ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εξαγωγή και ανάλυση θεμάτων και συναισθημάτων σε μηνύματα του Twitter με βάση τη χωρική και τη χρονική διάσταση

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΓΕΩΡΓΙΟΥ Δ. ΑΝΑΓΝΩΣΤΟΠΟΥΛΟΥ

Επιβλέπων : Ιωάννης Βασιλείου
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2015

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΈΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εξαγωγή και ανάλυση θεμάτων και συναισθημάτων σε μηνύματα του Twitter με βάση τη χωρική και τη χρονική διάσταση

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΓΕΩΡΓΙΟΥ Δ. ΑΝΑΓΝΩΣΤΟΠΟΥΛΟΥ

Επιβλέπων : Ιωάννης Βασιλείου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την

.....	.....	.....
Ιωάννης Βασιλείου	Κωνσταντίνος Κοντογιάννης	Ιωάννης Σταύρακας
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2015

.....

ΓΕΩΡΓΙΟΣ Δ. ΑΝΑΓΝΩΣΤΟΠΟΥΛΟΣ

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

Περίληψη

Ο σκοπός της διπλωματικής εργασίας ήταν σε πρώτη φάση η επεξεργασία κειμένων (tweets) από το Twitter για την αναγνώριση του θέματος και του συναισθήματός τους και η συσχέτισή τους με τη γεωγραφική περιοχή από την οποία δημοσιεύτηκαν και το χρόνο (ημερομηνία και ώρα) που έγινε η δημοσίευση. Έπειτα ομαδοποιήθηκαν τα tweets ανά θέμα και ανά περιοχή, και μελετήθηκε η διαφοροποίηση των θεμάτων και των συναισθημάτων των μηνυμάτων στο twitter καθώς αλλάζει η περιοχή ή/και το χρονικό παράθυρο που εξετάζεται. Φτιάχτηκε τέλος εφαρμογή η οποία παρουσιάζει στο χρήστη τα αποτελέσματα της εξαγωγής θεμάτων και συναισθημάτων από μεγάλο αριθμό tweets και το πώς αυτά διαφοροποιούνται ανά πόλη και ανά ημέρα.

Συγκεκριμένα, αφού ελήφθησαν tweets από το twitter με βάση τη γεωγραφική περιοχή (Λονδίνο, Νέα Υόρκη, Λος Άντζελες) και τις ημέρες τις οποίες δημοσιεύτηκαν (08 – 14 Ιουνίου 2015), έγινε η επεξεργασία του περιεχομένου τους, ομαδοποιήθηκαν ανά θέμα και εξήχθη το συναίσθημα του κάθε tweet. Για την κάθε θεματική ομάδα εξήχθησαν οι πιο σημαντικοί-αντιπροσωπευτικοί όροι και βρέθηκε το πώς μοιάζει με τις υπόλοιπες στην ίδια και σε διαφορετικές πόλεις. Τελικά, κάθε θεματική ομάδα συνδεδεμένη με τα tweets τα οποία περιέχει και με τις υπόλοιπες ομάδες με τις οποίες μοιάζει, εισήχθη σε βάση δεδομένων. Το συναίσθημα για ένα θέμα υπολογίζεται αθροιστικά από τα συναισθήματα του κάθε tweet που σχετίζεται με αυτό. Αναπτύχθηκε εφαρμογή web η οποία επικοινωνεί με τη βάση δεδομένων και παρουσιάζει τα αποτελέσματα της επεξεργασίας στο χρήστη, ο οποίος έχει δυνατότητα να προσδιορίσει το τι θέλει να δει κάθε φορά.

Στα αποτελέσματα που παρουσιάζονται από τη web εφαρμογή είναι φανερό το πώς αλλάζουν τα θέματα για τα οποία μιλούν οι χρήστες ανά πόλη και ανά μέρα, και ποια η διαφορά στους όρους του ίδιου θέματος μεταξύ διαφορετικών πόλεων. Ακόμα παρατηρείται το συναίσθημα που μπορεί να διακατέχει τους χρήστες όταν γράφουν (<<τουιτάρουν>>) για κάποιο θέμα που τους επηρεάζει.

Λέξεις Κλειδιά: κείμενο, tweet, Twitter, αναγνώριση, θέμα, συναίσθημα, ομαδοποίηση, περιοχή, επεξεργασία, ομοιότητες,θεματική ομάδα, σημαντικοί όροι, διαφορές, βάση δεδομένων, εφαρμογή ιστού

Abstract

The purpose of this thesis was the processing of texts (tweets) from Twitter so as to determine their topics and their sentiment and to connect them with the geographical area from which they were published and the time (date and time) the publication was done. Then the tweets were clustered according to their topics and their areas, and the diversification of their topics and sentiments was studied as the area and the timeframe changed. In the end, an application was created that presents the results of the processing of a great number of tweets to the user, and how these results differ by city and by day.

More precisely, after collecting tweets from Twitter based on their geographical area (London, New York, Los Angeles) and the days on which they were published (June 8th to June 14th 2015), their content was processed, they were clustered by subject and the sentiment of each individual tweet was determined. For every subject group, the most important-representative terms were extracted and then the similarity of each group to the others was calculated for different days and different cities. Finally, each cluster along with the tweets assigned to it and the other clusters to whom it is similar, were inserted in a database. The sentiment of a topic is calculated cumulatively from the sentiment of each related tweet. A web application was developed that communicates with the database and presents the results of the processing to the user, who may choose what specific information he wants to see.

From the results presented by the web application, the user can see how the topics change between different cities and days, and what the differences in the same topic are each time. Furthermore, he can get information about the sentiment of the users tweeting about a subject that concerns them.

Keywords: text, tweet, Twitter, extraction, topic, subject, sentiment, clustering, geographical area, processing, similarities, differences, subject group, top terms, important terms, database, web application

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή κ. Ιωάννη Βασιλείου που είχε την επίβλεψη της συγκεκριμένης διπλωματικής εργασίας για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα τόσο ενδιαφέρον και σύγχρονο αντικείμενο.

Θέλω να πω ένα μεγάλο ευχαριστώ στον κ. Δημήτριο Σκούτα για την αμέριστη βοήθεια του και το ενδιαφέρον του καθ' όλη την διάρκεια της παρούσας μελέτης. Από την πρώτη στιγμή ήταν δίπλα μου και πάντα πρόθυμος να με καθοδηγήσει και να με βοηθήσει κάθε φορά που βρισκόμουν σε κάποιο αδιέξοδο ή αντιμετώπιζα κάποια δυσκολία, θυσιάζοντας πολύτιμο προσωπικό χρόνο.

Πίνακας περιεχομένων

1 Εισαγωγή

1.1 Άντληση πληροφοριών από ιστοσελίδες κοινωνικής δικτύωσης και εξαγωγή συμπερασμάτων

Ένα κοινωνικό δίκτυο ορίζεται σαν ένα σύνολο αλληλεπιδράσεων και φυσικών σχέσεων. Οι κόμβοι του δικτύου είναι οι χρήστες και οι ακμές περιγράφουν τις μεταξύ τους σχέσεις και αλληλεπιδράσεις. Ένας χρήστης μπορεί να είναι ένα άτομο, μια ομάδα ατόμων, μια εταιρεία, μια οργάνωση, η οποία όμως στο δίκτυο συμπεριφέρεται σαν μία οντότητα και εκπροσωπεί τα μέλη της. Οι σχέσεις μεταξύ χρηστών περιλαμβάνουν συγγενικές σχέσεις, εργασιακές, φιλικές και άλλες, ενώ οι αλληλεπιδράσεις περιγράφουν οποιαδήποτε επικοινωνία πραγματοποιείται μεταξύ τους. Τον 21ο αιώνα έχουν κάνει την εμφάνισή τους ιστοσελίδες οι οποίες προσομοιάζουν στα πραγματικά κοινωνικά δίκτυα και προσφέρουν στους χρήστες τους τρόπους επικοινωνίας και πληθώρα άλλων υπηρεσιών.

Οι ιστοσελίδες κοινωνικής δικτύωσης έχουν γίνει πολύ δημοφιλείς τα τελευταία χρόνια χάρις στην αύξηση του αριθμού των συσκευών που μπορούν να μπαίνουν στο διαδίκτυο – όπως οι προσωπικοί υπολογιστές, τα κινητά, τα tablets- και στο μειωμένο κόστος απόκτησής τους. Συγχρόνως στις περισσότερες χώρες του κόσμου οι εταιρείες κινητής τηλεφωνίας προσφέρουν πια φθηνά προγράμματα δεδομένων για τους απλούς χρήστες, δίνοντας σε όλους τη δυνατότητα να μπορούν να μένουν συνδεδεμένοι ανά πάσα στιγμή, όπου κι αν βρίσκονται. Λόγω του μειωμένου κόστους και της αμεσότητας στην επικοινωνία, όλο και περισσότεροι χρήστες συνδέονται στο Facebook, στο Twitter και σε άλλες παρόμοιες σελίδες. Τέτοια κοινωνικά δίκτυα έχουν οδηγήσει σε μια έκρηξη από δικτυοκεντρικά δεδομένα σε μια ποικιλία από σενάρια. Τα δίκτυα αυτά μπορεί να προσδιορίζονται είτε στο πλαίσιο συστημάτων όπως το Facebook και το Twitter, τα οποία είναι σχεδιασμένα για κοινωνική αλληλεπίδραση, είτε συστημάτων όπως το Instagram και το Flickr, που προσφέρουν κυρίως δυνατότητες διαμοιρασμού περιεχομένου. Οι δυνατότητες που προσφέρει κάθε σύστημα γίνονται όλο και περισσότερες, συχνά επικαλύπτονται και χρησιμοποιούνται η μία μέσα στην άλλη καθώς για παράδειγμα μια δημοσίευση στο Twitter μπορεί πια να γίνει μέσω Instagram.

Ο όγκος και η ποικιλία της πληροφορίας που δημιουργείται και διακινείται στα μέσα κοινωνικής δικτύωσης έχει δημιουργήσει μία έντονη ανάγκη για την ανάπτυξη τεχικών που επιτρέπουν να αναλύονται με αυτόματο τρόπο τα θέματα που συζητούνται από τους χρήστες και οι γνώμες που διατυπώνονται. Επιπλέον, ένα μεγάλο ποσοστό αυτής της πληροφορίας συνοδεύεται, άμεσα ή έμμεσα, από χωρική πληροφορία. Η χωρική αυτή συνιστώσα είναι συχνά ιδιαίτερα ενδιαφέρουσα και χρήσιμη για την περαιτέρω επεξεργασία, ανάλυση και αξιοποίηση της παραπάνω πληροφορίας. Για παράδειγμα, μία ανάλυση τέτοιου είδους μπορεί να αποκαλύψει χρήσιμα συμπεράσματα για τη συσχέτιση ανάμεσα στην τοποθεσία των χρηστών και στα θέματα που συζητούνται ή τις γνώμες που εκφράζονται. Παρότι πολλές εργασίες έχουν εστιάσει στην επίλυση των επιμέρους προβλημάτων, λείπουν ακόμα επαρκείς και ολοκληρωμένες λύσεις που να λαμβάνουν υπόψη και να συνδυάζουν όλες αυτές τις παραμέτρους.

