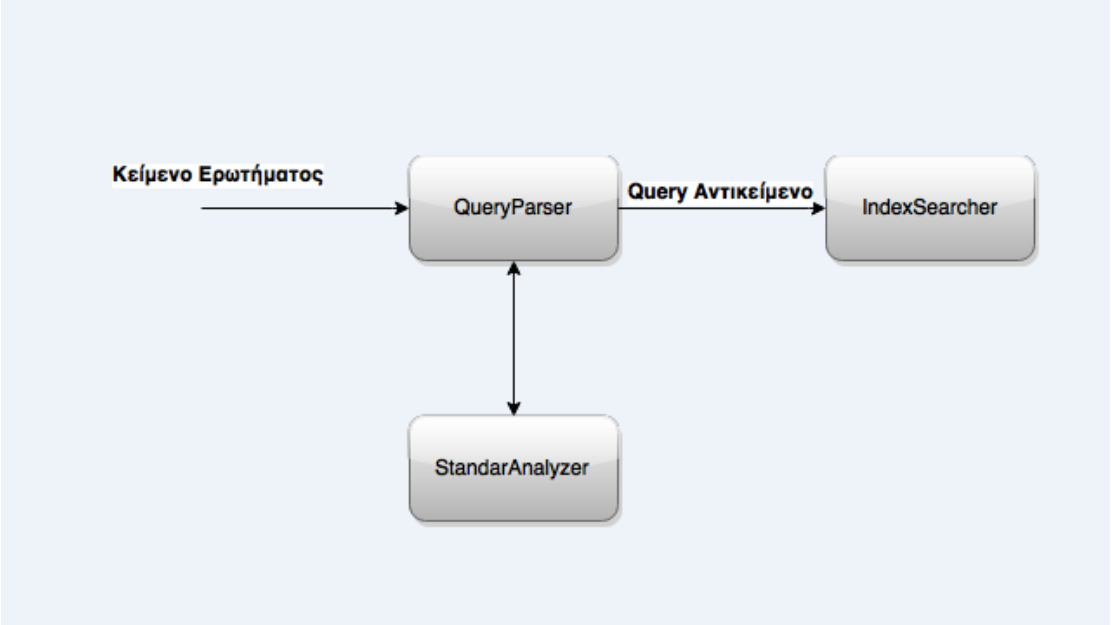
σύνθετες εκφράσεις ερωτήματος τις οποίες μπορεί να μετατρέψει ο QueryParser της εφαρμογής :

Ερώτημα	**Επιστροφή Document που περιέχουν…**
London	Τον όρο London στο επιλεγμένο πεδίο
London Cycling	Τον όρο London ή Cycling, ή και τους 2, στο επιλεγμένο πεδίο
Londond OR Cycling	
+London +Cycling	Και τους 2 όρους London και Cycling μαζί, στο επιλεγμένο πεδίο
London AND Cycling	
UserName: Nikos	Τον όρο Nikos στο UserName πεδίο
UserName: Nikos -Text: Cycling	Τον όρο Nikos στο UserName πεδίο, και δεν περιέχει τον όρο Cyclingστο πεδίο Text
UserName: Nikos AND NOT Text: Cycling	
(Walking OR Cycling) AND London	Τον όρο London και πρέπει να περιέχει ένα από τους 2 Όρους Walking ή Cycling, ή και τους 2 μαζί, στο επιλεγμένο πεδίο.
Text: "Yesterdays cycling event"	Ακριβώς την φράση Yesterdays cycling event, στο πεδίο Text.
Text: "Love cycling" ~5	Τους όρους Love και Cycling μεταξύ 5 θέσεων από αυτούς.
Lo*	Όρους που ξεκινούν με Lo, όπως Love, Loving, καθώς και τον όρο Lοκαθεαυτό.
Love~	Όρους οι οποίοι είναι παρόμοιοι με τον όρο Love, όπως Move, στο επιλεγμένο πεδίο.

Εικόνα 9 Εκφράσεις Ερωτήματος QueryParser

Είναι απαραίτητο στη QueryParser κλάση να δηλωθεί ο Analyzer που θα χρησιμοποιηθεί, έτσι ώστε να μετατραπεί το κείμενο του ερωτήματος (querytext). Σε αυτή την εργασία χρησιμοποιείται ο StandarAnalyzer που αφαιρεί χαρακτήρες, όπως σημεία στίξης, μετατρέπει τα κεφαλαία σε lowercase και αφαιρεί stop words, όπως

ακριβώς χρησιμοποιήθηκε το ίδιο analyzer και στη διαδικασία του Indexing. Η μέθοδος αναζήτησης κάνει ένα τεράστιο έργο πολύ γρήγορα. Επισκέπτεται κάθε ενιαίο document που είναι υποψήφιο για να ταιριάξει με την αναζήτηση, και συγκεντρώνει και επιστρέφει μόνο αυτά τα οποία πληρούν όλους τους όρους της αναζήτησης.



Εικόνα 10 QueryParser

Η δεύτερη παράμετρος που δέχεται η Search μέθοδος είναι το αντικείμενο τύπου TopScoreDocCollector, το οποίο συγκεντρώνει τα topDocs που επιστρέφονται κατά την αναζήτηση και τα ταξινομεί με φθίνουσα σειρά σύμφωνα με το Score τους. Έχει τεθεί ο αριθμός 400 ως το όριο των topDocs που θα επιστρέφει ο Collector. Στο τέλος χρησιμοποιώντας έναν επαναληπτικό βρόγχο για κάθε TopDoc λαμβάνονται ένα-ένα τα Documents και παρουσιάζεται το πεδίο JsonTweet στον χρήστη χρησιμοποιώντας ένα WinForm.DataGrid αντικείμενο.

Ανάλυση συναισθήματος (Sentiment analysis)

Χρησιμοποιείται η μέθοδος της ανάλυσης συναισθήματος στα επιστρέφοντα αποτελέσματα της αναζήτησης. Δηλαδή παρουσιάζονται στον χρήστη εκτός από τα tweets τα οποία πληρούν τους όρους της αναζήτησης και μια γραφική παράσταση, η οποία απεικονίζει το συνολικό συναίσθημα στο σύνολο των αποτελεσμάτων σε ποσοστό επί τοις εκατό. Έτσι υπάρχουν δυο γραφικές μπάρες, οι οποίες δείχνουν κατά πόσο τοις εκατό ήταν αρνητικό ή θετικό το αποτέλεσμα της ανάλυσης συναισθήματος. Για τον υπολογισμό των αποτελεσμάτων χρησιμοποιούνται δύο διαφορετικά API, ώστε να αφεθεί στον χρήστη η επιλογή της μηχανής ανάλυσης συναισθήματος που επιθυμεί. Παρακάτω παρουσιάζονται αυτά τα δύο API.

uclassify

Για την ανάλυση συναισθήματος ενός κείμενου από το Uclassify [10] στέλνονται Httpwebrequest σε αυτό. Τα δεδομένα εκπαίδευσης (training data) αποτελούνται από 2.8 εκατομμύρια αρχεία, τα οποία είναι μηνύματα από το Twitter, αξιολογήσεις προϊόντων από το Amazon, καθώς και κριτικές ταινιών από το RotenTomatos. Τα έγγραφα αυτά ταξινομούνται σε δύο τάξεις , θετικές και αρνητικές. Αυτό το dataset έχει δημιουργηθεί από τον Mark Dredze και το the John Hopkins University (JohnBlitzer, MarkDredze, FernandoPereira. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. Association of Computational Linguistics (ACL), 2007) και έχει χρησιμοποιηθεί και σε άλλες επιστημονικές δημοσιεύσεις. Μερικές από αυτές είναι :

- John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jenn Wortman. Learning Bounds for Domain Adaptation. Neural Information Processing Systems (NIPS), 2008.

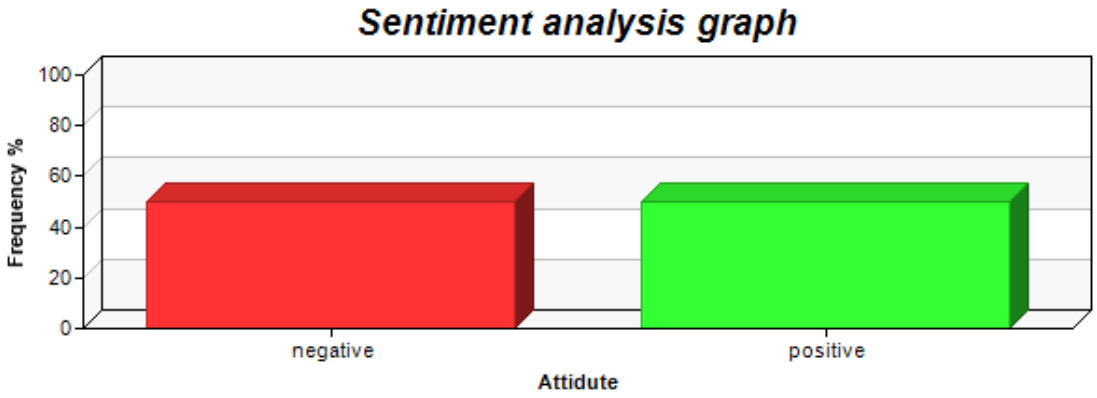
- Mark Dredze, Koby Crammer, and Fernando Pereira. Confidence-Weighted Linear Classification. International Conference on Machine Learning (ICML), 2008.
- Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain Adaptation with Multiple Sources. Neural Information Processing Systems (NIPS), 2009.

Σε αυτό έχει ενσωματωθεί ένας mood classifier, ο οποίος βελτιώνει τα ήδη υπάρχοντα δεδομένα. Σύμφωνα με το uclassify η αναμενόμενη ακρίβεια είναι περίπου 83 %, ενώ για τη λειτουργία επικύρωσης χρησιμοποιείται η μέθοδος <<10 fold cross validation>>.

Για να καλέσει η εφαρμογή το συγκεκριμένο API δίνεται ως είσοδο στο HTTPwebRequest ένα κείμενο που αποτελείται από τη σύνδεση του κειμένου των tweets από τα αποτελέσματα της αναζήτησης. Η διαδικασία της δημιουργίας ενός κειμένου από τη σύνδεση των tweets είναι απαραίτητη, καθώς το uclassify δεν προσφέρεται για μαζική αποστολή και ανάλυση ξεχωριστών μηνυμάτων tweets. Το response το οποίο λαμβάνεται έχει την παρακάτω xml μορφή.

```
<?xml version="1.0" encoding="UTF-8" ?>
<uclassify xmlns="http://api.uclassify.com/1/ResponseSchema" version="1.01">
        <status success="true" statusCode="2000"/>
        <readCalls>
        <classify id="cls1">
                <classification textCoverage="0.715116">
                        <class className="negative" p="0.0281803"/>
                        <class className="positive" p="0.97182"/>
                </classification>
        </classify>
        </readCalls>
</uclassify>
```

Όπως φαίνεται παραπάνω, επιστρέφονται δυο τιμές, μια τιμή που δείχνει κατά πόσο αρνητικό είναι ένα κείμενο (<className="negative" p="0.0281803"/>) και μια που δείχνει κατά πόσο θετικό είναι (<className="positive" p="0.97182"/>), έχοντας ως μεγίστη τιμή το 1 και ελάχιστη το 0. Για να παρουσιαστούν οι τιμές στην γραφική παράσταση τις πολλαπλασιάζουμε επί τοις 100, ώστε να επιστρέψουμε ποσοστά.



Εικόνα 11 Uclassify Γραφική παράσταση

Sentiment140

Ως εναλλακτική μηχανή ανάλυσης συναισθήματος χρησιμοποιείται το Sentiment140 API [6]. Το συγκεκριμένο API είναι σχεδιασμένο για να αναλύει το συναίσθημα σε microblogging υπηρεσίες, όπως το Twitter, καθώς τα δεδομένα εκπαίδευσης (training data) αποτελούνται από tweets. Σύμφωνα με το [7] το Sentiment140 API προσφέρει ακρίβεια στην ανάλυση συναισθήματος ανώτερης της τάξεως του 80%. Επιπλέον προσφέρει τη δυνατότητα για μαζική αποστολή και ανάλυση μοναδικών tweets (έως 10.000) σε ένα request σε αντίθεση με το uClassify το οποίο δεν προσέφερε αυτή την δυνατότητα. Η εφαρμογή καλεί το API αποστέλλοντας ένα json μήνυμα της παρακάτω μορφής που αποτελείται από τα επιστρέφοντα tweets της αναζήτησης.

{"data":[{"text":"Good morning Sun!! @ London Bridge https://t.co/dm3S7afYFg","polarity":0},{"text":"Crowd funding bid launched to get

WW1 song recorded http://t.co/VUSsddL9an #London
http://t.co/9yPEOEAyfp","polarity":0}]}

Το Sentiment140 API πραγματοποιεί Classification σε αρνητικά, θετικά ή ουδέτερα συναισθήματα από ένα συνδυασμό από machine learning Classifiers, όπως Naïve Bayes, Maximum Entropy (MaxEnt) και Support Vector Machines (SVM) χρησιμοποιώντας τα χαρακτηριστικά των Tweets. Η εξαγωγή αυτών των χαρακτηριστικών γίνεται χρησιμοποιώντας unigrams (π.χ Good, Bad) ,bigrams (π.χ Not good, Not bad), unigrams και bigrams,καθώς και μέρη του λογού. Ο αλγόριθμος που χρησιμοποιείται από τους machine learning classifiers χρησιμοποιεί ένα training set με Tweets τα οποία περιέχουν emoticons. Για παράδειγμα, ☺ σε ένα tweet παρουσιάζει ότι το συγκείμενο tweet περιέχει θετικό συναίσθημα και ☹ παρουσιάζει ότι το tweet περιέχει αρνητικό συναίσθημα [7]. Η χρήση hand-labeled training set θα ήταν αρκετά χρονοβόρα και δύσκολη σε ένα τεράστιο εύρος θεμάτων που υπάρχει στο twitter,επομένως η χρήση emoticons είναι ένας έξυπνος τρόπος για την δημιουργία του training set.

Το αποτέλεσμα του API είναι της παρακάτω μορφής.

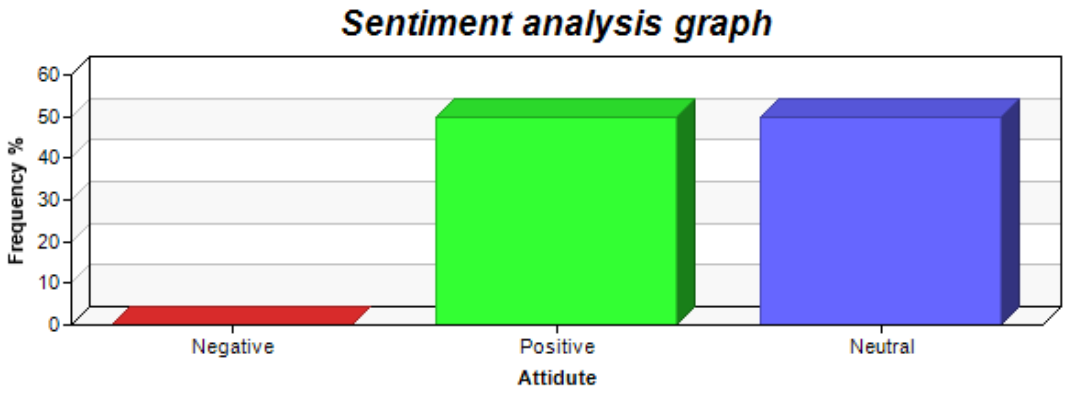
{"data":[{"text":"Good morning Sun!! @ London Bridge
https://t.co/dm3S7afYFg","polarity":4,"meta":{"language":"en"}},{"text":"Crowd
funding bid launched to get WW1 song recorded http://t.co/VUSsddL9an #London
http://t.co/9yPEOEAyfp","polarity":2,"meta":{"language":"en"}}]}

Οι τιμές οι οποίες αναφέρονται στο συναίσθημα ενός tweet (polarity) είναι:

- **0** : αρνητικό συναίσθημα
- **2** : ουδέτερο συναίσθημα
- **4** : θετικό συναίσθημα

Έπειτα, για την δημιουργία της γραφικής παράστασης η οποία θα αναφέρεται στο συνολικό συναίσθημα των tweets υπολογίζεται ο μέσος όρος επί τοις 100 για το κάθε polarity. Ο τύπος που δίνει αυτή την τιμή είναι ο ακόλουθος :

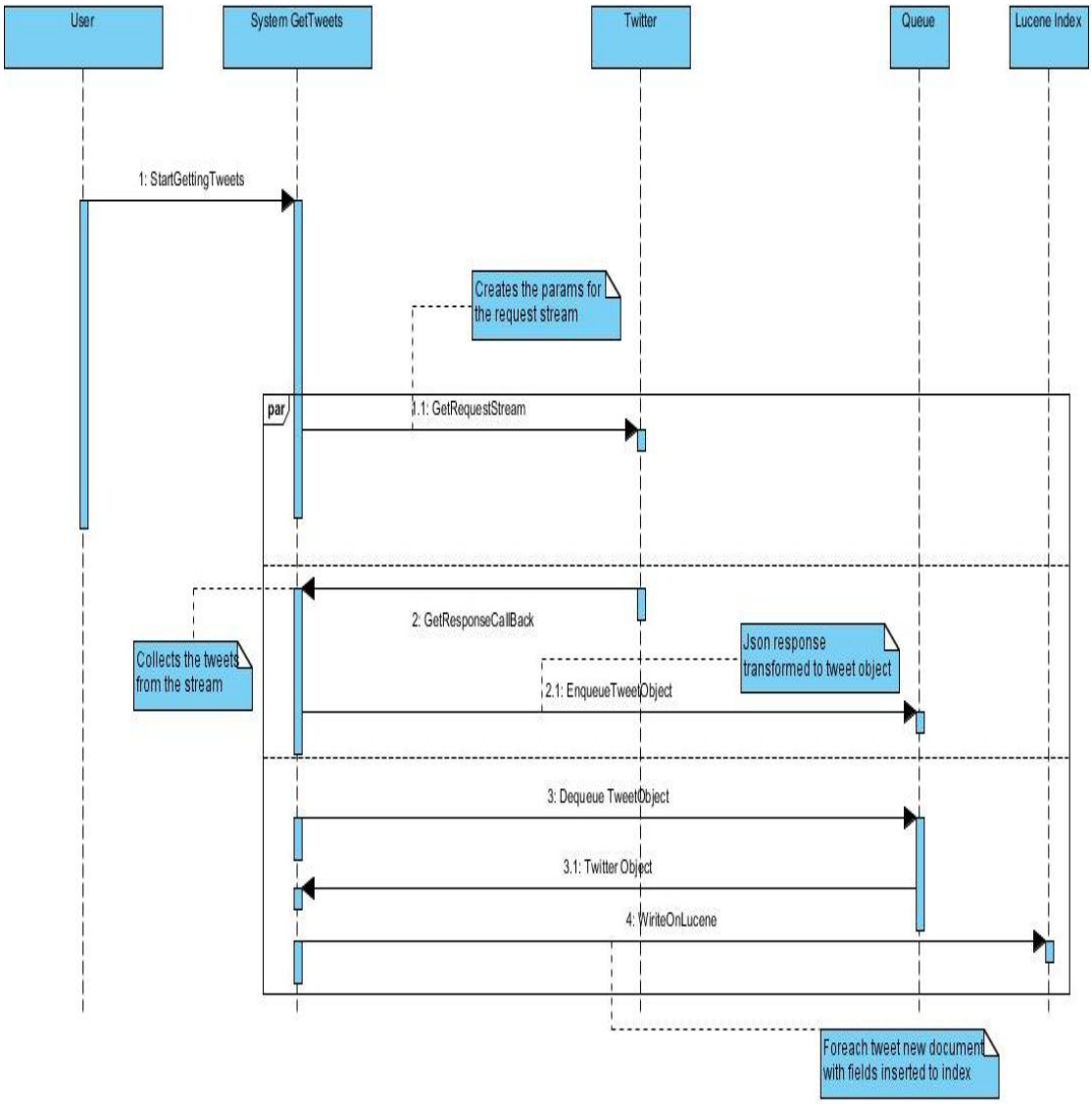
(TweetsCountWithPolarity/TotalTweets)*100



Εικόνα 12 Sentiment140 Γραφική παράσταση

Multithreading

Ένα κύριο χαρακτηριστικό που έχει η εφαρμογή είναι ότι θα πρέπει να λειτουργεί σε πραγματικό χρόνο. Αυτή η απαίτηση μας οδήγησε στη δημιουργία μιας εφαρμογής που εκμεταλλεύεται τη χρήση πολλαπλών πυρήνων λειτουργιάς (MultiThreding). Η αρχιτεκτονική του συστήματος για τη λήψη των tweets, καθώς και την επεξεργασία και τη δημιουργία του Index περιγράφεται στο παρακάτω σχεδιάγραμμα.



Εικόνα 3 Συλλογή Tweets αρχιτεκτονική

Εσωτερικά του πλαισίου με την ένδειξη "par", το οποίο ονομάζεται έτσι λόγω του parallel (δηλαδή ταυτόχρονη παράλληλη λειτουργία) έχουμε κατηγοριοποιήσει την εφαρμογή σε τρία επιπλέον threads, εκτός του κυρίως thread που είναι υπεύθυνο για το GUI.

Thread 1

Το συγκείμενο thread είναι υπεύθυνο για τη δημιουργία των παραμέτρων, απαραίτητων στην κλήση του Streaming API του twitter. Δηλαδή τη δημιουργία των καταλλήλων χαρακτηριστικών του OAuth και την ενσωμάτωση αυτών ως Headers στο HTTPWebRequest προς το Streaming API, όπως περιγράφηκε προηγουμένως.

Thread 2

Σε ένα άλλο διαφορετικό thread συγκεντρώνεται η ροή των μηνυμάτων tweets και εισχωρούνται μετατρέποντας τα πρώτα σε ειδικά προσαρμοσμένα αντικείμενα, τα οποία και ονομάζουμε RootObject, σε μια δομή τύπου ουράς FIFO.

Thread 3

Το τρίτο Thread αναλαμβάνει την επεξεργασία των tweets και τη δημιουργία του Index. Δηλαδή εξυπηρετεί τη δομή τύπου ουράς και αναλύει και επεξεργάζεται το κάθε tweet δημιουργώντας Documents και Fields απαραίτητα για το Index. Έπειτα δημιουργεί ή ανανεώνει το Index χρησιμοποιώντας το StandarAnalyzer για την μετατροπή strings σε tokens.

Εκτός από τα προαναφερθέντα threads,τα οποία και λειτουργούν σαν ατέρμονος βρόγχος, χρησιμοποιείται άλλο ένα thread το οποίο είναι υπεύθυνο για δημιουργία των γραφικών παραστάσεων που παρουσιάζουν τα δέκα πιο διαδεδομένα χρησιμοποιούμενα Hashtags στο σύνολο των καταγεγραμμένων και τις δέκα πιο διαδεδομένες γλώσσες που έχουν χρησιμοποιηθεί για τη σύνταξη των tweets. Αυτό το thread ενεργοποιείται κάθε μισό λεπτό, ώστε να ανανεώνει τα αποτελέσματα.

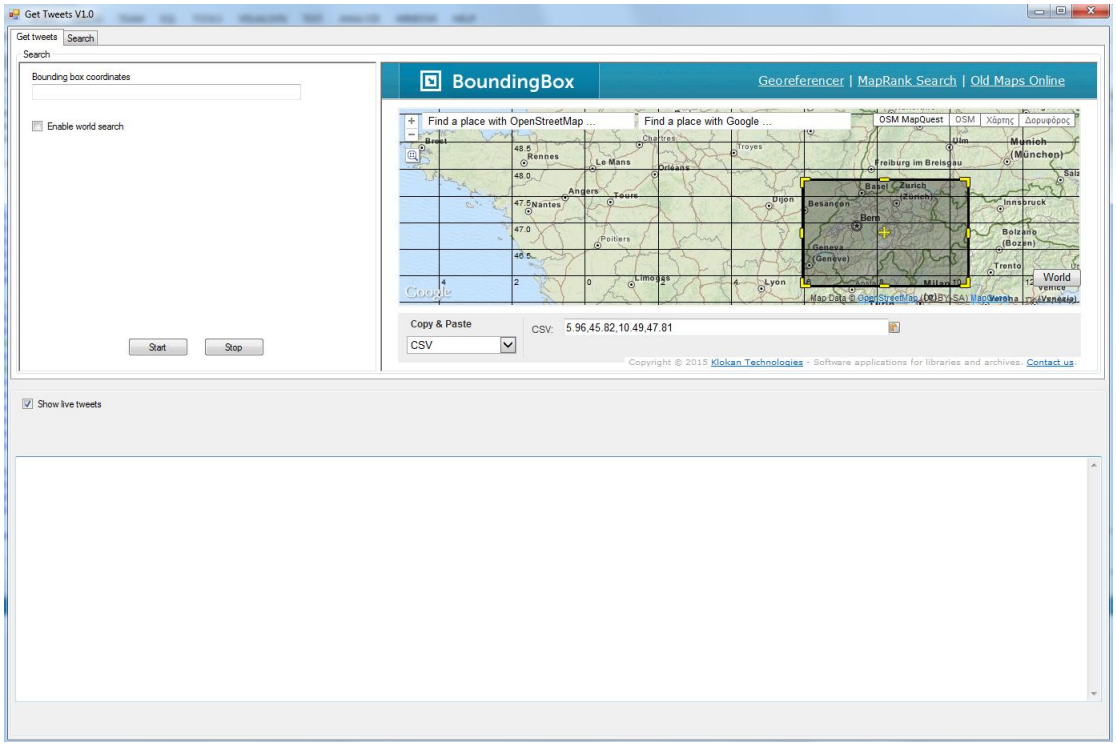
Εργαλεία και τεχνολογίες.

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκαν μερικά από τα πιο σύγχρονα εργαλεία, όπως η βιβλιοθήκη της Microsoft .NET Framework και ως γλώσσα προγραμματισμού η C#. Η εφαρμογή είναι τύπου WinForm Application. Ως ολοκληρωμένο περιβάλλον ανάπτυξης (integrated development environment, IDE) χρησιμοποιήσαμε το Visualstudio 2013. Για την επίτευξη του σκοπού της εργασίας βασιστήκαμε επιπλέον σε μερικές βιβλιοθήκες όπως :

- **Lucene.Net**:Apache Software Foundation Lucene .Net πλήρους κειμένου βιβλιοθήκη μηχανή αναζήτησης. Χρησιμοποιείται κυρίως στην εφαρμογή για το Indexing και την αναζήτηση. Για το Lucene υπάρχουν εκτενείς αναφορές στα προηγούμενα κεφάλαια.
- **NetChartDir**: Βιβλιοθήκη για γραφικές παραστάσεις στο .NET. Χρησιμοποιείται για την αναπαράσταση τριών γραφημάτων στην παρούσα εφαρμογή. Δυο γραφήματα πίτας που απεικονίζουν τα 10 πιο διαδεδομένα Hashtags/Γλώσσα και ένα ακόμα γράφημα για το αποτέλεσμα της ανάλυσης συναισθήματος.
- **Newtonsoft.Json :** JSON framework για .NET. Χρησιμοποιείται για την μετατροπή του μηνύματος tweet που στέλνεται από το Twitter σε object και το αντίστροφο.