1.2 Αντικείμενο και συνεισφορά διπλωματικής

Η εργασία αυτή έχει σαν αντικείμενο την άντληση κειμένων χρηστών από μέσα κοινωνικής δικτύωσης -και συγκεκριμένα από το Twitter- και την επεξεργασία τους με στόχο τον εντοπισμό δημοφιλών θεμάτων συζήτησης και του συναισθήματος των χρηστών το οποίο αποτυπώνεται στα κείμενά τους. Συγχρόνως επιδιώκεται η καταγραφή του χρόνου, δηλαδή της ημέρας που γράφτηκε ένα κείμενο, και του τόπου ,δηλαδή της πόλης ή ακόμα και της περιοχής από την οποία γράφτηκε, και η συσχέτισή τους με τα θέματα και συναισθήματα που εξήχθησαν από την επεξεργασία των κειμένων. Τέλος, είναι επιθυμητό να υπάρξει τρόπος παρουσίασης των 4 αυτών βασικών πληροφοριών **θέμα, συναίσθημα, τόπο και χρόνο** με τρόπο ώστε ο χρήστης να μπορέσει να εξαγάγει χρήσιμα συμπεράσματα για το τι απασχόλησε μια ομάδα ανθρώπων/χρηστών σε μια περιοχή και για μια συγκεκριμένη χρονική περίοδο.

Για την υλοποίηση των παραπάνω ζητουμένων, στα πλαίσια της παρούσας διπλωματικής εργασίας έγιναν τα ακόλουθα:

- Συλλογή πληροφοριών από μέσα κοινωνικής δικτύωσης, συγκεκριμένα από αναρτήσεις χρηστών στο Twitter, οι οποίες συνοδεύονται από χωρική πληροφορία.
- Επεξεργασία κειμένου με σκοπό τη θεματική ανάλυση. Για τη θεματική ανάλυση χρησιμοποιήθηκε υλοποίηση αλγορίθμου ομαδοποίησης (clustering)
- Επεξεργασία κειμένου με σκοπό την ανάλυση γνώμης και συναισθημάτων. Για την ανάλυση συναισθημάτων χρησιμοποιήθηκε υλοποίηση αλγορίθμου ταξινόμησης (classification) ώστε να ταξινομηθούν τα tweets σε θετικά,αρνητικά ή ουδέτερα
- Εξαγωγή συσχετίσεων μεταξύ των παραπάνω συνιστωσών.
- Εισαγωγή των ηντλημένων πληροφοριών σε βάση δεδομένων.
- Ανάπτυξη web εφαρμογής για την οπτική απεικόνιση των αποτελεσμάτων σε χάρτες με τρόπο που να διευκολύνει την κατανόησή τους.

1.2.1

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Ανάπτυξη εφαρμογής Java με χρήση έτοιμων βιβλιοθηκών για τη συλλογή αναρτήσεων (tweets) χρηστών στο Twitter. Τα αποτελέσματα περιορίζονται σε συγκεκριμένες πόλεις, συγκεκριμένη χρονική περίοδο και είναι μόνο στην αγγλική γλώσσα.

2. Χρήση υλοποίησης του αλγορίθμου K-means στην πλατφόρμα Hadoop με σκοπό την ομαδοποίηση των όρων-λέξεων και την εξαγωγή θεμάτων.

3. Χρήση υλοποίησης του αλγορίθμου Naive Bayes στην πλατφόρμα Hadoop με σκοπό την ταξινόμηση των tweets σε θετικά,αρνητικά ή ουδέτερα.

4. Ανάπτυξη εφαρμογής Java για την επεξεργασία των θεμάτων και της πορείας τους στο χρόνο (ανά μέρα).

5. Χρήση της βάσης για γράφους neo4j και της web console που τη συνοδεύει για την εισαγωγή των δεδομένων σε αυτή. Αναπτύχθηκε επίσης εφαρμογή Java για την εισαγωγή στη βάση των συσχετίσεων του βήματος 4.

6. Ανάπτυξη web εφαρμογής με χρήση HTML, CSS και Javascript που κάνει REST calls στη βάση και παρουσιάζει δεδομένα στο χρήστη με τρόπο ώστε αυτός να μπορεί να εξαγάγει χρήσιμα συμπεράσματα. Απεικονίζει τα tweets σε χάρτη και το περιεχόμενό τους, το θέμα τους και το συναίσθημα τους σε ειδικό πλαίσιο.

1.3 Οργάνωση κειμένου

Εργασίες σχετικές με το αντικείμενο της διπλωματικής παρουσιάζονται στο **Κεφάλαιο 2** . Το **Κεφάλαιο 3** αναλύει τους διάφορους αλγορίθμους που χρησιμοποιούνται και επίσης παρουσιάζει συνοπτικά τις τεχνικές και τις πλατφόρμες με τις οποίες έγινε η επεξεργασία των δεδομένων, η αποθήκευσή τους σε βάση δεδομένων και η παρουσίασή τους στο χρήστη. Στο **Κεφάλαιο 4** αναλύονται οι απαιτήσεις του συστήματος και παρουσιάζεται με λεπτομέρεια η λειτουργία που καλείται να επιτελέσει το κάθε υποσύστημα. Στο **5ο Κεφάλαιο** εξηγείται ο τρόπος με τον οποίο κάθε μέρος του συστήματός μας επιτελεί τη λειτουργία του. Δίνεται ο κώδικας ή οι εντολές που χρησιμοποιούνται, οι είσοδοι που δίνονται σε κάθε υποσύστημα και τα αποτελέσματα της επεξεργασίας των δεδομένων. Στο Κ**εφάλαιο 6** εξετάζεται η ορθότητα των αποτελεσμάτων με τη χρήση συγκεκριμένων σεναρίων. Συγκρίνεται η σχέση των αποτελεσμάτων με πραγματικά γεγονότα και αξιολογείται η ποιότητά τους με βάση το τι συνέβη στην πραγματικότητα και το πώς ο ανθρώπινος νους θα περίμενε να έχουν αποδωθεί συγκεκριμένα γεγονότα και θέματα συζήτησης. Στο **7ο Κεφάλαιο** γίνεται μια σύνοψη της διπλωματικής εργασίας, παρατίθενται συμπεράσματα και αναφέρονται περιοχές στις οποίες θα μπορούσαμε να επεκταθούμε στο μέλλον.

2 Σχετικές εργασίες

Στο κεφάλαιο αυτό περιγράφουμε ορισμένες από τις κύριες σχετικές εργασίες, επικεντρώνοντας στις εξής θεματικές περιοχές:

1. Εξαγωγή θεμάτων από κοινωνικά δίκτυα: Μελέτη του περιεχομένου δημοσιεύσεων των χρηστών και εξαγωγή πληροφορίας για το θέμα τους. Συχνά η πληροφορία για το θέμα συσχετίζεται με πληροφορία για τη γεωγραφική περιοχή και το χρόνο δημοσίευσης.
2. Επεξεργασία κατηγορικών δεδομένων του Twitter: Δημοσιεύσεις στο Twitter όπως και σε άλλα κοινωνικά δίκτυα περιέχουν κατηγορική πληροφορία (περιοχή, γλώσσα, είδος συσκευής κ.α), η οποία μπορεί να υποστεί ειδική επεξεργασία.
3. Συναισθηματική ανάλυση δεδομένων του Twitter

2.1 Εξαγωγή θεμάτων από το Twitter

Οι Skovsgaard et al. (2014) παρουσίασαν ένα πλαίσιο για την επεξεργασία χωροχρονικά περιορισμένων αναζητήσεων δημοφιλών (top-k) όρων σε κείμενο συνεχούς ροής, χαρακτηρισμένου ως προς το χώρο και το χρόνο δημοσίευσης. Καθώς νέα δεδομένα λαμβάνονται κάθε δευτερόλεπτο, το πλαίσιο επεκτείνει ήδη υπάρχουσες τεχνικές για να κρατάει τους δημοφιλείς όρους σε συγκεντρωτικά σύνολα και να επιτρέπει την δυναμική προσαρμογή του συνόλου αυτού σε αλλαγές που προέρχονται από τα νεοεισερχόμενα δεδομένα. Η εργασία των Anders Skovsgaard et al. ασχολείται με τη διατήρηση ενός συνόλου δημοφιλών όρων και την ανανέωσή του σύμφωνα με τους νέους που λαμβάνει συνεχώς. Τα δεδομένα αυτά έχουν χωροχρονικά χαρακτηριστικά. Όμως οι δημοφιλέστεροι όροι λαμβάνονται έτοιμοι από συγκεκριμένα queries. Στο πλαίσιο της μελέτης τους δεν γίνεται ανάλυση κειμένου για να εξαχθούν οι όροι από ανεπεξέργαστο κείμενο. Στην παρούσα διπλωματική εξάγονται οι top-k όροι κατευθείαν από το κείμενο και αποθηκεύονται στη βάση neo4j για κάθε θεματική ενότητα (cluster) που λαμβάνεται από την επεξεργασία (clustering). Το σύνολο των σημαντικότερων όρων δεν αλλάζει γιατί δεν λαμβάνονται νέα δεδομένα.

Οι Budak et al. (2014) εισήγαγαν το GeoScope: μια υλοποίηση που μπορεί να βρίσκει όλες τις δημοφιλείς συσχετίσεις ανάμεσα σε γεγονότα και περιοχές όπως για παράδειγμα ένας σεισμός ή μια αθλητική εκδήλωση. Η υλοποίησή τους λειτουργεί καλά και για μεγάλο όγκο δεδομένων. Το GeoScope επικεντρώνεται σε γεγονότα που έχουν γεωγραφικό ενδιαφέρον όπως οι σεισμοί και λαμβάνονται από τα hashtags. Δεν εξετάζει διαφόρων ειδών γεγονότα, ούτε γενικού ενδιαφέροντος δημοφιλή θέματα όπως η μουσική -εξετάζει όμως ένα μουσικό event το οποίο είναι χαρακτηριστικό μιας περιοχής (πχ Woodstock). Και η παρούσα διπλωματική εργασία ενδιαφέρεται για τα hashtags και τα γεγονότα που σχετίζονται άμεσα με το χώρο στον οποίο συμβαίνουν. Τα εξάγει μέσω του clustering. Όμως εξάγει και γεγονότα-θέματα όλων των υπόλοιπων ειδών. Κάθε θέμα για το οποίο θα μπορούσαν να συζητούν οι άνθρωποι.