Κεφάλαιο 6: Οδηγός χρήσης εφαρμογής.

Σε αυτό το κεφάλαιο παρουσιάζονται οδηγίες για την ορθή χρήση της εφαρμογής. Η κεντρική οθόνη της εφαρμογής αποτελείται από δυο Tabs, το <<Get tweets>> και το <<Search>>. Το πρώτο tab χωρίζει την οθόνη της εφαρμογής σε 3 πεδία, στο πάνω δεξιά πεδίο, όπως φαίνεται στην παρακάτω εικόνα, απεικονίζεται ένας γεωγραφικός χάρτης στον οποίο ο χρήστης ορίζει την γεωγραφική περιοχή για την καταγραφή των μηνυμάτων tweets.

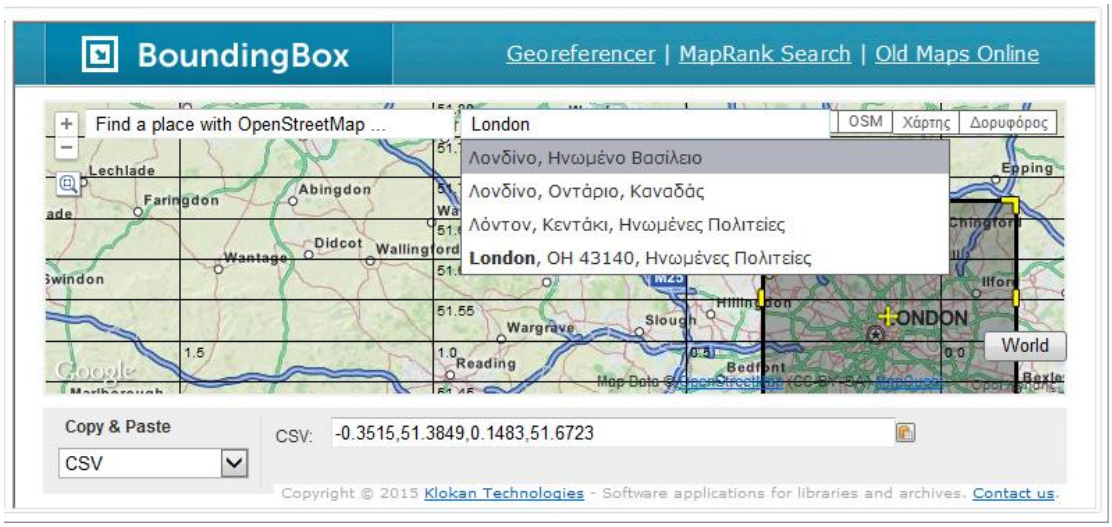


Εικόνα 13 Αρχική οθόνη εφαρμογής με καταγραφή tweets

Υπάρχουν 2 τρόποι για να επιλέξουμε μια γεωγραφική περιοχή.

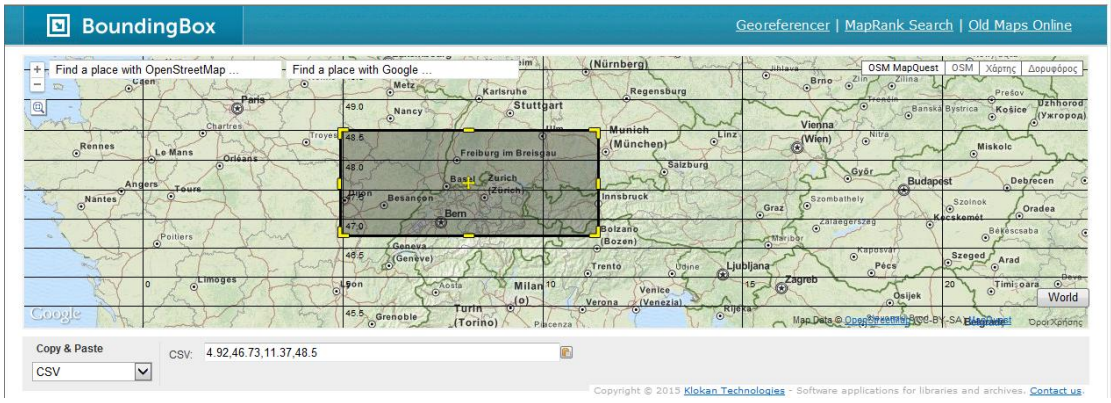
1. Αναζήτηση τοποθεσίας από τα 2 textboxes. Μπορούμε να αναζητήσουμε οποιαδήποτε τοποθεσία μας ενδιαφέρει συμπληρώνοντας το όνομά της, χρησιμοποιώντας είτε τα στοιχεία του OpenStreetMap είτε της Google. Για παράδειγμα, έστω ότι θέλουμε να αναζητήσουμε Tweets από το Λονδίνο,

επιλέγοντας την τοποθεσία <<London>> αυτόματα ένα Bounding Box σχεδιάζεται στο χάρτη.



Εικόνα 14 Επιλογή γεωγραφικού χώρου 1

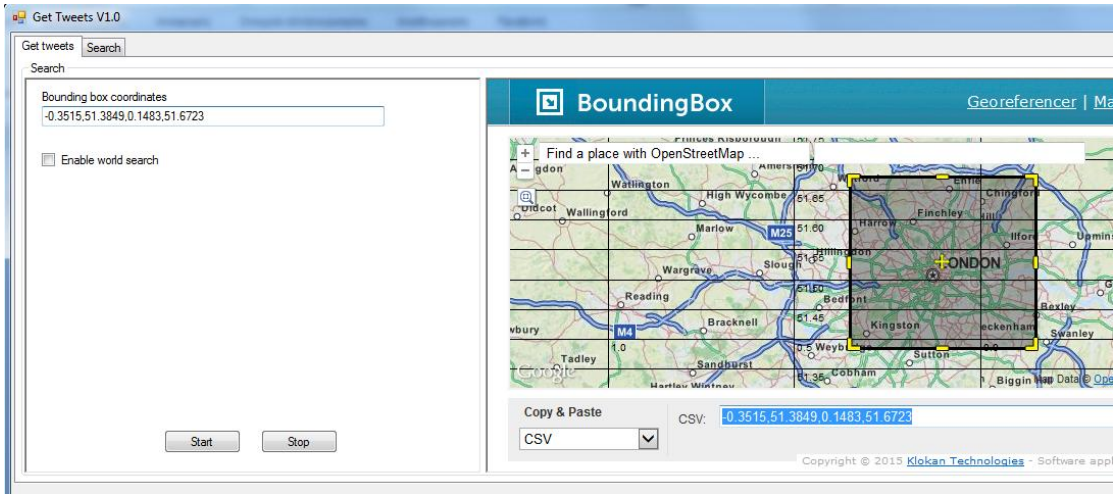
2. Αλλάζοντας το μέγεθος του Bounding Box από τον χάρτη μπορούμε να ορίσουμε τον χώρο της αναζήτησης.



Εικόνα 4 Επιλογή γεωγραφικού χώρου 2

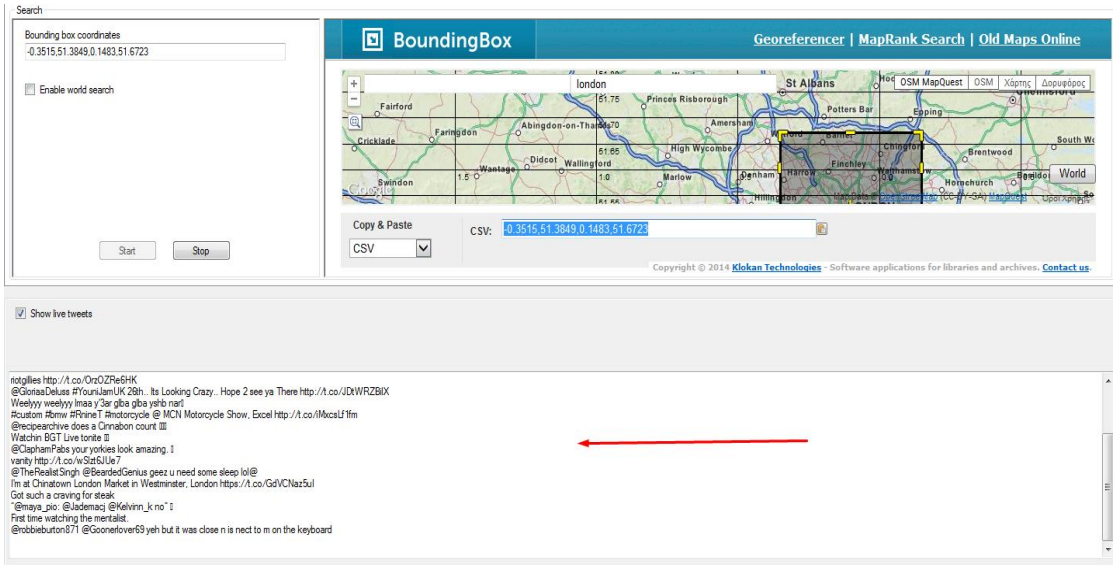
Αφού επιλεχτεί ο επιθυμητός γεωγραφικός χώρος, στο αριστερό μέρος της οθόνης υπάρχει ένα πεδίο που ονομάζεται<<Bounding box coordinates>>, το οποίο δέχεται ως τιμή τις συντεταγμένες που περικλείουν το επιλεγμένο bounding box. Επιπλέον δίνεται η δυνατότητα στο χρήστη πατώντας το κουμπί <<Enable world

search>> να καταγράψει παγκοσμίως τα tweets αγνοώντας έτσι κάποια συγκεκριμένη γεωγραφική περιοχή, καθώς πλέον ο επιλεγμένος χώρος για καταγραφή tweets είναι ολόκληρη η γη.



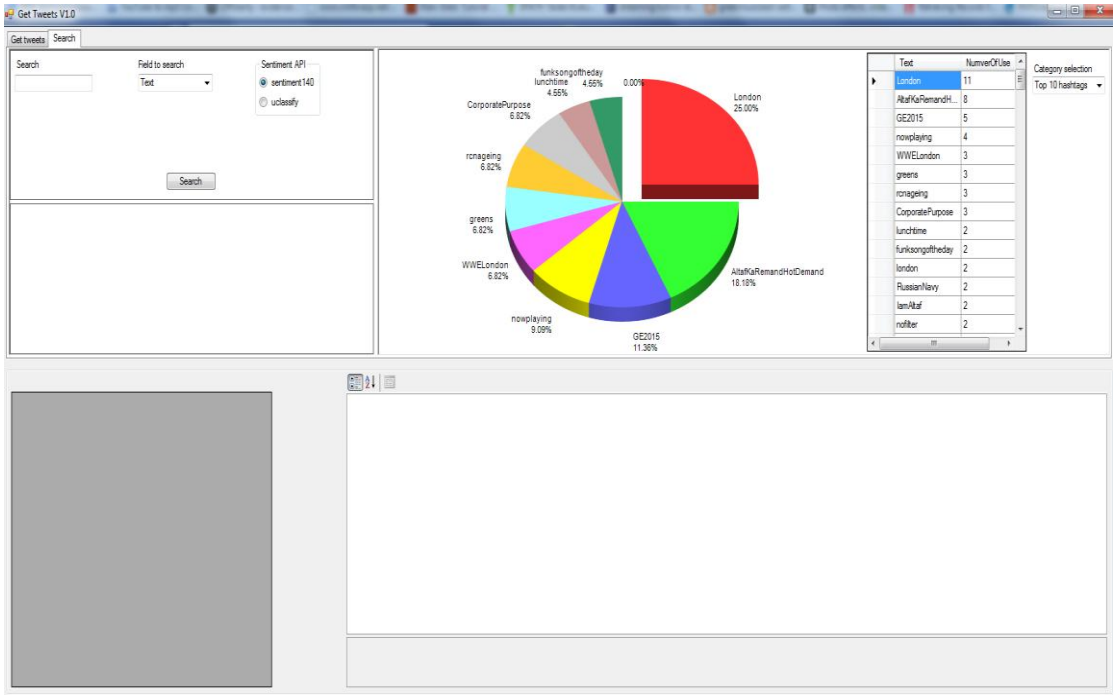
Εικόνα 5 Bounding box coordinates

Πατώντας το κουμπί <<Start>> η εφαρμογή ξεκινά να λαμβάνει και να επεξεργάζεται tweets δημιουργώντας το Index. Δηλαδή εσωτερικά η εφαρμογή, αφού λάβει τα μηνύματα τα επεξεργάζεται δημιουργώντας Documens που αποτελούνται από fields και εισχωρεί αυτά στο Index. Το κείμενο των εισερχόμενων tweets μπορεί να εμφανιστεί ή όχι στο κάτω πεδίο της οθόνης επιλέγοντας την επιλογή << Show live tweets >>. Ο χρηστής ανά πασά στιγμή μπορεί να πατήσει το κουμπί <<Stop>> για να σταματήσει η λήψη νέων μηνυμάτων. Παρόλα αυτά μπορεί, ακόμα και αν έχει πατήσει το κουμπί stop, να επεξεργαστεί και να κάνει αναζητήσεις στα ήδη υπάρχοντα ληφθέντα Tweets.



Εικόνα 6 Show live tweets

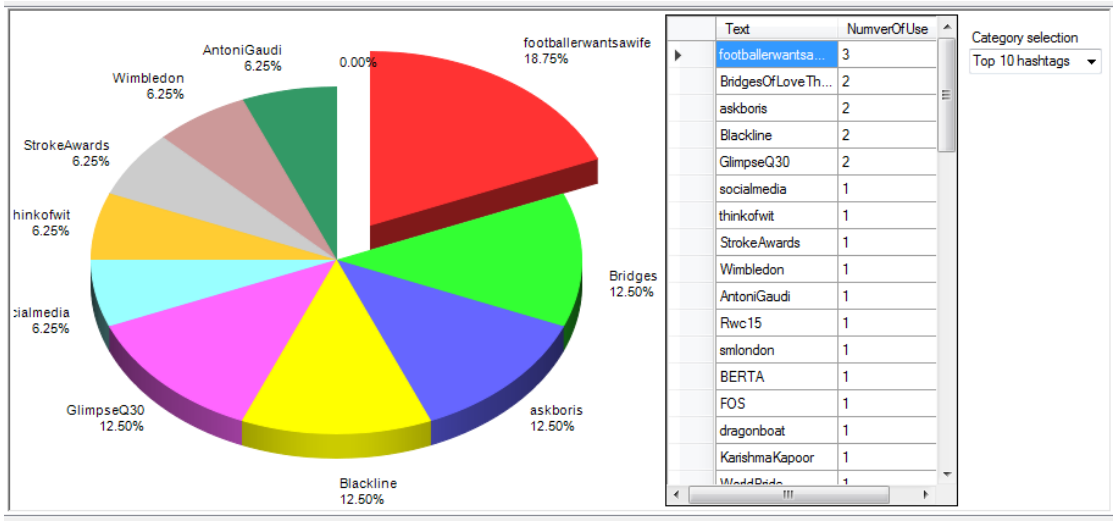
Στο δεύτερο tab της εφαρμογής ο χρήστης μπορεί να κάνει αναζητήσεις στο index της εφαρμογής, αλλά και να δει στατιστικά γραφήματα.



Εικόνα 7 Επεξεργασία Index

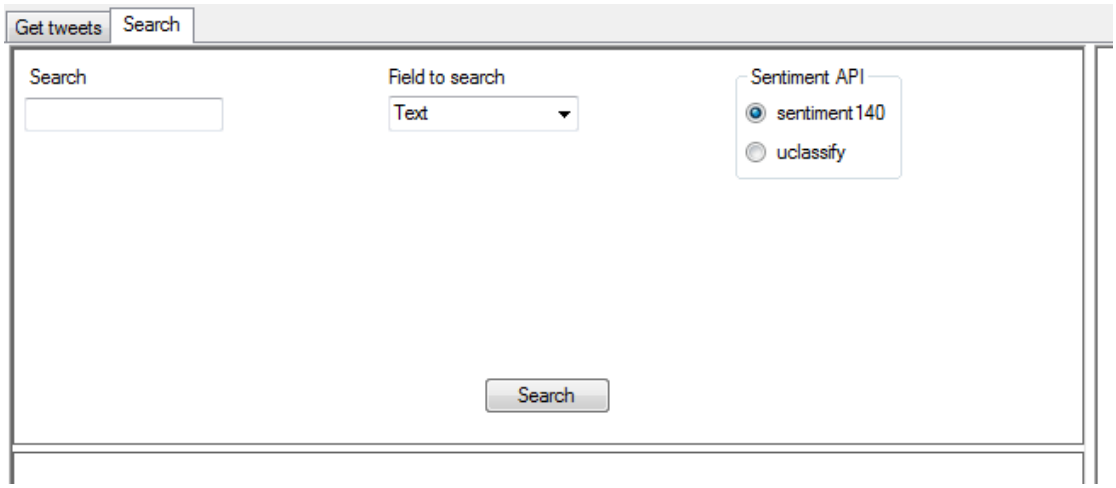
Ο χρηστής μπορεί να επιλέξει από το πεδίο <<Category selection>> ανάμεσα στις επιλογές, <<Top 10 hashtags>> και <<Top 10 languages used>>. Η επιλογή

Top10 hashtags παρουσιάζει τα δέκα πιο διαδεδομένα χρησιμοποιούμενα Hashtags στο σύνολο των tweets. Το κείμενο που περιγράφει το hashtag, καθώς και πόσες φόρες έχει χρησιμοποιηθεί παρουσιάζεται στο μεσαίο Grid. Η γραφική παράσταση σε μορφής πίτας παρουσιάζει το ποσοστό χρησιμοποίησης αυτών ανάμεσα στα 10 πιο διαδεδομένα tweets. Αντίστοιχα η επιλογή <<Top 10 languages used>> παρουσιάζει τις δέκα πιο διαδεδομένες γλώσσες που έχουν χρησιμοποιηθεί για τη σύνταξη των tweets. Καθώς η εφαρμογή λειτούργει σε πραγματικό χρόνο αυτά τα αποτελέσματα ανανεώνεται αυτόματα κάθε μισό λεπτό, ώστε να συμπεριληφθούν νέα tweets.



Εικόνα 8 Γραφική παράσταση Hashtags

Τα εργαλεία για αναζητήσεις, καθώς και η ανάλυση συναισθήματος παρέχονται στο αριστερό πεδίο της οθόνης.



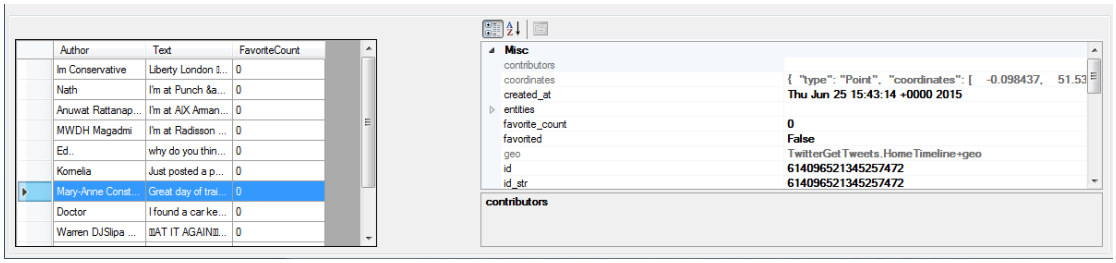
Εικόνα 9 Αναζήτηση Index

Αναζήτηση μπορεί να εφαρμοστεί σε διαφορετικά πεδία ενός Tweet επιλέγοντας το μενού από το πεδίο <<Field to search>>. Τα πεδία που είναι διαθέσιμα προς αναζήτηση είναι :

- Text : Αναζήτηση στο κείμενο ενός Tweet.
- Language : Αναζήτηση σύμφωνα με τη γλώσσα που χρησιμοποιεί ένα Tweet.
- UserName : Αναζήτηση σύμφωνα με το UserName του συγγραφέα του Tweet.
- PlaceName : Αναζήτηση σύμφωνα με τον τόπο αποστολής του Tweet.
- Hashtags : Αναζήτηση στα Hashtags του Tweet.

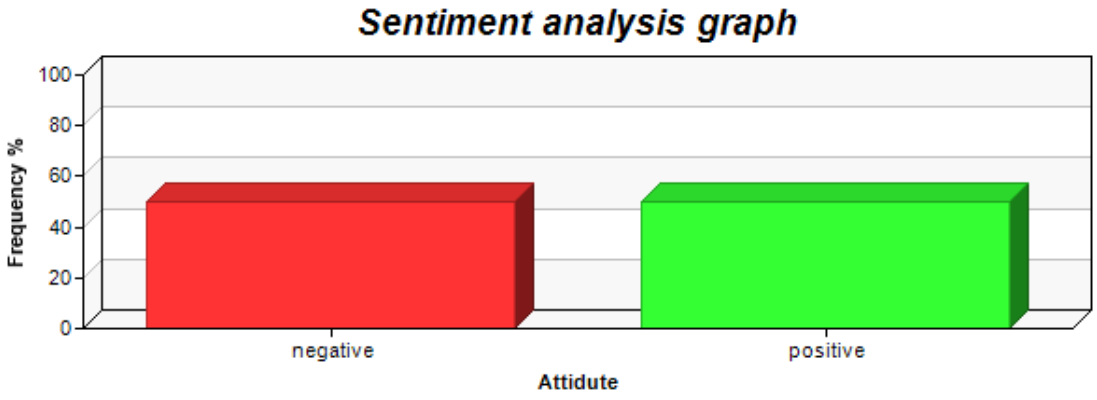
Επίσης δίνεται η δυνατότητα στον χρήστη στο πεδίο <<Sentiment API>> να διαλέξει πιο API ανάλυσης συναισθήματος επιθυμεί να χρησιμοποιηθεί μεταξύ του Sentiment140 και του Uclassify. Αφού γίνει η επιλογή του κατάλληλου πεδίου, καθώς και του API που θα χρησιμοποιηθεί για την ανάλυση συναισθήματος εισάγεται στο πεδίο <<Search>> το ερώτημα σε μορφή κειμένου. Ο χρηστής σε αυτό το πεδίο μπορεί να κάνει σύνθετα ερωτήματα, παράδειγμα των οποίων παρουσιάζεται στον πινάκα 9. Τελικώς, πατώντας το κουμπί <<Search>>, η εφαρμογή αναζητά και επιστρέφει τα Tweets του Index τα οποία ταιριάζουν στο ερώτημα. Η παρουσίαση των αποτελεσμάτων αποτελείται από τρία μέρη. Στο κάτω αριστερά μέρος της οθόνης παρουσιάζεται ένας πίνακας, ο οποίος περιέχει τα Tweets που επιστραφήκαν μετά την αναζήτηση. Ο πίνακας αυτός αποτελείται από 3 στήλες. Η πρώτη με τον τίτλο Author αναφέρεται στο όνομα του συγγραφέα του Tweet, η δεύτερη με τον τίτλο Text στο

κείμενο του Tweet και η τρίτη με όνομα Favorite Count στο πόσες φορές έχει επισημανθεί το συγκείμενο Tweet σαν Favorite.



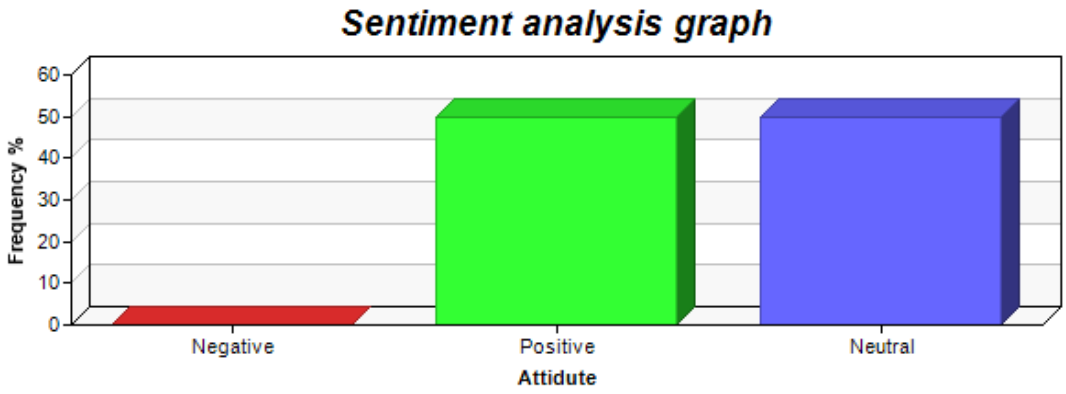
Εικόνα 10 Παρουσίαση αποτελεσμάτων

Επιλέγοντας μια σειρά από αυτόν τον πινάκα παρουσιάζονται στο δεξιά property Grid αναλυτικά όλα τα στοιχεία που αφορούν το επιλεγμένο Tweet. Επίσης παρουσιάζεται μια γραφική αναπαράσταση του συνολικού ποσοστού συναισθήματος για τα tweets τα οποία έχουν επιστραφεί από την αναζήτηση. Η κόκκινη μπάρα με τίτλο Negative αντιπροσωπεύει τα συνολικό ποσοστό του αρνητικού συναισθήματος και η πράσινη με τίτλο Positive αυτή του θετικού ποσοστού.



Εικόνα 11 Uclassify Γραφική παράσταση

Χρησιμοποιώντας το Sentiment140 Api υπάρχει άλλη μια μπάρα με το χρώμα μπλε και τίτλο Neutral η οποία αντιπροσωπεύει το ποσοστό του ουδέτερου συναισθήματος.