Οι Zhijun Yin et al. (2011) μελετούν το πρόβλημα της εξεύρεσης και σύγκρισης γεωγραφικών θεμάτων. Λαμβάνουν ένα σύνολο από δεδομένα του Flickr τα οποία περιέχουν χωρική πληροφορία από GPS. Επεξεργάζονται αυτά τα δεδομένα με σκοπό την εξαγωγή των γεωγραφικών θεμάτων. Στη συνέχεια συγκρίνουν τα θέματα διαφορετικών γεωγραφικών περιοχών. Πιο συγκεκριμένα, τα δεδομένα που συλλέγουν από το Flickr είναι εικόνες-φωτογραφίες οι οποίες περιέχουν πληροφορίες σχετικά με την τοποθεσία τους (συντεταγμένες GPS) και ορισμένες λέξεις-ετικέτες που ορίζει ο χρήστης και είναι σχετικές με τη φωτογραφία αυτή. Η επεξεργασία των δεδομένων συσχετίζει τη φωτογραφία με το γεωγραφικό χώρο από τον οποίο προέρχεται, και συνδυάζει τις ετικέτες για να βρει το θέμα της. Η παρούσα διπλωματική εργασία κάνει το ίδιο, χρησιμοποιώντας λέξεις από tweets του Twitter και τη χωρική πληροφορία που περιέχει σαν ετικέτα το κάθε tweet (πόλη/συντεταγμένες GPS). Η εργασία των Zhijun Yin et al. δεν εξετάζει συναισθήματα. Στα συμπεράσματα και των δύο εργασιών περιέχεται η διαπίστωση ότι τα θέματα μπορεί αλλάζουν ανά περιοχή, ενώ ταυτόχρονα μπορεί διαφορετικές περιοχές να ενδιαφέρονται για το ίδιο θέμα.

Οι Liangjie Hong et al. (2012) μελετούν tweets από τη ροή του Twitter και εξάγουν πληροφορίες σχετικά με τα θέματά τους και τη σχέση τους με τη γεωγραφική περιοχή από την οποία έχουν δημοσιευτεί. Εξετάζεται η διαφοροποίηση των θεμάτων και των εκφράσεων των tweets καθώς αλλάζουν οι περιοχές. Τα αποτελέσματα της επεξεργασίας χρησιμοποιούνται για να φτιαχτεί ένα μοντέλο με τη συσχέτιση κειμένου – περιοχής. Με το μοντέλο αυτό γίνεται έπειτα εφικτή η πρόβλεψη για την περιοχή προέλευσης ενός tweet με βάση το κείμενό του. Στην παρούσα διπλωματική εργασία εξετάζεται επίσης η σχέση μεταξύ θεμάτων και γεωγραφικών περιοχών, αλλά η μελέτη δεν επεκτείνεται σε τρόπους πρόβλεψης για το από πού δημοσιεύεται ένα tweet έχοντας μόνο το κείμενό του.

Οι Abdelhaq et al. (2013) Παρουσιάζουν ένα πλαίσιο για τον εντοπισμό τοπικών γεγονότων σε πραγματικό χρόνο από τη ροή του Twitter και την παρακολούθηση της εξέλιξής τους στο χρόνο. Γι αυτό εξάγουν συνεχώς χωρο-χρονικά χαρακτηριστικά λέξεων-κλειδιών ώστε να προσδιοριστούν οι υποψήφιες λέξεις για την περιγραφή γεγονότων. Έπειτα εξάγεται η τοπική πληροφορία για τα γεγονότα κάνοντας clustering των λέξεων-κλειδιών με βάση τη χωρική τους ομοιότητα. Για να προσδιοριστούν τα σημαντικότερα γεγονότα για ένα χρονικό πλαίσιο, εισάγουν ένα σχέδιο βαθμολόγησης για τα γεγονότα. Οι Hamed Abdelhaq et al. επικεντρώνονται σε γεγονότα και τα ομαδοποιούν με βάση το πόσο κοντά ήταν οι περιοχές από όπου δημοσιεύθηκαν τα tweets. Αντιθέτως, στην παρούσα διπλωματική εργασία η ομαδοποίηση έγινε με βάση την ομοιότητα του περιεχομένου των tweets και όχι την χωρική ομοιότητα.

Οι Μαθιουδάκης και Κούδας (2010) παρουσιάζουν το TwitterMonitor, ένα σύστημα που πραγματοποιεί ανίχνευση των τάσεων και τις αναλύει. Αρχικά εντοπίζει τις λέξεις που ξαφνικά φτάνουν σε ασυνήθιστα μεγάλο αριθμό εμφανίσεων. Έπειτα τις ομαδοποιεί σε τάσεις με βάση τις όμοιες εμφανίσεις τους. Μια τάση είναι ένα σύνολο από λέξεις που εμφανίζονται συχνά μαζί σε tweets. Αφού βρεθεί μια τάση, το TwitterMonitor επεξεργάζεται τα tweets που ανήκουν στο trend και βρίσκει επιπλέον πληροφορίες για αυτό. Η παρούσα διπλωματική εξετάζει εξ' αρχής όλο το περιεχόμενο των tweets τα οποία έχουμε στη διάθεσή μας, χωρίς να αναζητά ξαφνικές πολυπληθείς εμφανίσεις λέξεων. Η λογική της επεξεργασίας όμως είναι η ίδια, καθώς οι ομάδες φτιάχνονται βρίσκοντας λέξεις που εμφανίζονται συχνά μαζί.

2.2 Ανάλυση μόνο των κατηγορικών δεδομένων του Twitter

Οι Ghalem et al.(2014) μελέτησαν πώς διαφοροποιούνται τα κατηγορικά χαρακτηριστικά των tweets στο χώρο και στο χρόνο. Κατηγορικά είναι τα χαρακτηριστικά ενός tweet που προκαθορίζονται από χαρακτηριστικά του ίδιου του χρήστη και της συσκευής από την οποία γράφει το tweet. Τέτοια είναι η γλώσσα στην οποία γράφει (την οποία έχει ορίσει ο ίδιος στις επιλογές του τηλεφώνου του), το λειτουργικό σύστημα της συσκευής, η ίδια η συσκευή (αν είναι υπολογιστής γραφείου ή smartphone) και η εφαρμογή από την οποία δημοσιεύεται το tweet (web browser/twitter application). Στην εργασία αυτή καταδείχτηκαν οι διαφορές που υπάρχουν ως προς τέτοια κατηγορικά χαρακτηριστικά σε περιοχές που ξεκινούν από γειτονιές μιας πόλης και φτάνουν σε διαφορετικές χώρες. Στην παρούσα διπλωματική δεν εξετάστηκε κανένα κατηγορικό χαρακτηριστικό πέραν των γεωγραφικών συντεταγμένων καθώς σκοπός είναι η εξαγωγή θεμάτων και συναισθημάτων και όχι η ανάλυση των διαφορετικών χαρακτηριστικών των χρηστών. Επιπροσθέτως, τα κατηγορικά χαρακτηριστικά λαμβάνονται απευθείας από το χρήστη και τη συσκευή του με τις επιλογές που έχει προκαθορίσει. Από τους Thanaa M. Ghalem et al. δεν εξετάζεται το περιεχόμενο του κειμένου για να βρεθούν δυναμικά τα θέματα και τα συναισθήματα. Η μέθοδος όμως παρουσίασης των αποτελεσμάτων σε χάρτες των περιοχών από τις οποίες έχουν δημοσιευτεί τα tweets είναι αρκετά όμοια με της παρούσας εργασίας.

2.3 Ανάλυση συναισθήματος

Οι Agarval και Sabharwal (2012) ασχολούνται με τη συναισθηματική ανάλυση δεδομένων του Twitter. Κατατάσσουν τα tweets σε μια από 4 κατηγορίες: θετικά, αρνητικά, ουδέτερα, αντικειμενικά. Το κάνουν αυτό έχοντας εξ' αρχής κατηγοριοποιημένα tweets τα οποία χρησιμοποιούν για να φτιάξουν 4 ταξινομητές: Ο καθένας εξετάζει αν το tweet ανήκει σε αυτόν, ή αν δεν ανήκει. Έτσι φτιάχνουν ένα μοντέλο που εκπαιδεύεται και τελικώς μπορεί να ταξινομήσει ένα νέο tweet σε μια από τις 4 κατηγορίες. Σε αντίθεση με τους Agarval και Sabharwal, στην παρούσα διπλωματική εκπαιδεύτηκε ένας ταξινομητής που δίνει ένα θετικό και ένα αρνητικό βάρος σε κάθε Tweet. Ανάλογα με το βάρος το tweet κατατάσσεται σε θετικό ή αρνητικό, ή αν τα βάρη δεν διαφέρουν πολύ θεωρείται ότι το συναίσθημα είναι ουδέτερο ή δεν έχει σημασία να εξετάσουμε το συγκεκριμένο tweet για συναίσθημα.

Άλλοι, όπως οι Go et al. (2009) Ταξινομούν τα tweets σε θετικά ή αρνητικά χρησιμοποιώντας τα emoticons. Οι ταξινομητές μηχανικής μάθησης που χρησιμοποιούνται είναι οι: Naive Bayes που χρησιμοποιείται και στην παρούσα διπλωματική εργασία, Maximum Entropy, και Support Vector Machines.

3 Θεωρητικό και τεχνικό υπόβαθρο

Για να υλοποιηθούν τα μέρη της παρούσας εργασίας χρησιμοποιήθηκαν οι εξής τεχνικές:

1. Μετά τη λήψη των tweets από το Twitter, ήταν αναγκαία η επεξεργασία τους ώστε να καθαριστούν από τις "άχρηστες" λέξεις. Αυτό έγινε με χρήση τεχνικών επεξεργασίας φυσικής γλώσσας και πιο συγκεκριμένα με Part-of-speech-tagging.

2. Ύστερα, για την εξαγωγή θεμάτων και συναισθημάτων χρησιμοποιήθηκαν συγκεκριμένοι αλγόριθμοι και οι αντίστοιχες υλοποιήσεις τους στο πλαίσιο του Apache Mahout. Το Mahout τρέχει πάνω στην πλατφόρμα Apache Hadoop. Ειδικότερα, το Mahout προσφέρει υλοποιήσεις αλγορίθμων ομαδοποίησης (clustering) ώστε να εξαχθούν τα κοινά θέματα από τα διάφορα tweets

3. Το Mahout προσφέρει επίσης υλοποιήσεις αλγορίθμων ταξινόμησης (classification) που επιτρέπουν την επεξεργασία κειμένου και την κατανομή του σε κατηγορίες, εν προκειμένω σε θετικό ή αρνητικό συναίσθημα.

4. Για την επεξεργασία των αρχείων κειμένων από το Mahout γίνεται η μετατροπή τους σε διανύσματα TF-IDF.

5. Ο αριθμός k των θεματικών ομάδων στο clustering δίδεται στον αλγόριθμο πριν την εκτέλεση. Για να υπολογιστεί το k χρησιμοποιήθηκε η μέθοδος του αγκώνα (Elbow Method)

6. Επίσης έγινε χρήση της τεχνικής cosine similarity ώστε να βρεθούν ομοιότητες-συσχετίσεις μεταξύ των θεμάτων διαφορετικών ημερών με στόχο την παρακολούθηση της διαφοροποίησης των θεμάτων στο χρόνο.

7. Χρησιμοποιήθηκε η βάση δεδομένων για γράφους neo4j και η γλώσσα Cypher Query Language για την εισαγωγή των δεδομένων σε αυτήν.

8. Για την επικοινωνία της web εφαρμογής μας με τη βάση neo4j γίνονται REST calls. Η βάση neo4j προσφέρει έτοιμο REST API.

3.1 Επεξεργασία φυσικής γλώσσας

Η Επεξεργασία Φυσικής Γλώσσας (Natural Language Processing - NLP) είναι ένας τομέας της επιστήμης των υπολογιστών, τεχνητής νοημοσύνης και υπολογιστικής γλωσσολογίας που ασχολείται με την αλληλεπίδραση ανάμεσα στους υπολογιστές και στις ανθρώπινες (φυσικές) γλώσσες. Έτσι το NLP σχετίζεται με το χώρο της αλληλεπίδρασης ανθρώπου-μηχανής. Πολλά προβλήματα του NLP περιλαμβάνουν την κατανόηση της φυσικής γλώσσας, δηλαδή την ικανότητα του υπολογιστή να βγάλει νόημα από ανθρώπινη γλώσσα. Άλλα προβλήματα περιλαμβάνουν την παραγωγή φυσικής γλώσσας.

Στη γλωσσολογία, η επισημείωση μερών του λόγου (part-of-speech tagging, POST) είναι η διαδικασία της σήμανσης μιας λέξης σε ένα κείμενο σαν μέρος του λόγου. Η σήμανση αυτή βασίζεται τόσο στη σημασία της λέξης όσο και στα συμφραζόμενα, δηλαδή στη σχέση της με γειτονικές λέξεις σε μια φράση ή πρόταση. Το POS tagging γίνεται στο πλαίσιο της

υπολογιστικής γλωσσολογίας με τη χρήση αλγορίθμων οι οποίοι συνδέουν διακριτούς όρους και κρυφά μέρη του λόγου με ένα σύνολο από περιγραφικές ετικέτες (tags).

3.2 Apache Mahout και Hadoop

Το Mahout είναι ένα project του Apache Software Foundation γραμμένο σε java για τη δημιουργία δωρεάν εφαρμογών κατανεμημένων/επεκτάσιμων αλγορίθμων μηχανικής μάθησης. Οι αλγόριθμοι αυτοί επικεντρώνονται κυρίως στους τομείς της συνεργατικής διήθησης (collaborative filtering), της ομαδοποίησης clustering) και της κατάταξης (classification). Πολλές από τις εφαρμογές του Mahout χρησιμοποιούν την πλατφόρμα Apache Hadoop (hadoop.apache.org). Το Mahout περιέχει επίσης βιβλιοθήκες java για την εκτέλεση μαθηματικών πράξεων (κυρίως της γραμμικής άλγεβρας και της στατιστικής). Το Mahout αναπτύσσεται συνεχώς και αριθμός των αλγορίθμων που περιλαμβάνει αυξάνεται ταχύτατα. Αν και οι αλγόριθμοι του Mahout εφαρμόζονται πάνω στο Apache Hadoop χρησιμοποιώντας τη μέθοδο map/reduce, υπάρχουν εφαρμογές που τρέχουν αλγορίθμους του Mahout χωρις να χρειάζεται το Hadoop.

Το Apache Hadoop είναι ένα πλαίσιο λογισμικού ανοιχτού κώδικα γραμμένο σε java με σκοπό τη διαμοιρασμένη αποθήκευση και επεξεργασία πολύ μεγάλου όγκου δεδομένων σε συστοιχίες υπολογιστών. Όλα τα μέρη του Hadoop έχουν σχεδιαστεί με την υπόθεση ότι οι αστοχίες υλικού (μεμονωμένων μηχανημάτων η και μεγαλύτερων συνόλων) είναι κοινές και γι' αυτό θα πρέπει να τις χειρίζεται αυτόματα το λογισμικό.

Ο πυρήνας του Apache Hadoop αποτελείται από το κομμάτι της αποθήκευσης (Hadoop Distributed File System HDFS) και το κομμάτι της επεξεργασίας (MapReduce). Το Hadoop χωρίζει τα αρχεία σε μεγάλες ομάδες και τα κατανέμει σε υπολογιστικούς κόμβους. Για την επεξεργασία, το MapReduce μεταφέρει κώδικα στους κόμβους για να τον επεξεργάζονται παράλληλα, ανάλογα με τα δεδομένα που χρειάζεται κάθε κόμβος να επεξεργαστεί. Η προσέγγιση αυτή επωφελείται από την τοπικότητα των δεδομένων (κόμβοι διαχειρίζονται τα δεδομένα που έχουν διαθέσιμα) με αποτέλεσμα η επεξεργασία του συνόλου των δεδομένων να γίνεται ταχύτερα και πιο αποδοτικά απ' όσο αν γινόταν σε ένα συμβατικό υπερυπολογιστή βασισμένο σε ένα παράλληλο σύστημα αρχείων όπου οι υπολογισμοί και τα δεδομένα συνδέονται με υψηλής ταχύτητας δίκτυο.

Το πλαίσιο του Hadoop περιέχει τα ακόλουθα βασικά μέρη:

- Hadoop Common - Περιέχει βιβλιοθήκες και άλλα εργαλεία που χρησιμοποιούνται από τα υπόλοιπα συστήματα του Hadoop.
- Hadoop Distributed File System (HDFS) - ένα διαμοιρασμένο σύστημα αρχείων που αποθηκεύει δεδομένα σε απλά μηχανήματα και παρέχει υψηλή αθροιστικό εύρος ζώνης στην υπολογιστική συστάδα (cluster).
- Hadoop YARN - μια πλατφόρμα διαχείρισης πόρων υπεύθυνη για τη διαχείριση των πόρων στους clusters
- Hadoop MapReduce - ένα προγραμματιστικό μοντέλο για επεξεργασία δεδομένων μεγάλης κλίμακας.

Στην εργασία αυτή το Hadoop χρησιμοποιήθηκε πάνω σε ένα κόμβο (single-node) που στήθηκε σε virtual machine στην υπηρεσία okeanos της ΕΔΕΤ Α.Ε.

Το Apache Mahout επελέγη γιατί περιέχει υλοποιήσεις των απαραίτητων αλγορίθμων για την επεξεργασία των δεδομένων της παρούσας διπλωματικής εργασίας. Επίσης είναι λογισμικό ανοιχτού κώδικα και διανέμεται δωρεάν. Το Mahout υλοποιείται πάνω στο Hadoop, προσφέροντας τη δυνατότητα κατανομής τεράστιου όγκου δεδομένων σε πολλούς κόμβους και την επεξεργασία τους σε αυτούς.

3.3 Αλγόριθμοι του Apache Mahout για ομαδοποίηση (clustering)

Το Mahout περιέχει τους εξής αλγορίθμους clustering:

- K-means
- Fuzzy k-means
- Streaming k-means

3.3.1 K-means

Το K-Means clustering είναι μια μέθοδος διανυσματικής κβάντωσης (vector quantization), προερχόμενη από την επεξεργασία σημάτων, που χρησιμοποιείται για την ομαδοποίηση (clustering analysis) στην εξόρυξη δεδομένων (data mining). Το k-means clustering κατανέμει n παρατηρήσεις σε k clusters όπου κάθε παρατήρηση ανήκει στο cluster με το πλησιέστερο κέντρο.

Η διαδικασία έχει ως εξής: Κάθε παρατήρηση-αντικείμενο αναπαρίσταται με ένα διάνυσμα περιεχομένων σε ένα χώρο n διαστάσεων, όπου n ο αριθμός όλων των περιεχομένων που χρησιμοποιούνται για να περιγράψουν τα αντικείμενα στον cluster. Ο αλγόριθμος έπειτα επιλέγει τυχαία k σημεία μέσα σε αυτό το διανυσματικό χώρο, τα οποία είναι και τα αρχικά κέντρα των clusters. Ύστερα κάθε αντικείμενο κατανέμεται στον cluster του οποίου το κέντρο είναι πιο κοντά στο διάνυσμα αναπαράστασης. Ο τρόπος μέτρησης της απόστασης επιλέγεται από το χρήστη.

Στη συνέχεια για κάθε cluster υπολογίζεται ένα νέο κέντρο παίρνοντας τη μέση τιμή όλων των διανυσμάτων αναπαράστασης κάθε αντικειμένου που ανήκει στον cluster. Η διαδικασία κατανομής των αντικειμένων και επανυπολογισμού των κέντρων επαναλαμβάνεται έως ότου η διαδικασία να συγκλίνει. Αποδεικνύεται ότι ο αλγόριθμος συγκλίνει ύστερα από πεπερασμένο αριθμό υπολογισμών.

3.3.2 Fuzzy k-means

Ο αλγόριθμος Fuzzy K-Means είναι μια επέκταση του K-Means. Ενώ ο K-Means τοποθετεί ένα αντικείμενο σε ένα μόνο cluster, ο Fuzzy K-Means βρίσκει ομάδες όπου ένα αντικείμενο μπορεί να ανήκει σε περισσότερους από ένα clusters με κάποια πιθανότητα.

Η διαδικασία έχει ως εξής: Όπως και στον απλό K-Means, κάθε παρατήρηση-αντικείμενο αναπαρίσταται με ένα διάνυσμα περιεχομένων σε ένα χώρο n διαστάσεων, όπου n ο αριθμός όλων των περιεχομένων που χρησιμοποιούνται για να περιγράψουν τα αντικείμενα στον cluster. Ο αλγόριθμος έπειτα επιλέγει τυχαία k σημεία μέσα σε αυτό το διανυσματικό χώρο, τα οποία είναι και τα αρχικά κέντρα των clusters. Για κάθε ζεύγος αντικειμένου και cluster υπολογίζεται η πιθανότητα να ανήκει το αντικείμενο σε αυτό τον cluster και αποθηκεύεται σε συντελεστές. Στη συνέχεια για κάθε cluster υπολογίζεται ένα νέο κέντρο παίρνοντας τη μέση τιμή όλων των διανυσμάτων αναπαράστασης κάθε αντικειμένου που ανήκει στον cluster, σταθμισμένη με το βαθμό (την πιθανότητα) που το κάθε αντικείμενο ανήκει στον cluster

$$c_k = \frac{\sum_x w_k(x)^m x}{\sum_x w_k(x)^m}$$

Ο βαθμός Wk που ένα αντικείμενο x ανήκει σε ένα cluster είναι αντιστρόφως ανάλογος της απόστασης του x από το κέντρο του cluster όπως αυτή υπολογίστηκε στο εκάστοτε προηγούμενο βήμα. Επίσης εξαρτάται από μια παράμετρο m που προσδιορίζει πόσο βάρος δίνεται στο πιο κοντινό κέντρο.

Η διαδικασία κατανομής των αντικειμένων και επανυπολογισμού των κέντρων επαναλαμβάνεται έως ότου η διαδικασία να συγκλίνει (δηλαδή μέχρι η μεταβολή των συντελεστών που δείχνουν την πιθανότητα να ανήκει ένα αντικείμενο σε έναν cluster μεταξύ

δύο υπολογισμών να είναι μικρότερη από ένα δοσμένο κατώφλι ευαισθησίας (sensitivity threshold).