Εικόνα 12 Sentiment140 Γραφική παράσταση

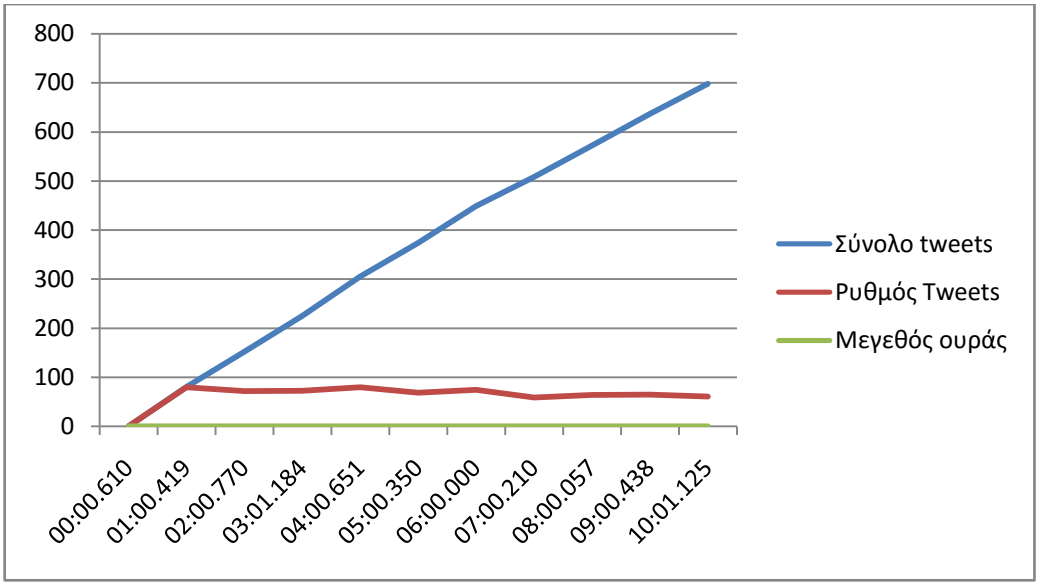
Κεφάλαιο 7 : Πειραματικές μετρήσεις

Σε αυτό το κεφάλαιο θα πραγματοποιηθούν πειραματικές μετρήσεις χρησιμοποιώντας την εφαρμογή, έτσι ώστε να απαντηθούν ερωτήματα, όπως πόσο αποδοτική είναι η αρχιτεκτονική της εφαρμογής, ποιος ο χρόνος μεταξύ της δημιουργίας του Index και της ανάκτησης πληροφορίας από αυτό και να πραγματοποιηθούν συγκρίσεις μεταξύ των δυο μηχανών ανάλυσης συναισθήματος. Οι πειραματικές μετρήσεις πραγματοποιήθηκαν σε υπολογιστή με επεξεργαστή IntelCore i7-2600 CPU @ 3.4 GHz και μνήμη RAM 6 GB. Τα δεδομένα τα οποία χρησιμοποιήθηκαν είναι συλλογή tweets με γεωγραφική περιοχή που περικλείει το Λονδίνο. Ο γεωγραφικός χώρος του Λονδίνου έχει επιλεχτεί, καθώς είναι η τρίτη πόλη με τον μεγαλύτερο αριθμό αποστολής tweets παγκοσμίως συμφώνα με την έρευνα [11] κατέχοντας το 2%. Επιπλέον η χρησιμοποίηση της αγγλικής γλώσσας μας επιτρέπει να επεξεργαστούμε τα αποτελέσματα σωστότερα.

Αξιολόγηση αρχιτεκτονικής συστήματος στην συλλογή tweets.

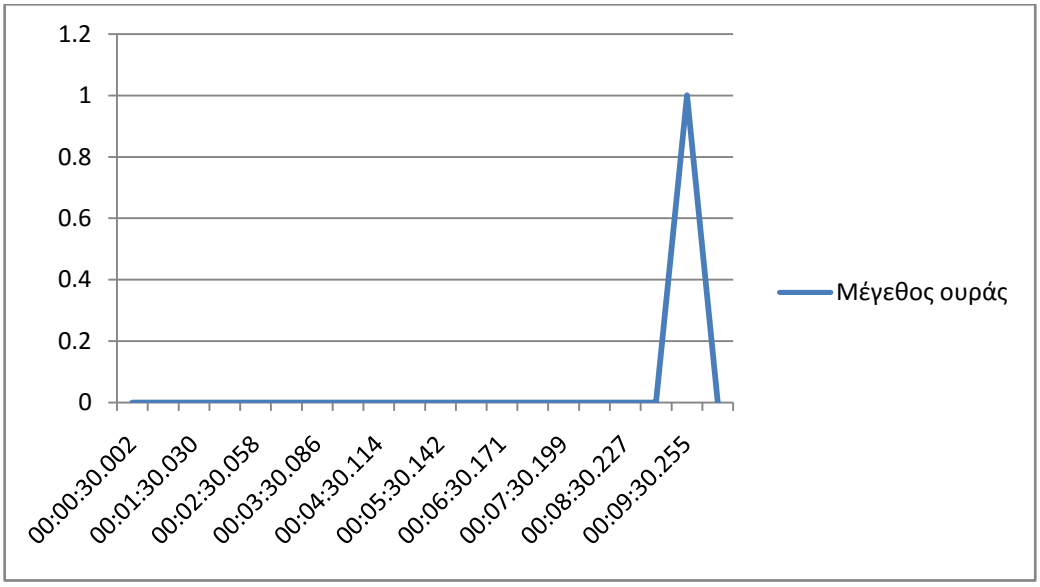
Όπως παρουσιάστηκε στα προηγούμενα κεφάλαια για τη λήψη και επεξεργασία των μηνυμάτων tweets εισήχθη ενδιάμεσα της λειτουργιάς της επεξεργασίας των μηνυμάτων και αυτή του Indexing, μια δομή τύπου ουράς FIFO (First-In-First-Out, Πρώτο-Μέσα-Πρώτο-Έξω). Σε αυτήν την αρχιτεκτονική υπάρχει ο κίνδυνος ο ρυθμός μηνυμάτων που εισέρχονται στη δομή τύπου ουράς να είναι μεγαλύτερος από αυτόν που εξέρχονται με αποτέλεσμα να γεμίσει η μνήμη RAM και η εφαρμογή να αποτύχει. Πειραματικά αποδεικνύεται ότι η εφαρμογή της εργασίας χρησιμοποιεί τη συγκεκριμένη αρχιτεκτονική αποδοτικά, καθώς δεν επιβαρύνει τη μνήμη RAM και έτσι απομακρύνεται και η πιθανότητα να αποτύχει η εφαρμογή .

Αρχικά, δειγματολειπτούμε περίπου κάθε λεπτό και εξετάζουμε ποσά tweets έχει καταγράψει το σύστημα. Παρατηρούμε ότι το σύστημα μπορεί να λαμβάνει περίπου εβδομήντα tweets το λεπτό και ότι στα δέκα λεπτά καταγράφτηκαν 698 tweets. Επιπλέον η δομή ουράς που χρησιμοποιείται έχει μηδενικό μέγεθος.



Εικόνα 13 Συμπεριφορά συστήματος στη λήψη tweets

Στο επόμενο διάγραμμα δειγματολειπτούμε κάθε μισό λεπτό και αναλύουμε μόνο το μέγεθος της ουράς. Παρατηρούμε ότι ακόμα και με μικρότερο χρόνο δειγματοληψίας η ουρά έχει μέγεθος μηδέν.



Εικόνα 14 Ανάλυση δομή ουράς

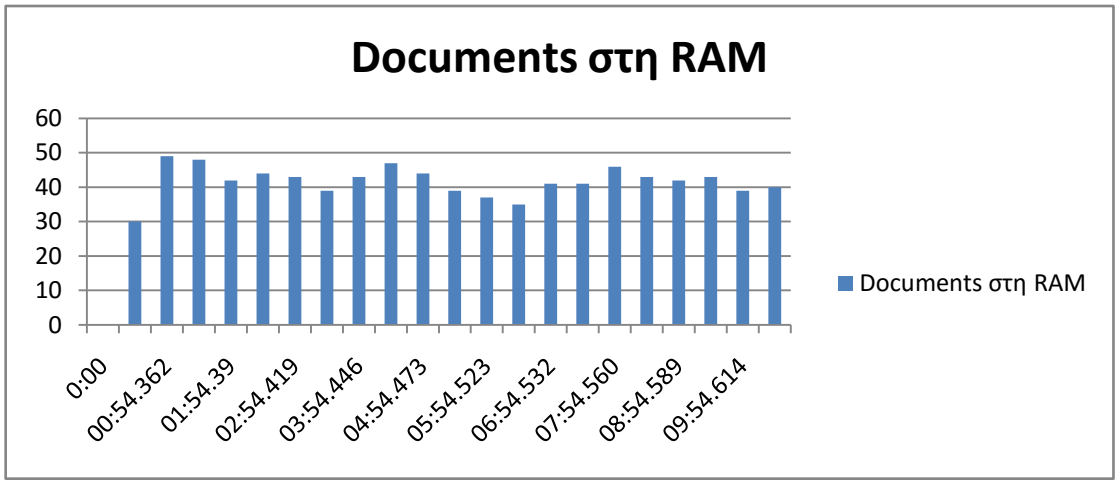
Το συμπέρασμα που καταλήγουμε είναι ότι το σύστημα είναι αποδοτικό, καθώς επεξεργάζεται τη δομή της ουράς χωρίς να επιβαρύνει τη μνήμη RAM και ότι η αρχιτεκτονική που χρησιμοποιείται, η οποία βασίζεται στο ότι ένα threat εισάγει ένα μήνυμα tweet στη δομή ουράς και ένα διαφορετικό thread διαβάζει αυτή την τιμή, είναι σωστή. Παρακάτω εξετάζεται ακόμα πιο αναλυτικά το πώς επιτυγχάνεται μέσω της εναλλαγής διαφορετικών threads να εξυπηρετείται σωστά η δομή ουράς. Στην εικόνα παρουσιάζεται το πώς λειτούργει η δομή ουράς κατά τη λειτουργιά της εφαρμογής. Παρατηρούμε ότι το σύστημα χρειάζεται μόλις 0.1 msec για να εισάγει το μήνυμα στην ουρά. Επιπλέον φαίνεται το πώς ο thread scheduler, αλλά και οι πολλαπλοί επεξεργαστές του υπολογιστή βοηθούν στο να καταμερίζονται οι εργασίες εισαγωγής και εξαγωγής δεδομένων από τη δομή ουράς.

```
Enqued tweet,Time=00:00:02.6105777, NoOfTweets=0
Enqued tweet,Time=00:00:02.6259440, NoOfTweets=1
Enqued tweet,Time=00:00:02.6334240, NoOfTweets=2
dequeued tweet,Time=00:00:02.6361548
Enqued tweet,Time=00:00:02.6996556, NoOfTweets=3
dequeued tweet,Time=00:00:02.7361394
dequeued tweet,Time=00:00:02.8361487
dequeued tweet,Time=00:00:02.9361584
Enqued tweet,Time=00:00:02.9569490, NoOfTweets=4
dequeued tweet,Time=00:00:03.0361653
Enqued tweet,Time=00:00:04.6898843, NoOfTweets=5
dequeued tweet,Time=00:00:04.7362085
Enqued tweet,Time=00:00:05.5159680, NoOfTweets=6
dequeued tweet,Time=00:00:05.5363215
Enqued tweet,Time=00:00:06.8509530, NoOfTweets=7
dequeued tweet,Time=00:00:06.9363703
Enqued tweet,Time=00:00:07.0799959, NoOfTweets=8
dequeued tweet,Time=00:00:07.1363669
Enqued tweet,Time=00:00:07.2670303, NoOfTweets=9
dequeued tweet,Time=00:00:07.3363336
Enqued tweet,Time=00:00:07.6967159, NoOfTweets=10
dequeued tweet,Time=00:00:07.7363293
Enqued tweet,Time=00:00:08.6259956, NoOfTweets=11
dequeued tweet,Time=00:00:08.6363885
Enqued tweet,Time=00:00:10.2619977, NoOfTweets=12
dequeued tweet,Time=00:00:10.3364966
```

Εικόνα 15 Δομή ουράς κατά την λειτουργιά της εφαρμογής

Μελέτη χρόνου write/read στο Lucene Index

Σε αυτό το πείραμα θα εξετάσουμε το τι συμβαίνει στο Lucene Index κατά τη διάρκεια της δημιουργίας του, αλλά και κατά τη διάρκεια της αναζήτησης δεδομένων από αυτό. Το κάθε μήνυμα tweet αντιστοιχεί σε ένα Lucene document, κατά το διάστημα λειτουργίας της εφαρμογής τα μηνύματα τα οποία εξάγονται από τη δομή ουράς επεξεργάζονται και εισάγονται στο Index. Τα documents αυτά δεν εξάγονται απευθείας στο index,αλλά κρατούνται στο Buffer του IndexWritter. Στην εφαρμογή μας ο buffer καταχωρεί τα Documents στο index κάθε φορά που χρησιμοποιούμε το index για αναζητήσεις. Αυτό συμβαίνει κάθε μισό λεπτό, καθώς ανανεώνουμε τις γραφικές παραστάσεις που παρουσιάζονται στην εφαρμογή. Στο παρακάτω διάγραμμα εξετάζουμε πόσα Documents υπάρχουν στη RAM (Buffer) για κάθε μισό λεπτό που ανανεώνουμε τις γραφικές παραστάσεις.