3.3.3 Streaming k-means

Ο αλγόριθμος Streaming K-Means αποτελείται από 2 βήματα:

1. Βήμα streaming
2. Βήμα BallKMeans

Το πρώτο βήμα είναι ένας τυχαιοποιημένος αλγόριθμος που εξετάζει μια φορά τα δεδομένα και παράγει έναν αριθμό κέντρων που θεωρεί βέλτιστο. Αν ο όγκος των δεδομένων προς επεξεργασία είναι n και το αναμενόμενο πλήθος clusters είναι k, το βήμα streaming θα παραγάγει περίπου k*log(n) clusters οι οποίοι θα προωθηθούν στο βήμα BallKMeans στο οποίο θα μειωθεί περαιτέρω ο αριθμός των clusters σε k.

3.3.4 Επιλογή k-means

Για την ομαδοποίηση των δεδομένων επελέγη ο απλός αλγόριθμος K-Means. Ο απλός k-means παράγει παρόμοια αποτελέσματα με τον fuzzy για τα δεδομένα που δίνουμε σαν είσοδο, ενώ δεν μας χρειάζεται να ξέρουμε αν ένα tweet ανήκει σε περισσότερους από ένα clusters με διαφορετική πιθανότητα καθώς θέλουμε να κατανείμουμε κάθε tweet σε ένα cluster.

3.4 Mahout Naive Bayes Classification

3.4.1 Ο Ταξινομητής Bayes

Στη μηχανική μάθηση,οι ταξινομητές naive Bayes είναι μια οικογένεια από απλούς πιθανοτικούς ταξινομητές που βασίζονται στην εφαρμογή του θεωρήματος του Bayes με ισχυρές (αφελείς-naive) υποθέσεις ανεξαρτησίας μεταξύ των χαρακτηριστικών.

Ο Naive Bayes έχει μελετηθεί εκτενώς από τη δεκαετία του 1950. Εισήχθη με άλλο όνομα στην κοινότητα ανάκτησης κειμένου στην αρχή της δεκαετίας του '60 και παραμένει μια δημοφιλής μέθοδος για κατηγοριοποίηση κειμένου: το πρόβλημα της απόφασης για το αν αρχεία ανήκουν σε μια κατηγορία ή σε άλλη (πχ αν είναι spam ή έγκυρο, αν μιλάει για αθλητικά ή πολιτική, κλπ) με τις συχνότητες των λέξεων σαν χαρακτηριστικά.

Το Mahout προσφέρει δύο υλοποιήσεις του Naive Bayes. Η πρώτη είναι ο πρότυπος Multinomial Naive Bayes. Η δεύτερη είναι μια υλοποίηση του Transformed Weight-normalized Complement Naive Bayes όπως εισήχθη από τους Rennie et al. (2003).

Ενώ ο Naive Bayes έχει εδώ και καιρό καθιερωθεί για την ταξινόμηση κειμένου, ο Complement Naive Bayes (CBayes) είναι μια επέκτασή του που αποδίδει ιδιατέρως καλά σε σύνολα δεδομένων με μη ισορροπημένες κλάσεις.

3.4.2 Τα βήματα της ταξινόμησης

Ο ταξινομητής Bayes χρειάζεται πρώτα να "πεκπαιδευτεί" με ένα σύνολο αρχείων τα οποία είναι ήδη ταξινομημένα. Ύστερα με βάση τις πληροφορίες που έχει συλλέξει από την επεξεργασία των έτοιμων δεδομένων θα μπορέσει να ταξινομήσει κάποια νέα που θα του δοθούν.

Τα βήματα για την εκτέλεση του αλγορίθμου είναι τα ακόλουθα:

- Έστω $\vec{d} = (\vec{d}_1, ..., \vec{d}_n)$ ένα σύνολο αρχείων. $\mathbf{d_{ij}}$ είναι ο αριθμός εμφανίσεων της λέξης **i** στο αρχείο **j**.
- Έστω $\vec{y} = (\vec{y}_1, ..., \vec{y}_n)$ οι ετικέτες τους (δηλαδή το πού έχουν ταξινομηθεί) .
- Έστω $\boldsymbol{\alpha_i}$ μια παράμετρος ομαλοποίησης για κάθε λέξη **i** στο λεξικό. Έστω $a = \sum_{i=1}^{n} (a_i)$
- **Προεπεξεργασία**: Μέσω της εντολής seq2sparse γίνεται ο μετασχηματισμός TF-IDF και η κανονικοποίηση του μήκους του διανύσματος **d**.

$$d_{ij} = \sqrt{(d_{ij})}$$

$$d_{ij} = d_{ij}\left(\log \frac{\sum_k 1}{\sum_k (\delta_{ik} + 1)} + 1\right)$$

$$d_{ij} = \frac{d_{ij}}{\sqrt{\sum_k d_{kj}^2}}$$

- **Training: Bayes(d,y)** υπολογίζονται τα βάρη $\mathbf{W_{ci}}$ κάθε όρου όπως ακολούθως:

$$\theta_{ci} = \frac{d_{ic} + a_i}{\sum_k d_{kc} + a}$$

$$w_{ci} = log\theta_{ci}$$

- **Training: Complement Bayes(d,y)** υπολογίζονται τα βάρη **Wci** κάθε όρου όπως ακολούθως:

$$\theta_{ci} = \frac{\sum_{j=y_j \neq c} d_{ij} + a_i}{\sum_{j=y_j \neq c} \sum_k d_{kj} + a}$$

$$w_{ci} = -log\theta_{ci}$$

$$w_{ci} = \frac{w_{ci}}{\sum_i |(w_{ci})|}$$

Τώρα ο αλγόριθμος έχει "εκπαιδευτεί" και είναι έτοιμος να ελέγξει και να ταξινομήσει τα νέα αρχεία που θα του δωθούν.

- **Έλεγχος (Testing) και ταξινόμηση:** Έστω $\vec{t} = (\vec{t_1}, \ldots, \vec{t_n})$ ένα αρχείο προς έλεγχο όπου $\mathbf{t_i}$ ο αριθμός εμφανίσεων της λέξης i. Ταξινομείται το αρχείο σύμφωνα με τον τύπο:

$$l(t) = argmax_c \sum_i t_i w_{ci}$$

Όπως φαίνεται, η κύρια διαφορά μεταξύ Bayes και CBayes είναι στον υπολογισμό των βαρών **Wci**. Ενώ ο Bayes δίνει μεγαλύτερη σημασία στην πιθανότητα μια λέξη να ανήκει σε μια τάξη A, ο CBayes μεγιστοποιεί τα βάρη των λέξεων με βάση την πιθανότητα να μην ανήκουν σε οποιαδήποτε άλλη κλάση εκτός της A.

3.4.3 Επιλογή naive Bayes

Λόγω του διαφορετικού υπολογισμού των βαρών Wci, ο αλγόριθμος CBayes παράγει ορθότερο αποτέλεσμα από τον απλό Bayes όταν οι κλάσεις στην εκμάθηση είναι άνισες, δηλαδή όταν ο αριθμός των θετικών και των αρνητικών tweets που δίνονται για training δεν είναι ίδιος. Στην παρούσα διπλωματική εργασία επελέγη ο αλγόριθμος Naive Bayes για να γίνει η ταξινόμηση των tweets, καθώς είχαμε ίδιο αριθμό αρνητικών και θετικών αρχείων στην εκπαίδευση.

3.5 Διανύσματα TF-IDF

3.5.1

Η Συχνότητα Όρου - Αντίστροφη Συχνότητα Όρου είναι ένας αριθμός που δείχνει πόσο σημαντικός είναι ένας όρος (λέξη) για ένα αρχείο μέσα σε μια συλλογή από αρχεία. Η τιμή tf-idf αυξάνεται ανάλογα με τον αριθμό εμφανίσεων μιας λέξης μέσα στο αρχείο, αλλά αντισταθμίζεται από τη συχνότητα με την οποία η λέξη εμφανίζεται στη συλλογή όλων των αρχείων, ώστε λέξεις που εμφανίζονται σε μεγάλο αριθμό αρχείων να μη θεωρούνται τόσο σημαντικές όσο οι πιο σπάνιες. Αν μια σπάνια λέξη εμφανιστεί σε ένα αρχείο, σημαίνει ότι το αρχείο πολύ πιθανώς χαρακτηρίζεται (και) από τη λέξη αυτή.

3.5.2

Το Term Frequency κάποιου όρου υπολογίζεται για ένα αρχείο ως ο αριθμός εμφανίσεων αυτού του όρου μέσα στο αρχείο προς το συνολικό αριθμό όρων στο αρχείο αυτό.

3.5.3

Το Inverse Document Frequency κάποιου όρου υπολογίζεται για τη συλλογή αρχείων από τον τύπο: log(numDocs/(docFreq+1)) + 1.0

όπου numDocs: ο συνολικός αριθμός των αρχείων στη συλλογή, docFreq: ο αριθμός των αρχείων στα οποία εμφανίζεται ο όρος που εξετάζουμε.

3.5.4

Το Term Frequency πολλαπλασιάζεται με το Inverse Document Frequency και για κάθε όρο σε κάθε αρχείο παίρνουμε την τιμή tfidf. Ύστερα για κάθε αρχείο φτιάχνεται ένας πίνακας με τις τιμές tfidf. Στον πίνακα αυτόν, για κάθε όρο που εμφανίζεται στη συλλογή από αρχεία, αποθηκεύεται η τιμή tfidf του συγκεκριμένου όρου. Αν κάποιος όρος δεν εμφανίζεται στο συγκεκριμένο αρχείο, θα έχει tfidf=0 καθώς το Term Frequency θα είναι 0 για το συγκεκριμένο αρχείο. Έτσι κάθε όρος της συλλογής αρχείων που εξετάζουμε μετατρέπεται σε μια διάσταση του διανύσματος TF-IDF.

3.6 Elbow Method

Αρχικά, υπολογίζεται το άθροισμα τετραγώνων των αποκλίσεων sum of squared error (SSE) για διάφορες τιμές του k. Το SSE είναι το άθροισμα του τετραγώνου της απόστασης μεταξύ κάθε μέλους *x* ενός cluster και του κέντρου *ci* του cluster. Μαθηματικά:

$$SSE = \sum_{i=1}^{K} \sum_{x \in c_i} dist(x, c_i)^2$$

Αναπαριστώντας σε διάγραμμα τη σχέση μεταξύ του k και του SSE είναι φανερό ότι το error μειώνεται καθώς αυξάνεται το k. Αυτό συμβαίνει γιατί όταν ο αριθμός τους μεγαλώνει, οι clusters θα πρέπει να είναι πιο μικροί, άρα η απόκλιση είναι επίσης μικρότερη. Επιλέγεται το k μετά το οποίο το SSE σταματά να μειώνεται τόσο απότομα. Αυτό σημαίνει ότι αν προστεθούν κι άλλοι clusters επιπλέον από k, δεν λαμβάνονται πολύ διαφορετικά αποτελέσματα. Σε εκείνο το σημείο η γραφική παράσταση θα έχει τη μορφή "αγκώνα".

3.7 Cosine Similarity

Το cosine similarity είναι ένα μέτρο ομοιότητας μεταξύ δυο διανυσμάτων και μετράει το συνημίτονο της μεταξύ τους γωνίας θ.

$$similarity = \cos(\theta) = \frac{A * B}{\|A\| \|B\|}$$

cos(0)=1 , cos(Θ)<1 για 0<θ<2π.

Συγκρίνεται δηλαδή η κατεύθυνση των διανυσμάτων και όχι το μέτρο τους. Δυο διανύσματα με ίδια κατεύθυνση έχουν cosineSimilarity=1. Δύο διανύσματα με γωνία 90° έχουν cosineSimilarity=0 και δύο διανύσματα με ίδια διεύθυνση και αντίθετη φορά έχουν cosineSimilarity=-1, ανεξάρτητα του μέτρου τους. Το CosineSimilarity χρησιμοποιείται κυρίως στο θετικό χώρο, όπου το αποτέλεσμα είναι στο [0,1].

Τα όρια αυτά ισχύουν για οποιοδήποτε αριθμό διαστάσεων, και το CosineSimilarity εφαρμόζεται κυρίως σε πολυδιάστατους θετικούς χώρους. Στην περίπτωση της άντλησης πληροφορίας από κείμενο, κάθε όρος αναπαριστάται από διαφορετική διάσταση και ένα αρχείο χαρακτηρίζεται από ένα διάνυσμα όπου η τιμή κάθε διάστασης αντιστοιχεί στον αριθμό εμφανίσεων που έχει αυτός ο όρος μέσα στο αρχείο. Το cosine similarity δίνει έτσι μια χρήσιμη μέτρηση του πόσο όμοια είναι δύο αρχεία ως προς το θέμα τους.

3.8 Η βάση neo4j και η γλώσσα cypher

Η Neo4j είναι μια βάση δεδομένων για γράφους, ανοιχτού λογισμικού, υλοποιημένη σε java. Αναπτύχθηκε από την Neo Technology Inc., και το version 1.0 κυκλοφόρησε το Φεβρουάριο του 2010. Η Neo4j απηθηκεύει τα δεδομένα σε γράφους αντί για πίνακες. Είναι η δημοφιλέστερη graph database.

Στη Neo4j, τα πάντα αποθηκεύονται με μορφή ακμής, κόμβου ή γνωρίσματος (attribute). Κάθε κόμβος και ακμή μπορεί να έχει απεριόριστο αριθμό από γνωρίσματα. Και οι κόμβοι και οι ακμές μπορούν να χαρακτηριστούν με ετικέτες. Οι ετικέτες είναι χρήσιμες γιατί επιτρέπουν τον περιορισμό της περιοχής αναζήτησης σε συγκεκριμένα labels.

Η Cypher είναι μια δηλωτική γλώσσα ερωτημάτων για γράφους που χρησιμοποιείται από τη βάση Neo4j. Επιτρέπει να γίνονται αποδοτικά τα ερωτήματα στη βάση και η ενημέρωση των γράφων. Πολύπλοκα ερωτήματα μπορούν να εκφραστούν εύκολα μέσω της Cypher.

3.9 REST API, REST calls προς τη βάση neo4j

Το REpresentational State Transfer (REST) είναι ένα στυλ αρχιτεκτονικής λογισμικού που αποτελείται από κατευθυντήριες γραμμές και πρακτικές για τη δημιουργία επεκτάσιμων υπηρεσιών διαδικτύου. Το REST είναι ένα σύνολο περιορισμών που εφαρμόζεται στη σχεδίαση των μερών σε ένα διαμοιρασμένο σύστημα υπερμέσων (υπερμέσα: graphics, audio, video, plain text and hyperlinks) που μπορεί να οδηγήσει σε μια πιο αποδοτική και εύκολα συντηρήσιμη αρχιτεκτονική.

Το REST είναι πια ευρέως αποδεκτό σαν μια απλούστερη εναλλακτική αντί του SOAP.

Τα συστήματα RESTful επικοινωνούν συνήθως μέσω του πρωτοκόλλου HTTP με τα ίδια HTTP verbs (GET, POST, PUT, DELETE...) που χρησιμοποιούν και οι web browsers για να λάβουν σελίδες του διαδικτύου και να στείλουν δεδομένα σε απομακρυσμένους servers.

Το αρχιτεκτονικό στυλ REST αναπτύχθηκε από το W3C Technical ARchitecture Group παράλληλα με το HTTP 1.1. Ο παγκόσμιος ιστός αναπαριστά τη μεγαλύτερη υλοποίηση ενός συστήματος που συμμορφώνεται με το REST.

Εφαρμογή σε υπηρεσίες ιστού (web)

Τα APIs υπηρεσιών ιστού που συμμορφώνονται προς τους περιορισμούς της αρχιτεκτονικής REST ονομάζονται RESTful APIs. Τα RESTful APIs που βασίζονται στο πρωτόκολλο HTTP προσδιορίζονται από τις ακόλουθες πτυχές:

- base URI, όπως `http://example.com/resources/`
- ένα τύπο Internet media για τα δεδομένα. Ο τύπος είναι συνήθως JSON, αλλά μπορεί να είναι οποιοσδήποτε άλλος τύπος internet media (πχ XML, Atom, microformats, images, κλπ.)
- μεθόδους HTTP (πχ GET, PUT, POST, ή DELETE)
- συνδέσμους υπερκειμένου που αναφέρουν κατάσταση
- συνδέσμους υπερκειμένου προς σχετικούς πόρους

Η βάση neo4j δέχεται και επιστρέφει με το REST API της δεδομένα σε τύπο JSON. Παράδειγμα HTTP κλήσης και απάντησης:

Example request

```
        POST
        http://localhost:7474/db/data/transaction/commit
{           Accept: application/json; charset=UTF-8
    "statements" : [ {
            "statement" : "CREATE (n) RETURN id(n)"
    } ]
}
```

Example response

```
200: OK

   Content-Type: application/json


{
        "results" : [ {
                "columns" : [ "id(n)" ],
                "data" : [ {
                        "row" : [ 15 ]
                } ]
        } ],
        "errors" : [ ]
}
```

Τα statements και τα results είναι json objects. Εδώ έχουμε μόνο ένα statement και ένα result.

4 Ανάλυση Απαιτήσεων Συστήματος

Στο κεφάλαιο αυτό ακολουθεί η περιγραφή της αρχιτεκτονικής του συστήματος και γίνεται η ανάλυση απαιτήσεων για τις λειτουργίες του.

4.1 Αρχιτεκτονική

Το σύστημά μας αποτελείται από τα ακόλουθα επιμέρους υποσυστήματα:

1. Εφαρμογή Java για τη συλλογή δεδομένων από το Twitter: Χρησιμοποιήθηκε κώδικας από το twitter4j.org για τη συλλογή tweets από τις πόλεις του Λονδίνου, της Νέας Υόρκης και του Los Angeles κρατώντας τη χωρική πληροφορία (δηλαδή συντεταγμένες) σε όσα την περιείχαν.

2. Επεξεργασία κειμένου με χρήση Natural Language Processing Software: Έγινε επεξεργασία των tweets με σκοπό από ένα tweet να δημιουργηθούν δύο παράγωγα: #Tweet1 όπου έχουν απομείνει μόνο τα ουσιαστικά και τα hashtags. Οι υπόλοιποι όροι έχουν αφαιρεθεί ώστε να γίνει η θεματική ανάλυση. #Tweet2 όπου έχουν αφαιρεθεί μόνο τα links και τυχόν όροι που δήλωναν re-tweet ή check-in σε χώρο, ώστε να γίνει η ανάλυση συναισθήματος.

3. Επεξεργασία όλων των Tweets της μορφής 1 χρησιμοποιώντας το Apache Mahout. Κάνοντας k-means clustering, κατανέμονται τα tweets σε clusters ανά θέμα ώστε κάθε cluster να περιέχει tweets που μιλούν για κάτι κοινό. Για κάθε cluster λαμβάνονται οι κυριότεροι-αντιπροσωπευτικότεροι 30 όροι με τα βάρη του καθενός (δηλαδή το πόσο σημαντικός-αντιπροσωπευτικός είναι ένας όρος για το συγκεκριμένο cluster.

4. Επεξεργασία όλων των Tweets της μορφής 2 χρησιμοποιώντας το Apache Mahout (https://mahout.apache.org/). Κάνοντας training χρησιμοποιώντας ένα dataset με ήδη ταξινομημένα tweets και ύστερα classification με τον αλγόριθμο Complement Naive Bayes κατανέμονται τα tweets σε positive και negative, κάθε ένα έχοντας έναν αριθμό-βάρος που δείχνει το πόσο "κοντά" είναι στο negative και έναν άλλο αριθμό-βάρος που δείχνει πόσο κοντά είναι στο positive.

5. Επεξεργασία των αποτελεσμάτων του clustering (χρήση java) με σκοπό την εύρεση ομοιοτήτων ανάμεσα σε ομάδες διαφορετικών ημερών.

6. Επεξεργασία των αποτελεσμάτων του clustering (χρήση java) με σκοπό τη δημιουργία αρχείου .csv που να περιέχει τα απαραίτητα δεδομένα σχετικά με τα topics-clusters και τη σχέση τους με κάθε tweet και μεταξύ τους, ώστε να φτιαχτεί η βάση δεδομένων της εφαρμογής μας.

7. Επεξεργασία των αποτελεσμάτων του classification με σκοπό τη δημιουργία αρχείου .csv που να περιέχει για κάθε tweet τους αριθμούς που εκφράζουν το πόσο αρνητικό και το πόσο θετικό είναι.