Εικόνα 16 Αριθμός Documentsστο IndexBuffer

Παρατηρούμε ότι κατά μέσο όρο βρίσκονται 42 documents στο buffer του indexWritter σε κάθε χρονικό παράθυρο του μισού λεπτού. Επιπλέον ο μικρότερος αριθμός documents βρίσκεται κατά την πρώτη φορά εισαγωγής ενός document στο Index. Εξετάζοντας το χρόνο που χρειάζεται η εντολή **p_Indexwriter.AddDocument(document);** για να εκτελεστεί παρατηρούμε ότι ο μεγαλύτερος χρόνος παρουσιάζεται κατά την πρώτη φορά της εισαγωγής ενός document στο indexWritter. Πιο συγκριμένα βλέπουμε ότι το σύστημα χρειάζεται 42

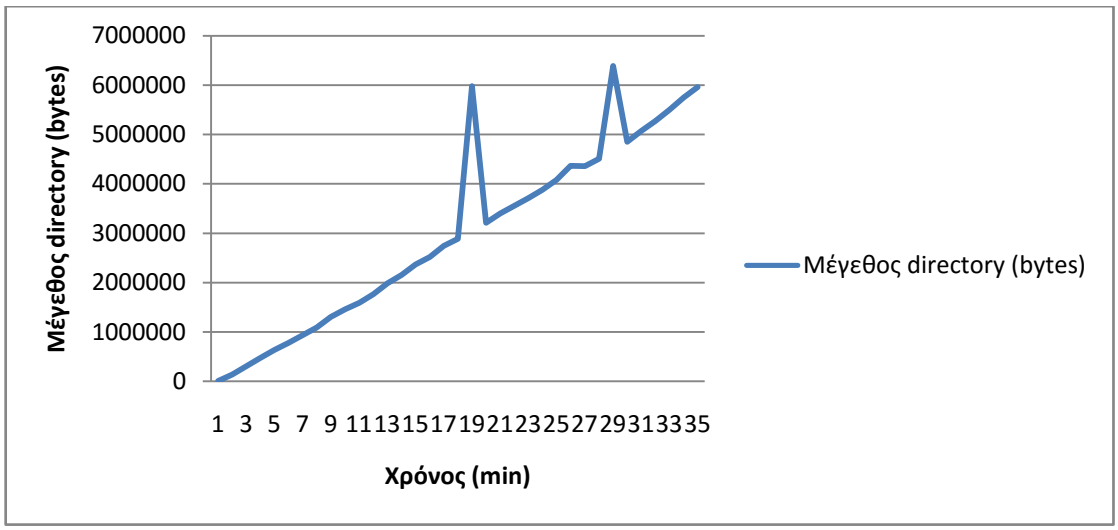
msec την πρώτη φορά και 0.5 msec τη δεύτερη καταγράφοντας μείωση 98.81 % . Στην τρίτη εισαγωγή ο χρόνος μειώνεται ακόμα περισσότερο στα 0.1 msec καταγράφοντας μείωση 99 % από το χρόνο εκτέλεσης της εισαγωγής του πρώτου document.

```
Reader is called, Docs on buffer=0, Time=00:00:00
Time to create a document= 00:00:00.0429072
Time to create a document= 00:00:00.0005813
Time to create a document= 00:00:00.0001376
Time to create a document= 00:00:00.0000811
Time to create a document= 00:00:00.0004506
Time to create a document= 00:00:00.0006112
Time to create a document= 00:00:00.0001325
Time to create a document= 00:00:00.0000914
Time to create a document= 00:00:00.0001264
Time to create a document= 00:00:00.0001316
Time to create a document= 00:00:00.0002381
Time to create a document= 00:00:00.0001596
Time to create a document= 00:00:00.0001141
Time to create a document= 00:00:00.0000763
Time to create a document= 00:00:00.0002106
Time to create a document= 00:00:00.0001883
Time to create a document= 00:00:00.0001820
Time to create a document= 00:00:00.0001343
Time to create a document= 00:00:00.0001162
Time to create a document= 00:00:00.0000908
Time to create a document= 00:00:00.0002641
Time to create a document= 00:00:00.0001056
Time to create a document= 00:00:00.0001005
Time to create a document= 00:00:00.0001608
Time to create a document= 00:00:00.0001765
Time to create a document= 00:00:00.0000941
Time to create a document= 00:00:00.0002128
Time to create a document= 00:00:00.0001343
Time to create a document= 00:00:00.0000781
Time to create a document= 00:00:00.0001255
Reader is called, Docs on buffer=30, Time=00:00:24.3478875
```

Εικόνα 17 Χρόνος εκτέλεσης AddDocument(document);

Στην συνέχεια μελετούμε τη διαδικασία αναζήτησης δεδομένων από το Index. Χρησιμοποιώντας την εφαρμογή, η οποία καταγράφει μηνύματα tweets σε χρονικό ορίζοντα των 35 λεπτών, αναζητούμε tweets τα οποία περιέχουν τη λέξη <<London>>. Η αναζήτηση, καθώς και η δειγματοληψία στην συγκεκριμένη μέτρηση, πραγματοποιείται κάθε ένα λεπτό και μελετούμε το μέγεθος που καταλαμβάνει στο δίσκο το Lucene Index, καθώς και το χρόνο εκτέλεσης της εντολής <<**indexSearch.Search(query, collector);**>>. Για την εύρεση του μεγέθους που καταλαμβάνει το Index αθροίζουμε τα μεγέθη σε Bytes για κάθε αρχείο το οποίο περιέχεται στο Lucene directory της εφαρμογής. Αρχικά μελετάμε το πώς

συμπεριφέρεται το μέγεθος του Index κατά τη διάρκεια του χρόνου. Τα αποτελέσματα παρουσιάζονται στην παρακάτω γραφική παράσταση.

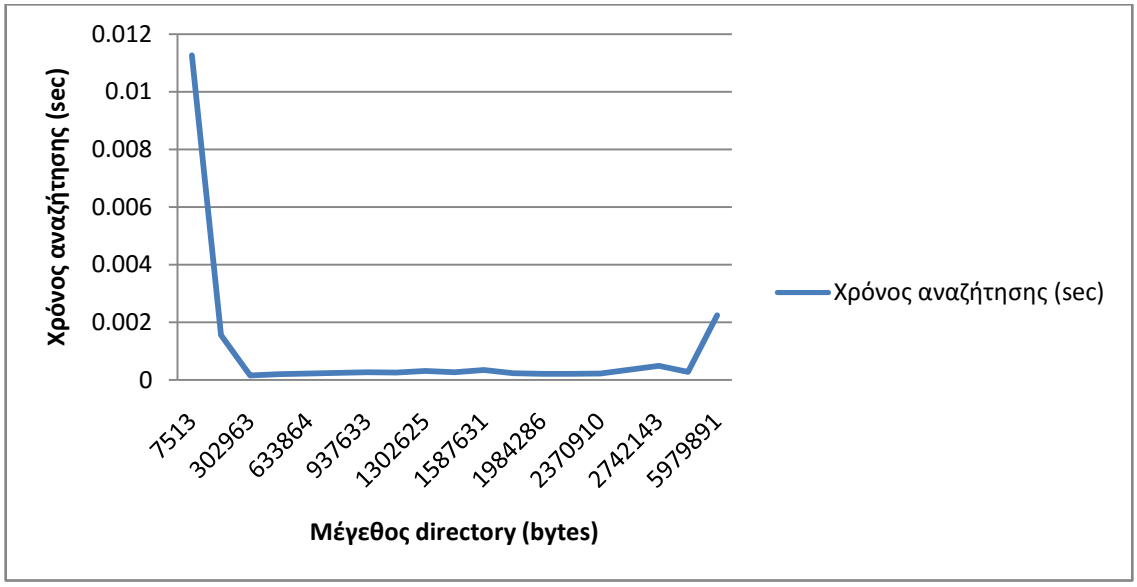


Εικόνα 18 Μέγεθος directory ανά λεπτό

Παρατηρούμε ότι υπάρχει μια αυξητική τάση του μεγέθους του Index το οποίο είναι απολύτως λογικό, καθώς η εφαρμογή κατά τη διάρκεια της διεξαγωγής των μετρήσεων εξακολουθεί να προσθέτει δεδομένα στο Index. Αυτό το οποίο αρχικά φαίνεται περίεργο είναι η απότομη αυξομείωση που υπάρχει σε δυο σημεία της γραφικής παράστασης, η οποία παρουσιάζεται περίπου στο 19 λεπτό και στο 29 λεπτό. Υπεύθυνος για αυτές τις αλλαγές είναι ο MergeScheduler του Lucene. Η εφαρμογή κάθε μισό λεπτό, όπως είδαμε και προηγουμένως καταχωρεί στο index τα Documents που βρίσκονται στο Buffer. Αυτό έχει ως αποτέλεσμα τη δημιουργία καινούργιων segments για κάθε καταχώρηση στο Index. Όταν σε ένα Index υπάρχουν πολλά segments,τότε ο IndexWritter επιλέγει κάποια από αυτά και τα συγχωνεύει σε ένα μεγάλο segment. Το καινούργιο μεγαλύτερο segment χρησιμοποιεί λιγότερα bytes για να παρουσιάσει ακριβώς τα ίδια lucene documents [9]. Η συγκεκριμένη λειτουργιά του Lucene παρουσιάζεται στις αυξομειώσεις που βλέπουμε στη γραφική παράσταση και εκτός από το όφελος της μείωσης του όγκου των δεδομένων, προσφέρει και τη δυνατότητα γρηγορότερης αναζήτησης στο Index, όπως παρουσιάζουμε στις επόμενες γραφικές παραστάσεις.

Χωρίσαμε τις επόμενες γραφικές παραστάσεις σύμφωνα με τις αυξομειώσεις που παρουσιάζονται κατά τη διάρκεια της λειτουργίας της διαδικασίας του

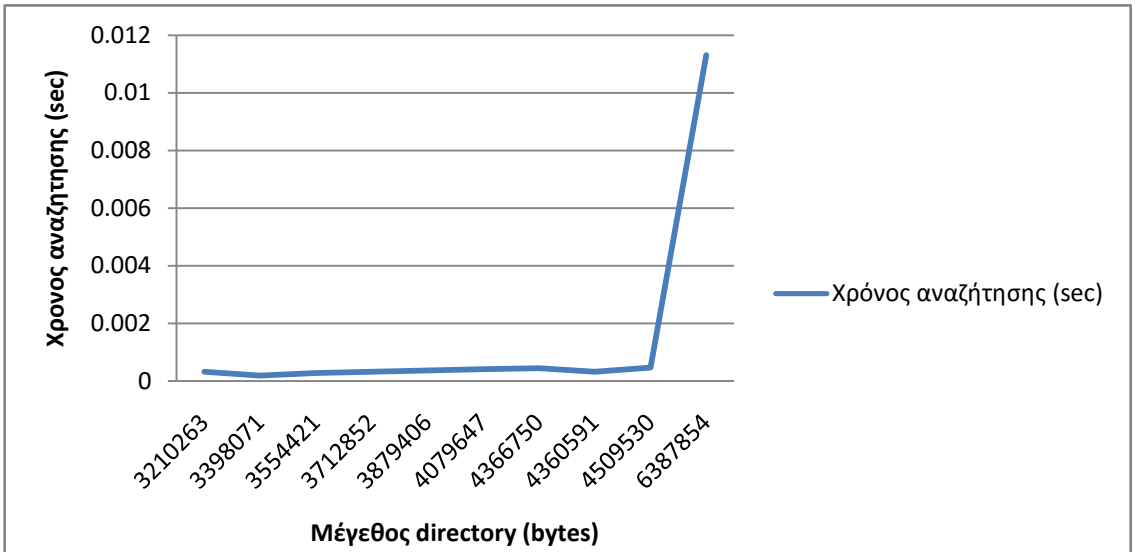
MergeScheduler. Στην πρώτη γραφική παράσταση παρουσιάζουμε το χρόνο που απαιτείται για να αναζητηθεί η λέξη <<London>> από το Index κατά το χρονικό διάστημα 1 έως 19 λεπτά.



Εικόνα 19 Χρόνος αναζήτησης/Μέγεθος Index

Παρατηρούμε ότι ο μέγιστος χρόνος αναζήτησης βρίσκεται κατά τη διάρκεια της πρώτης αναζήτησης απαιτώντας περίπου 0,012 δευτερόλεπτα για μόλις 7513 Bytes. Η επομένη γραφική παράσταση αναφέρεται στο χρονικό διάστημα 19 έως 29 λεπτά.

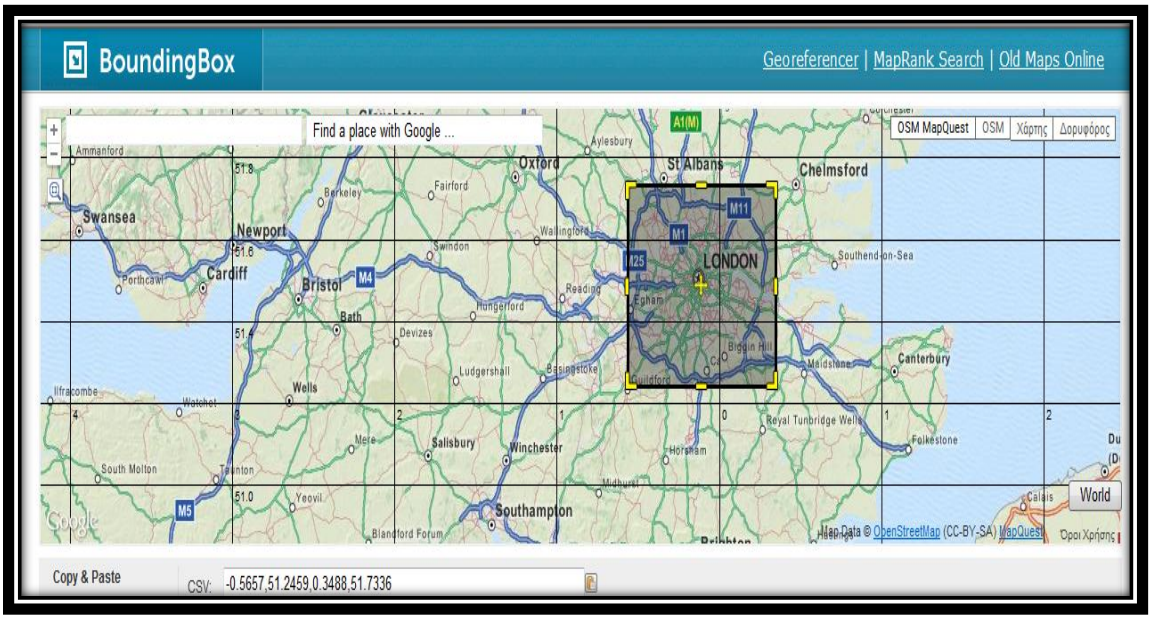
Συγκρίνοντας αυτές τις δυο γραφικές παρατηρούμε ότι το Lucene καταφέρνει και διατηρεί σταθερό χρόνο αναζήτησης πάρα την αύξηση του μεγέθους του Index. Το μεγαλύτερο μέρος των αναζητήσεων εκτελέστηκε σε περίπου 0,0003 δευτερόλεπτα. Επίσης αξιοσημείωτο είναι ότι εκτελείται αναζήτηση σε index μεγέθους 5.4MB σε μόλις 0,00027 δευτερόλεπτα.