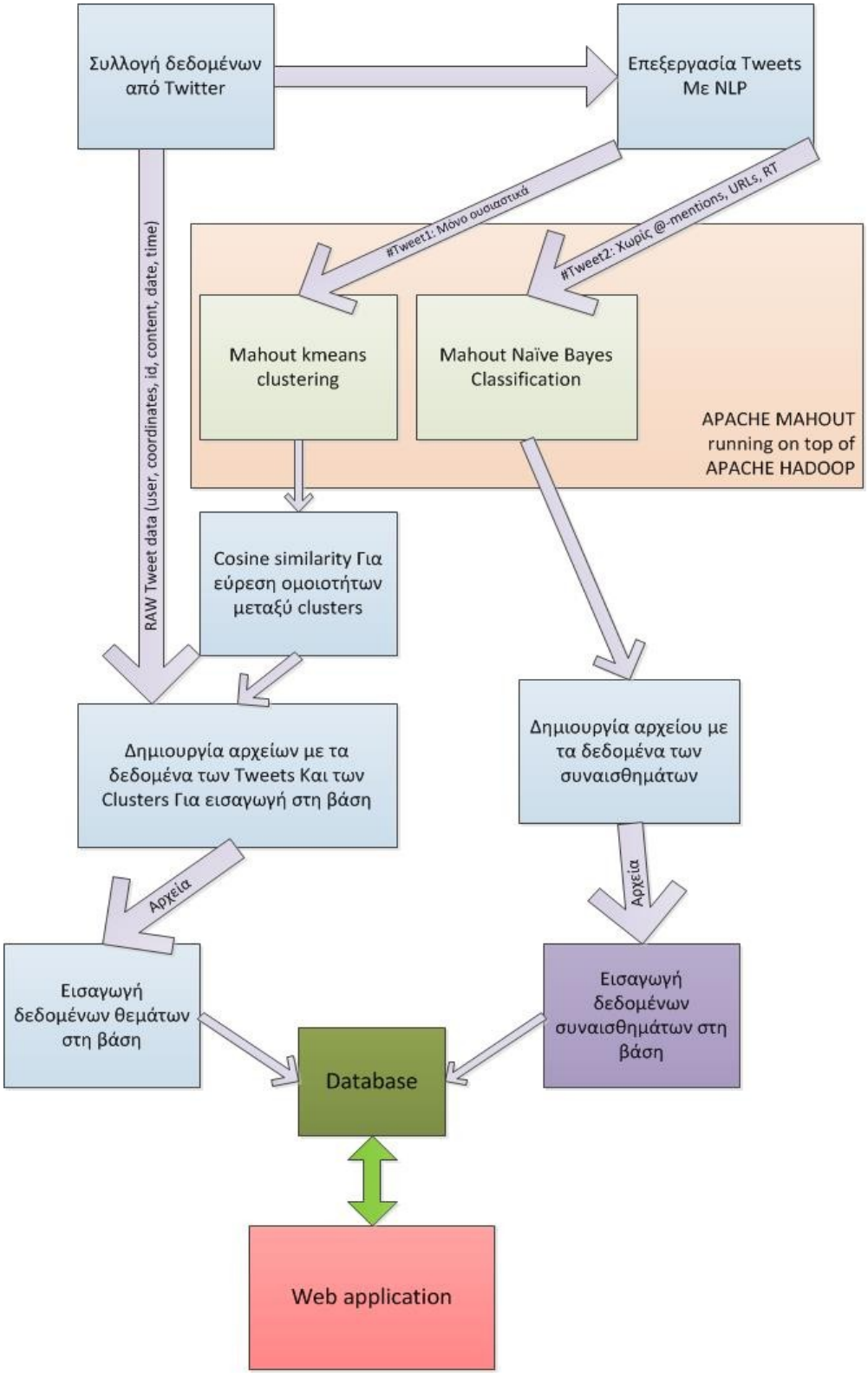
8. Επεξεργασία των αρχικών tweets και δημιουργία αρχείου .csv με τις πληροφορίες του καθενός

9. Δημιουργία αρχείου .txt με Cypher Queries που εισάγουν τις σχέσεις μεταξύ των clusters στη βάση

10. Εισαγωγή των δεδομένων στη βάση neo4j (neo4j.com)

11. Δημιουργία web εφαρμογής (javascript, ajax, html, css) που επικοινωνεί με τη βάση κάνοντας REST CALLS. Ύστερα παρουσιάζει τα αποτελέσματα στο χρήστη με κατανοητό τρόπο

Σχηματικά:



4.2 Περιγραφή Λειτουργιών

4.2.1 Συλλογή δεδομένων από το Twitter

Αναπτύχθηκε εφαρμογή σε Java που χρησιμοποιώντας βιβλιοθήκες του Twitter4j.org επικοινωνεί με το API του Twitter και ψάχνει για tweets με συγκεκριμένα χαρακτηριστικά. Συγκεκριμένα ψάχνει για tweets από Λονδίνο, Νέα Υόρκη και Los Angeles. Τα tweets αυτά πρέπει να είναι γραμμένα στην αγγλική γλώσσα, και να έχουν δημοσιευτεί συγκεκριμένες μέρες. Η ίδια εφαρμογή αποθηκεύει αυτά τα tweets ένα-ένα σαν αρχεία .txt σε φακέλους ανά πόλη και ανά μέρα. Σε κάθε αρχείο κρατιέται η πληροφορία για το χρήστη που έκανε τη δημοσίευση, την ημερομηνία και την ώρα, τις γεωγραφικές συντεταγμένες (όπου αυτές υπάρχουν) από τις οποίες έγινε η δημοσίευση και βεβαίως το μοναδικό status ID του tweet όπως αυτό λαμβάνεται από το Twitter.

Ενδεικτική μορφή ενός tweet:

filename: 609511068499492865.txt (όπου 609511068499492865 είναι το status ID)

Περιεχόμενο αρχείου:

```
Sat Jun 13 03:02:17 EEST 2015
33.8371374,-118.0268801
iitsme_t3rd
@villaford @mythicalmonique @JAEGARCIA99 @iAmDJ_DUCK @frazgo
good looking out
```

4.2.2 Επεξεργασία κειμένου

Μετά τη συλλογή των tweets έπρεπε να γίνει ο καθαρισμός των "άχρηστων" όρων ώστε να γίνει η σημασιολογική και η συναισθηματική επεξεργασία τους. Έπρεπε δηλαδή στο αρχικό κείμενο του tweet να αναγνωριστεί τι μέρος του λόγου είναι η κάθε λέξη και να αφαιρεθούν όσες δεν χρειάζονταν.

Για τη σημασιολογική ανάλυση κρατήθηκαν μόνο όσες λέξεις ήταν ουσιαστικά ή #hashtags. Για τη συναισθηματική ανάλυση κρατήθηκαν όλες οι λέξεις εκτός από τα @-mentions (οι αναφορές σε χρήστες) και οι σύνδεσμοι URL.

Στο τέλος από ένα αρχείο tweet παίρνουμε 2 καινούργια:

1. Ένα που περιέχει μόνο ουσιαστικά και #hashtags που θα χρησιμοποιηθεί για τη σημασιολογική ανάλυση
2. Ένα που περιέχει τα πάντα εκτός από at-mentions και URLs που θα χρησιμοποιηθεί για συναισθηματική ανάλυση

Παράδειγμα αρχικού tweet:

```
Mon Jun 08 03:00:06 EEST 2015
40.68888889,-73.94444444
for_thelovers
#BOSS roddy_rodd - I told you I got you brother
#ftlphotography #fortheloversmusic @miss_luna we made…
https://t.co/0sAef1dKSs
```

Tweet (μορφή 1) ύστερα από επεξεργασία για να μείνουν μόνο ουσιαστικά και #hashtags:

```
#BOSS roddy_rodd brother #ftlphotography #fortheloversmusic
```

Tweet (μορφή 2) ύστερα από επεξεργασία για να αφαιρεθούν at-mentions και URLs:

```
BOSS roddy_rodd - I told you I got you brother ftlphotography
fortheloversmusic we made …
```

4.2.3 Ομαδοποίηση (clustering) των tweets της μορφής 1

Όλα τα tweets της μορφής 1 υφίστανται επεξεργασία ανά πόλη και ανά μέρα και ομαδοποιούνται με χρήση του αλγορίθμου k-means. Για κάθε μέρα σε κάθε πόλη βγαίνουν k ομάδες(clusters) από tweets και για κάθε ομάδα βγαίνουν οι 30 πιο σημαντικοί-αντιπροσωπευτικοί όροι και το βάρος του καθενός. Παράδειγμα ενός cluster που έχει συνδεδεμένα πάνω του n=1303 tweets:

```
:VL-9726{n=1303 c=[2k:0.022, 2k
Top Terms:
work => 0.3697761468309123
things => 0.3210664538356403
guys => 0.2796880036981309
fuck => 0.2263714914036456
beer => 0.19506051699564445
nigga => 0.18944655979768368
house => 0.17241025099091958
bae => 0.1713769437712335
room => 0.15710620704469366
days => 0.15689439839431166
song => 0.1555156663849275
video => 0.14323641937326123
name => 0.1387571494394509
jets => 0.13596446717233723
season => 0.12933849721163487
team => 0.12885071914743115
bruh => 0.12485830946693216
halloween => 0.1239503478418381
hall => 0.12226382860840967
anyone => 0.12152380075989004
webster => 0.12052382676306818
bed => 0.11079365714915605
pic => 0.10770311340952687
head => 0.10486451278534287
hell => 0.10133876859089638
selfie => 0.1009361431036925
dreams => 0.09851018374275447
fans => 0.0933051003187506
bitches => 0.0914363897679315
menu => 0.09142014217303518
```

4.2.4 Ταξινόμηση (classification) των tweets της μορφής 2

Όλα τα tweets της μορφής 2 ταξινομούνται ανάλογα με το συναίσθημά τους σε θετικά ή αρνητικά χρησιμοποιώντας τον αλγόριθμο Complement Naive Bayes. Για κάθε tweet παίρνουμε δύο νούμερα. Το ένα περιγράφει το πόσο θετικό και το άλλο το πόσο αρνητικό (δείχνει να) είναι.

4.2.5 Επεξεργασία των αποτελεσμάτων του clustering

Η κάθε ομάδα από tweets που λαμβάνεται από την ομαδοποίηση για μια πόλη συγκρίνεται με όλες τις άλλες ομάδες της ίδιας πόλης αλλά διαφορετικών ημερών. Αναζητείται ο βαθμός ομοιότητας της ομάδας αυτής με τις ομάδες από τις υπόλοιπες μέρες. Για κάθε ομάδα παίρνουμε 6 άλλες ομάδες (μία για καθεμιά από τις υπόλοιπες μέρες της εβδομάδας) με τις οποίες αυτή μοιάζει πιο πολύ. Θεωρείται ότι μια ομάδα είναι σαν ένα θέμα συζήτησης. Βλέπουμε έτσι την αλλαγή του θέματος αυτού στο χρόνο παρατηρώντας τις διαφορές μεταξύ των όρων-λέξεων καθεμιάς από τις 7 ομάδες στις οποίες έχουμε εστιάσει. Τα αποτελέσματα εξάγονται σε αρχείο csv. Παράδειγμα 7 συσχετισμένων clusters από Δευτέρα έως Κυριακή:

nr. 13215 on 20150608	nr. 268 on 20150609	nr. 11512 on 20150610	nr. 6666 on 20150611	nr. 11036 on 20150612	nr. 2949 on 20150613	nr. 8551 on 20150614
show tonight truman degree lcfba15 brewery notes scala day gendered somenews feminism gmt goldsmiths catwalk	show anniversary ggv lizquen guests pag tony beartooth records banquet truman illness gig goldsmiths london	show tickets ep launch makeup makeupartist bugatti catwalk fashion trent ghanapartyinthepark anniversary degree thelizmartins headline	show governments tickets fools price books evening affordable plus heath enough rest art tv hampstead	show music studio tickets storyteller secrets copy trunk cherrybomb bath photoshoot detail shift gig ladies	lcm ss16 office show londoncollectionsmen menswear day2 lcmss16 fmlondon ymc london fashion berthold_uk finale mensfashion	show lcm belstaff coach desertexplorers londoncollections fashion lcmss16 portrait davidgandy ki thetimes teklife britishfashioncouncil grooming