Εικόνα 20 Χρόνος αναζήτησης/Μέγεθος Index

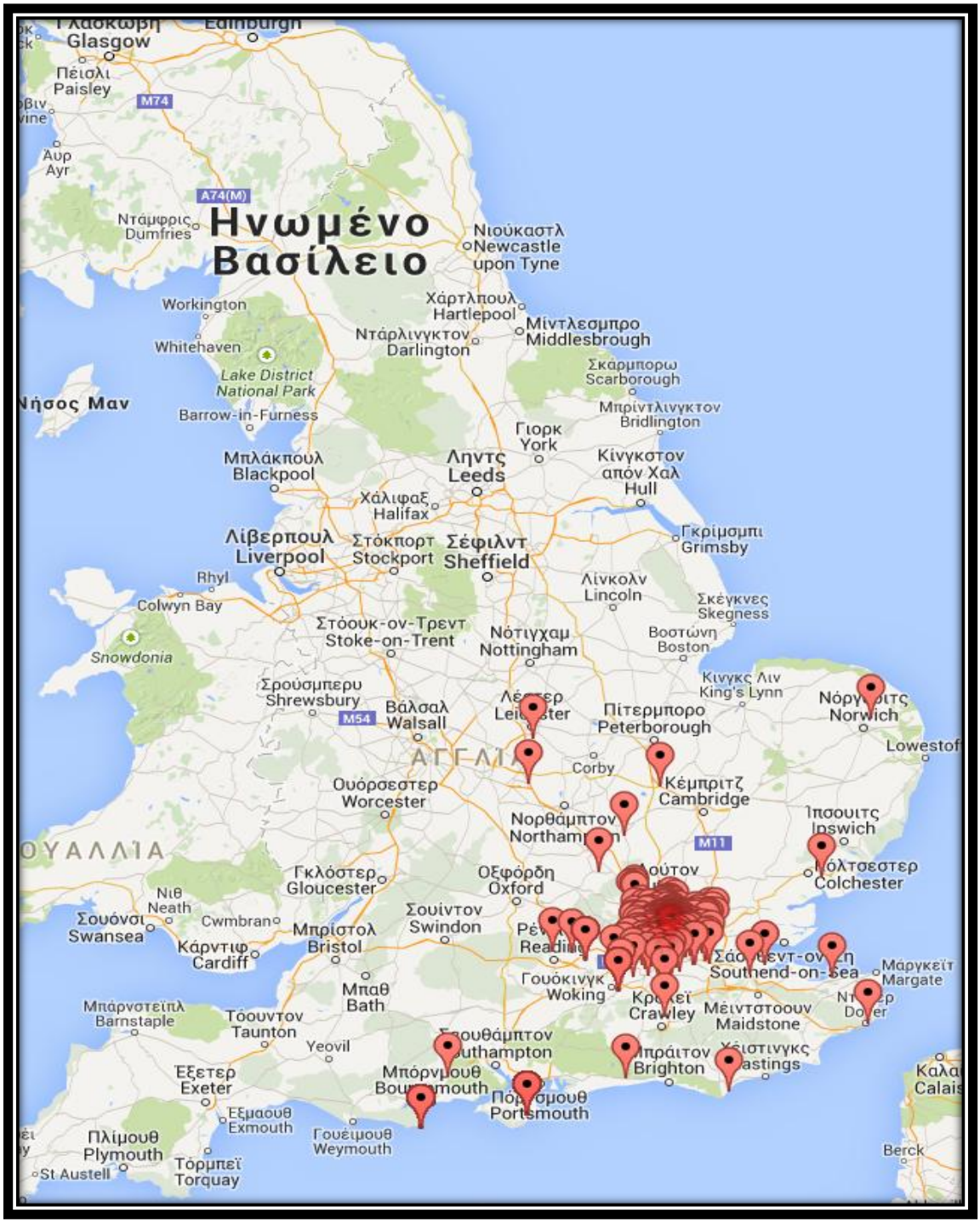
Αξιολόγηση ακριβείας τοποθεσίας του Streaming API

Θέλοντας να εξετάσουμε την ακρίβεια που προσφέρει το Streaming API του Twitter φιλτραρισμένο από μία γεωγραφική περιοχή, εκτελέσαμε το παρακάτω πείραμα. Χρησιμοποιώντας την εφαρμογή λάβαμε 500 tweets τα οποία είναι geotagged με γεωγραφικό χαρακτηριστικό την γεωγραφική περιοχή που περικλείει το Λονδίνο (-0.5657,51.2459,0.3488,51.7336).

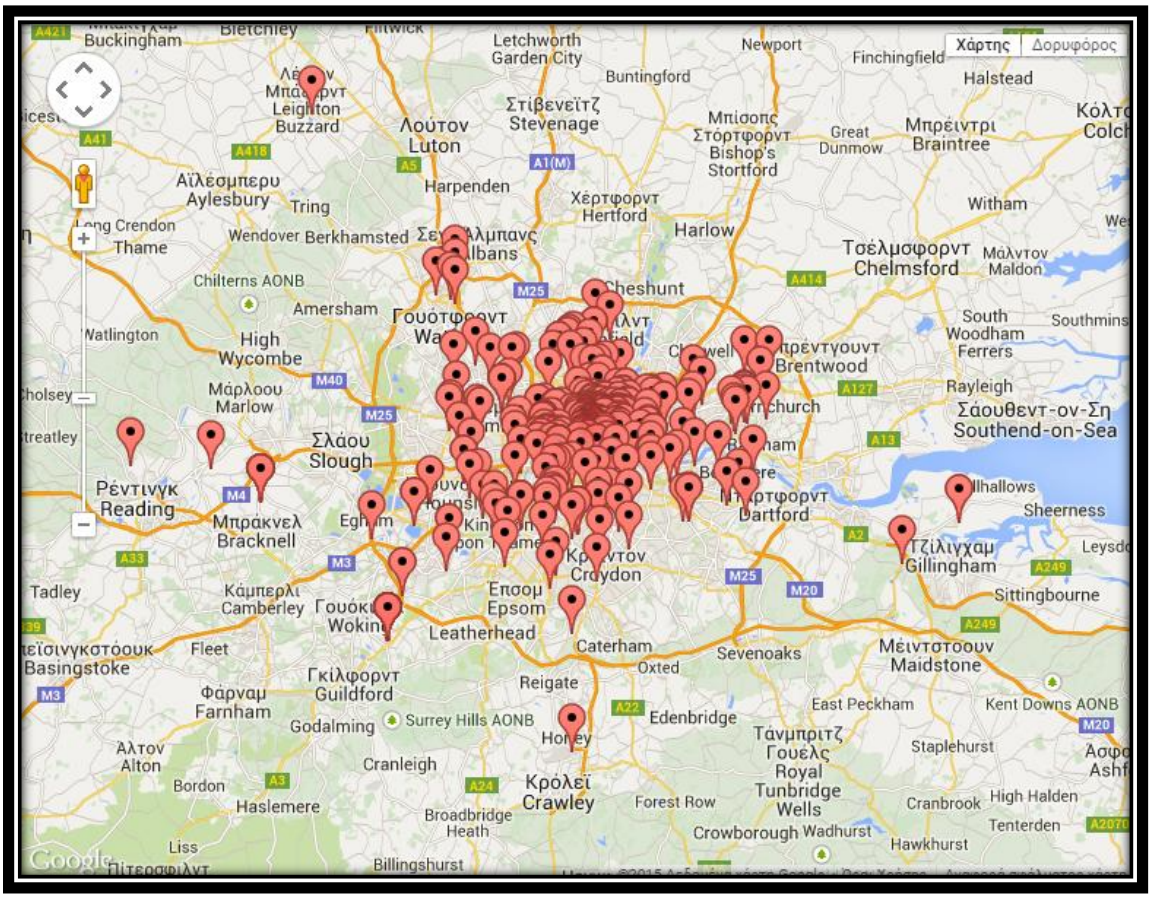


Εικόνα 21 Γεωγραφικός χώρος αναζήτησης

Έπειτα καταγράφοντας το γεωγραφικό πλάτος και μήκος αυτών των 500 tweets τα παρουσιάζουμε στις παρακάτω εικόνες :



Εικόνα 22 Γεωγραφική θέση tweets 1

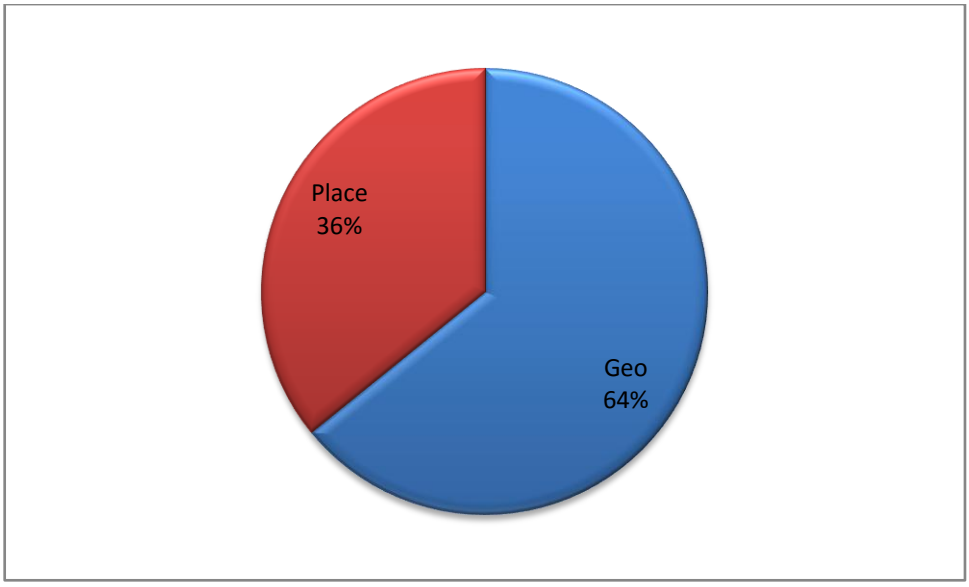


Εικόνα 23 Γεωγραφική θέση tweets 2

Αυτό το οποίο παρατηρούμε είναι το Streaming API του twitter δεν έχει 100% ακρίβεια στα επιστρεφόμενα tweets. Βρέθηκε ότι 23 από τα 500 tweets βρίσκονται εκτός του ορισμένου γεωγραφικού χώρου που έχουμε επιλέξει, επομένως το twitter API μας παρείχε στο συγκεκριμένο πείραμα ακρίβεια 95.4%.

Στη συνέχεια εξετάζουμε τον τρόπο με τον οποίο καταγράφτηκε η γεωγραφική περιοχή από την οποία στάλθηκαν τα tweets. Διαχωρίζουμε τους τρόπους με τους οποίους καταγράφτηκε η τοποθεσία του tweet.

- Geo : Το Tweet δημιουργήθηκε με ένα γεωγραφικό πλάτος και μήκος, η τοποθεσία καταγράφθηκε αυτόματα από μια κινητή έξυπνη συσκευή.
- Place: Το Tweet δημιουργήθηκε με ένα place_id, αλλά χωρίς γεωγραφικό πλάτος και μήκος . Δηλαδή ο χρήστης έχει θέσει χειροκίνητα την τοποθεσία από την οποία επιθυμεί να φαίνεται ότι έχει σταλεί το tweet.



Εικόνα 24 Διαμόρφωση geotagged tweets

Παρατηρούμε ότι το 64% των μηνυμάτων στάλθηκαν με προσδιορισμένο γεωγραφικό πλάτος και μήκος και μπορούμε να υποθέσουμε ότι προήλθαν από έξυπνες κινητές συσκευές.

Σύγκριση μηχανών ανάλυσης συναισθήματος

Χρησιμοποιώντας την εφαρμογή αναζητούμε tweets τα οποία περιέχουν την λέξη <<London>>. Καταγράφουμε 10 από αυτά και έπειτα εξετάζουμε και συγκρίνουμε τα αποτελέσματα που επιστρέφουν τα δυο API ανάλυσης συναισθήματος που χρησιμοποιούμε στην εφαρμογή.

No	Tweet	Συναίσθημα Sentiment140	Συναίσθημα uClassify
1	Lovely stroll through #jubileepark #Canarywharf #london. @ Jubilee Park https://t.co/RFH9t8GQtv	Ουδέτερο	Θετικό 91% Αρνητικό 9%
2	@bitbybitco @GA_London thanks, great meeting you. Keep in touch	Θετικό	Θετικό85 % Αρνητικό 15%
3	@Pauline_Latham The policy was one of the	Ουδέτερο	Θετικό 86% Αρνητικό 14%

	causes of the current housing crisis, particularly in London!		
4	I've always loved this sneaky view of Tower Bridge since the first time I came to London on the train as a child. http://t.co/l7gqXA8gHy	Ουδέτερο	Θετικό 96% Αρνητικό 4%
5	@OobaCreative @GA_London thanks for the tweet - good luck with the Twitter hustle :)	Θετικό	Θετικό 93% Αρνητικό 7%
6	Sunny day in London. :') love this weather. Even if I have to go to work.....	Αρνητικό	Θετικό 93% Αρνητικό 7%
7	go sibs ?? @ London Aquatics Centre https://t.co/E5trPr1a8u	Ουδέτερο	Θετικό 100% Αρνητικό 0%
8	1.3m new families will get the right to buy their home. Big boost for workers in London http://t.co/LOxwKJ30qa.	Ουδέτερο	Θετικό 90% Αρνητικό 10%
9	London's Olympic stadium is in one of the most unimpressive, underdeveloped areas of the city. What must the tourists have thought?	Ουδέτερο	Θετικό 33% Αρνητικό 67%
10	Breakfast of champions before filming... #London #GoodMorning #Breakfast #UK #Foodie #Filming #BusyBee #LongDay http://t.co/VNIUEQqK0l	Ουδέτερο	Θετικό 100% Αρνητικό 0%

Συμπερασματικά παρατηρούμε ότι το Uclassify API έχει επιστρέψει πιο σωστά αποτελέσματα από αυτό του Sentiment140 API το οποίο στις περισσότερες των περιπτώσεων δε κατάφερε να εξάγει το σωστό συναίσθημα και επέστρεφε την τιμή ουδέτερο.

Κεφάλαιο 8: Συμπεράσματα

Σε αυτό το κεφάλαιο θα δοθεί μια περίληψη στην οποία θα αξιολογηθεί ο στόχος της έρευνας. Ακολούθως, ορισμένοι περιορισμοί της έρευνας αυτής θα πρέπει να επισημανθούν και να συζητηθούν. Τέλος , θα προταθούν θέματα τα οποία δε καλύφθηκαν, ως έναυσμα για μελλοντικές εργασίες .

Περίληψη

Στο εισαγωγικό κεφάλαιο 1 αναγνωρίστηκε το πρόβλημα και τέθηκαν οι στόχοι της παρούσας έρευνας. Σε αυτό το κεφάλαιο θα εξεταστεί αν ολοκληρώθηκαν οι στόχοι που είχαν τεθεί και τι συμπεράσματα βγήκαν. Παρακάτω δίνεται μια περίληψη της έρευνας ανά τους ερευνητικούς στόχους που τέθηκαν.

Αφού μελετήθηκαν τα χαρακτηριστικά ενός tweet μηνύματος εξετάστηκε το Rest, αλλά και το Streaming API τα οποία προσφέρει το twitter για εξαγωγή πληροφοριών. Έπειτα παρουσιάστηκε το πώς χρησιμοποιείται το Open Authentication (OAuth) έτσι ώστε να πιστοποιηθεί η ταυτότητα μιας εφαρμογής. Χρησιμοποιώντας τις παραπάνω γνώσεις δημιουργήθηκε η εφαρμογή της παρούσας εργασίας η οποία καλεί το streaming API για να λάβει tweets. Η αρχιτεκτονική λήψης Tweets αναλύθηκε στο κεφάλαιο 5. Στο κεφάλαιο 3 παρουσιάστηκε η δομή του Lucene και ο τρόπος με τον οποίο λειτουργεί. Αναλύθηκαν διεξοδικά οι κυρίως διεργασίες του, όπως AnalyzeDocument, IndexDocument, Index, RunQuery και BuildQuery-RenderResults και στο κεφάλαιο 5 παρουσιάστηκε η υλοποίηση του Lucene στην εφαρμογή της μελέτης. Πειραματικά εξετάστηκε ο αριθμός και ο ρυθμός των tweets που λαμβάνει η εφαρμογή και η ακρίβεια της χρήσης του φίλτρου Location στο Streaming API. Οι μετρήσεις έδειξαν ότι το Location φίλτρο δεν παρέχει πλήρη ακρίβεια στο φιλτράρισμα του προεπιλεγμένου γεωγραφικού χώρου. Επίσης παρουσιάστηκαν τα αποτελέσματα, για το τι συμβαίνει στο Lucene Index κατά τη διάρκεια της δημιουργίας του, αλλά και τη διάρκεια της αναζήτησης δεδομένων από αυτό. Τέλος, εξετάστηκε και παρουσιάστηκε ο χρόνος που χρειάζεται η εφαρμογή, ώστε να επιστρέψει τα αποτελέσματα της αναζήτησης εν αναλογία με το μέγεθος του Lucene Index.

Αξιολόγηση ερευνητικών στόχων

- **Στατιστική απεικόνιση μέσω γραφικών παραστάσεων πληροφοριών στο σύνολο των tweets.**
  Στο κεφάλαιο 5 παρουσιάστηκε ο τρόπος δημιουργίας των γραφημάτων. Πιο αναλυτικά, το πρώτο γράφημα παρουσιάζει τα δέκα πιο διαδεδομένα χρησιμοποιούμενα Hashtags και το δεύτερο παρουσιάζει τις δέκα πιο διαδεδομένες γλώσσες που έχουν χρησιμοποιηθεί για τη σύνταξη των tweets. Τα δεδομένα των γραφημάτων ανανεώνονται κάθε μισό λεπτό.

- **Εφαρμογή sentiment analysis και γραφική απεικόνιση των αποτελεσμάτων στα αποτελέσματα της αναζήτησης.**
  Στην παρούσα μελέτη χρησιμοποιήθηκαν 2 διαφορετικές μηχανές ανάλυσης συναισθήματος, αυτή του Sentiment140 και αυτή του Uclassify. Ο τρόπος με τον οποίο λειτουργούν αυτά τα 2 API αναλύεται στο κεφάλαιο 5. Επιπλέον πραγματοποιηθήκαν μετρήσεις για να εξεταστεί ποιο από αυτά τα API είναι πιο αξιόπιστο.

- **Απαίτηση όλα τα παραπάνω να συμβαίνουν σε πραγματικό χρόνο.**
  Η εφαρμογή της μελέτης αυτής δημιουργήθηκε για να λειτούργει σε πραγματικό χρόνο. Έτσι, την στιγμή κατά την οποία καταγράφονται τα tweets, να μπορούμε να κάνουμε αναζητήσεις στο Index. Επιπλέον, τα δεδομένα των γραφικών παραστάσεων ανανεώνονται αυτόματα κάθε μισό λεπτό, έτσι ώστε να συμπεριλαμβάνονται καινούργια στοιχεία από τα νέα μηνύματα tweets.

Μελλοντική έρευνα

Είδαμε ότι η αρχιτεκτονική της παρούσας εργασίας μπορεί να ανταπεξέλθει αρκετά καλά στο να συγκεντρώνει, αλλά και να αναλύει πολύ γρήγορα μηνύματα tweets με το συνδυασμό της βιβλιοθήκης Lucene. Παρόλα αυτά πιστεύουμε ότι η

εργασία μπορεί να βελτιωθεί περαιτέρω. Παρακάτω παρουσιάζονται μερικές ιδέες που θα μπορούσαν να συμβάλλουν προς αυτή τη κατεύθυνση.

- **Μεγαλύτερη ακρίβεια στο γεωγραφική περιοχή των tweets.**
  Παρατηρήθηκε ότι το StremingAPI του twitter δεν έχει 100% ακρίβεια στα επιστρεφόμενα tweets, καθώς βρέθηκε ότι το twitter API μας παρείχε ακρίβεια 95.4%. Παρεμβάλλοντας μια διαδικασία η οποία θα ελέγχει εάν το ληφθέν μήνυμα βρίσκεται εντός του ορισμένου γεωγραφικού χώρου ή όχι θα μπορούσε να προσδώσει στο σύστημα ακρίβεια 100%.

- **Αναγνώριση Spam tweets**
  Τον Αύγουστο του 2009 , σχεδόν το 11 τοις εκατό όλων των μηνυμάτων Twitter ήταν spam [12], ενώ μια άλλη έρευνα [13] διαπίστωσε μια αύξηση 355 % του spam στα κοινωνικά δίκτυα. Επομένως μια μελλοντική εργασία θα μπορούσε να επικεντρωθεί στο πως θα μπορούσε να μειώσει το spam έτσι ώστε τα αποτελέσματα από την αναζήτησή μας να είναι ορθότερα.

- **Event detection**
  Η εφαρμογή που έχει δημιουργηθεί για την έρευνα της εργασίας προσφέρει το μεγαλύτερο μέρος του σχεδιασμού, έτσι ώστε να δημιουργηθεί μια εφαρμογή που θα μπορεί να εντοπίζει αυτόματα και σε πραγματικό χρόνο γεγονότα που συμβαίνουν σε μια γεωγραφική περιοχή, δηλαδή την αναγνώριση tweets τα οποία αναφέρονται στο ίδιο θέμα, σε κοντινή γεωγραφική περιοχή και με υψηλότερη συχνότητα από το σύνηθες.

Βιβλιογραφία

[1] Mustafa Sofean and Matthew Smith(2012): **A Real-Time Disease Surveillance Architecture Using Social Networks**, Proceedings of the 24th European Medical Informatics Conference (MIE 2012)

[2] **Twitter API wiki.** [Ηλεκτρονικό]. Available :http://apiwiki.twitter.com/TwitterAPIDocumentation. [Πρόσβαση 2015].

[3] **Apache Lucene.** [Ηλεκτρονικό]. Available :http://lucene.apache.org/[Πρόσβαση 2015].

[4] Watanabe, K.; Ochi, M.; Okabe, M. & Onai, R. (2011), **Jasmine: a real-time local-event detection system based on geolocation information propagated to microblogs.**, in Craig Macdonald; Iadh Ounis & Ian Ruthven, ed., 'CIKM' , ACM, , pp. 2541-2544 .

[5] Berlingerio, M.; Calabrese, F.; Lorenzo, G. D.; Dong, X.; Gkoufas, Y. & Mavroeidis, D. (2013), **SaferCity: A System for Detecting and Analyzing Incidents from Social Media**., in Wei Ding 0003; Takashi Washio; Hui Xiong; George Karypis; Bhavani M. Thuraisingham; Diane J. Cook & Xindong Wu, ed., 'ICDM Workshops' , IEEE Computer Society, , pp. 1077-1080 .

[6] **Sentiment140.** [Ηλεκτρονικό]. Available :http://www.sentiment140.com/ [Πρόσβαση 2015].

[7] Go, A.; Bhayani, R. & Huang, L. (2009), **'Twitter Sentiment Classification using Distant Supervision'**, Processing , 1--6 .

[8] Sakaki, T.; Okazaki, M. & Matsuo, Y. (2010), **Earthquake shakes Twitter users: real-time event detection by social sensors**, in 'Proceedings of the 19th international conference on World wide web' , pp. 851--860 .

[9] Hatcher, E.; Gospodnetic, O. & McCandless, M. (2010), **Lucene in Action** , Manning .

[10] **Uclassify.** [Ηλεκτρονικό]. Available :http://uclassify.com/browse/uClassify/Sentiment[Πρόσβαση 2015].

[11] **Semiocast : Geolocation analysis of Twitter accounts and tweets by Semiocast.**[Ηλεκτρονικό]. Available

:http://semiocast.com/en/publications/2012_07_30_Twitter_reaches_half_a_billion_accounts_140m_in_the_US [Πρόσβαση 2015].

[12] **A new look at spam by the numbers**. [Ηλεκτρονικό]. Availabe :http://scitech.blogs.cnn.com/2010/03/26/a-new-look-at-spam-by-the-numbers/[Πρόσβαση 2015].

[13] Hashemi, R. R.; Bahar, M.; Nguyen, H. & Tift, K. (2006), **Spam Detection: A Syntax and Semantic-based Approach.**, in Hamid R. Arabnia & Ray R. Hashemi, ed., 'IKE' , CSREA Press, , pp. 354-360 .

[14] **Twitter Company.**[Ηλεκτρονικό]. Available : https://about.twitter.com/company[Πρόσβαση 2015].

[15] **OAuth.** [Ηλεκτρονικό]. Available :http://oauth.net/ [Πρόσβαση 2015].

[16] **Google, How Search Works**. [Ηλεκτρονικό]. Available :https://www.google.com/search/about/insidesearch/howsearchworks/[Πρόσβαση 2015].

[17] **The Louvain method for community detection in large networks.**[Ηλεκτρονικό]. Available :https://perso.uclouvain.be/vincent.blondel/research/louvain.html [Πρόσβαση 2015].

[18] **IPSV**. [Ηλεκτρονικό]. Available :http://www.esd.org.uk/standards/ipsv/2.00/ [Πρόσβαση 2015].

Παράρτημα Α

Στο παράρτημα Α παρουσιάζονται τα αποτελέσματα των μετρήσεων που χρησιμοποιήθηκαν για τις μετρήσεις του κεφαλαίου 6.

Συμπεριφορά συστήματος στη λήψη tweets

Ανάλυση δομή ουράς

Χρόνος (min)	Μέγεθος ουράς
00:00:30.002	0
00:01:00.015	0
00:01:30.030	0
00:02:00.044	0
00:02:30.058	0
00:03:00.072	0
00:03:30.086	0
00:04:00.100	0
00:04:30.114	0
00:05:00.128	0
00:05:30.142	0
00:06:00.156	0
00:06:30.171	0
00:07:00.185	0
00:07:30.199	0
00:08:00.213	0
00:08:30.227	0
00:09:00.241	0
00:09:30.255	1
00:10:00.269	0

Αριθμός Documents στο IndexBuffer

Χρόνος (min)	Documents στη RAM
0:00	0
00:24.347	30

00:54.362	49
01:24.376	48
01:54.39	42
02:24.404	44
02:54.419	43
03:24.432	39
03:54.446	43
04:24.460	47
04:54.473	44
05:24.488	39
05:54.523	37
06:24.517	35
06:54.532	41
07:24.546	41
07:54.560	46
08:24.572	43
08:54.589	42
09:24.601	43
09:54.614	39
10:24.630	40

Χρόνος αναζήτησης/Μέγεθος Index

Χρόνος αναζήτησης (sec)	Μέγεθος directory (bytes)
0,0112643	7513
0,0015621	137746
0,000162	302963
0,0002106	475612
0,0002321	633864
0,0002502	781811
0,0002734	937633
0,0002574	1092926
0,0003151	1302625
0,0002719	1461990
0,0003507	1587631
0,0002408	1765111
0,0002218	1984286
0,0002221	2156203
0,0002288	2370910
0,0003613	2514150
0,000492	2742143
0,0002807	2888498

0,0022482	5979891
0,0003247	3210263
0,0001901	3398071
0,0002837	3554421
0,0003278	3712852
0,0003625	3879406
0,0004099	4079647
0,0004494	4366750
0,0003287	4360591
0,0004687	4509530
0,011302	6387854
0,0005074	4850784
0,0003051	5073004
0,0003401	5270835
0,0003042	5496901
0,0002725	5746805
0,0003513	5959757