4.2.6 Επεξεργασία των αποτελεσμάτων του clustering και δημιουργία αρχείου με τα δεδομένα των ομάδων

Τα αποτελέσματα του clustering βρίσκονται σε αρχεία τα οποία διαβάζονται από μια εφαρμογή που αναπτύχθηκε σε java και εξάγονται συνοπτικά σε ένα αρχείο txt. Στο αρχείο αυτό οι πληροφορίες για τους clusters εμπεριέχονται σε Cypher Queries ώστε να είναι έτοιμα για εισαγωγή στη βάση . Παράδειγμα γραμμής του αρχείου txt:

```
MERGE (c:Cluster {id:10024,date:20150608,city:"London",terms:
["sa","crib","sessions","moments","westwood's","tim","tomorrow","hop","jump","hip","histor
y"],weights:
[5.159849754654535,5.154936075210571,5.154936075210571,5.151454448699951,4.60723
5899065981,4.584874292411427,4.3806842459310396,0.043981934537982,0.04055051048
20327,0.0388848073411696,0.03394159468093721]})
```

4.2.7 Επεξεργασία των αποτελεσμάτων του classification και δημιουργία αρχείου με τα δεδομένα των συναισθημάτων

Τα αποτελέσματα του classification βρίσκονται σε αρχεία τα οποία διαβάζονται από μια εφαρμογή που αναπτύχθηκε σε java και εξάγονται συνοπτικά σε ένα αρχείο csv. Το αρχείο σε κάθε γραμμή περιέχει το μοναδικό status-ID του tweet και άλλους δύο αριθμούς. Ο ένας δείχνει το πόσο αρνητικό και ο άλλος το πόσο θετικό είναι. Παράδειγμα γραμμής του αρχείου csv:

```
"521450270606503936","373.73354904980806","399.6912322611348"
```

4.2.8 Επεξεργασία των αρχικών tweets και δημιουργία αρχείου με τις πληροφορίες του καθενός

Από την ίδια εφαρμογή διαβάζονται τα αρχεία των tweets ένα-ένα και εξάγονται οι πληροφορίες τους (ημερομηνία, ώρα, χρήστης, συντεταγμένες, περιεχόμενο,status-ID) σε αρχείο csv. Το αρχείο σε κάθε γραμμή περιέχει τις πληροφορίες για ένα tweet. Παράδειγμα γραμμής:

```
"521450276444966913","20141013","1030008","40.75150004","-
73.99878471","Snappleyard","Let\'s talk about Maggie wearing
Daryl\'s poncho","0.7857568303063941"
```

Με τη σειρά οι πλοροφορίες είναι:"ID","Date","Time(αφαιρώντας το '1' από την αρχή της συβολοσειράς),"latitude","longitude","χρήστης","περιεχόμενο","απόσταση από το κεντρικό διάνυσμα του cluster"

4.2.9 Δημιουργία αρχείου με τις σχέσεις μεταξύ των clusters

Τα αποτελέσματα του 4.2.5 γράφονται σε ένα αρχείο csv με την εξής μορφή: `"20141017","10284","20141013","7666","0.19405981940682088"` που σημαίνει ότι ο cluster 10284 της 17ης Οκτωβρίου 2014 μοιάζει με τον 7666 της 13ης Οκτωβρίου 2014 και το συνημίτονο της γωνίας μεταξύ των διανυσμάτων τους είναι 0.19405981940682088

4.2.10 Εισαγωγή των δεδομένων στη βάση

Εισαγωγή από web interface:

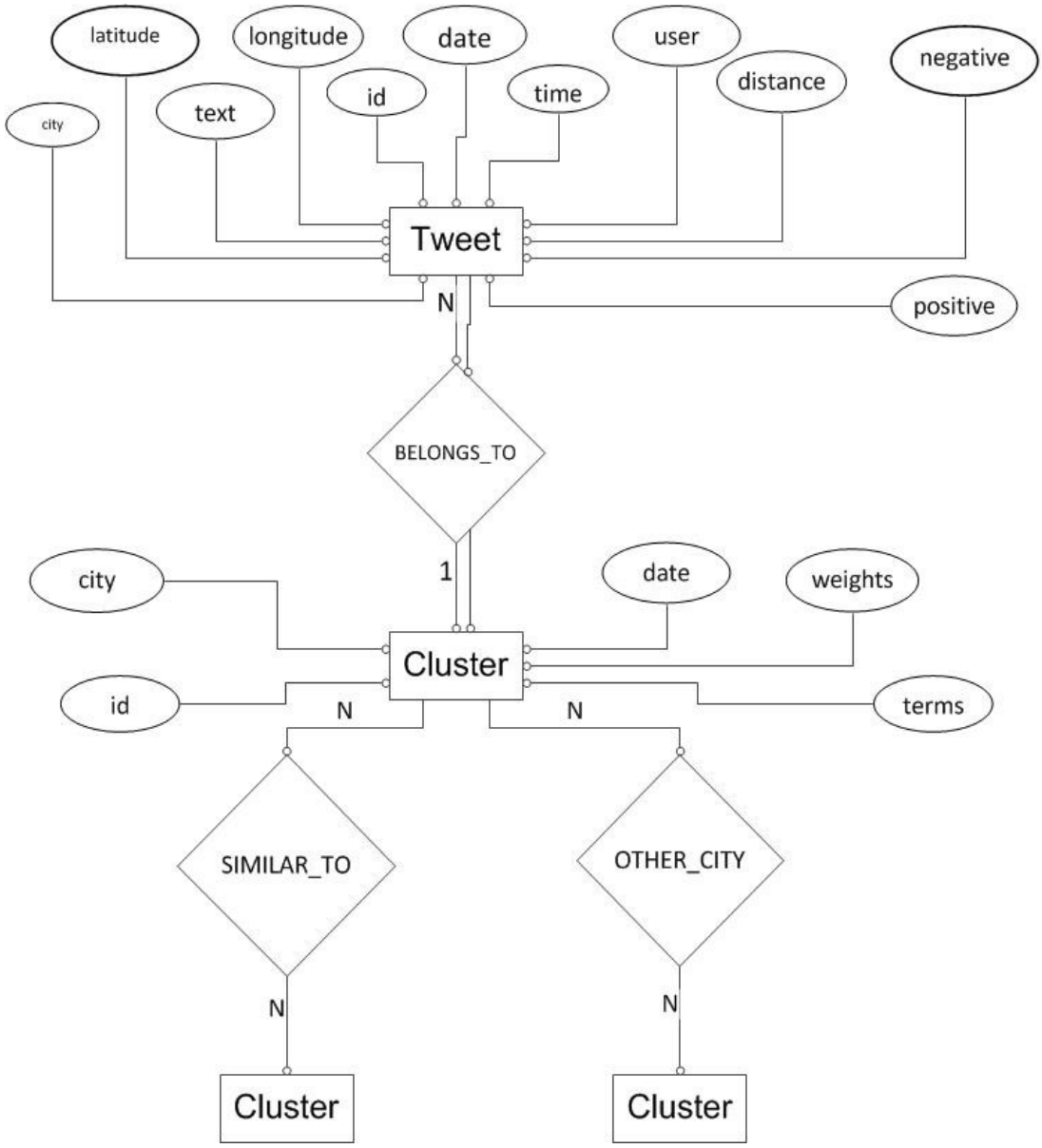
Η neo4j προσφέρει web interface από όπου μπορούν να εκτελεστούν όλα τα cypher queries και να εισαχθούν στη βάση οι πληροφορίες που περιέχονται στα αρχεία csv.

Εισαγωγή με java:

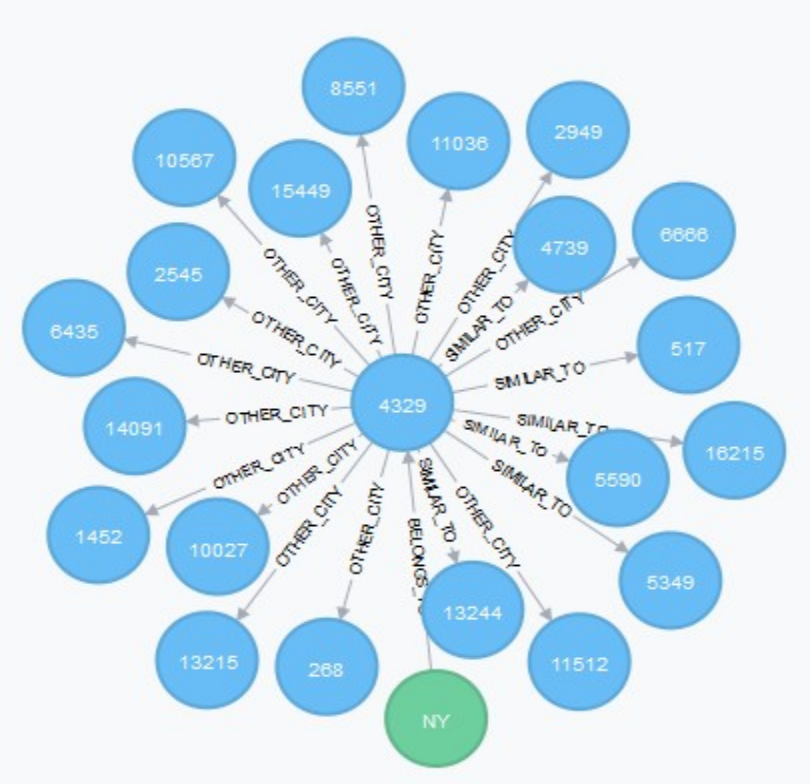
Προσφέρονται επίσης βιβλιοθήκες με τις οποίες μια εφαρμογή java μπορεί να επικοινωνήσει με τη βάση και να εκτελέσει cypher queries.

4.3 Μοντέλο Βάσης Δεδομένων

Σχεδιάστηκε το ER-διάγραμμα της βάσης και παρουσιάζεται στην ακόλουθη εικόνα:



Η βάση τελικά έχει αυτή τη μορφή:



Όπως φαίνεται στην παραπάνω εικόνα, το πράσινο tweet συνδέεται με σχέση BELONGS_TO με τον cluster 4329 της Νέας Υόρκης. Ο cluster αυτός έχει και άλλα tweets (τα οποία δεν φαίνονται στο σχήμα) συνδεδεμένα πάνω του. Ο cluster 4329 συνδέεται με σχέσεις SIMILAR_TO με τους 4739, 517, 5590, 16215, 5349, 13244 της Νέας Υόρκης. Οι clusters αυτοί είναι από τις υπόλοιπες 6 μέρες τις εβδομάδας, εκτός της μέρας του 4329. Ο 4329 συνδέεται επίσης με σχέσεις OTHER_CITY με clusters άλλων πόλεων με τους οποίους παρουσιάζει ομοιότητα.

5 Υλοποίηση

Στη συνέχεια περιγράφουμε λεπτομερώς θέματα υλοποίησης του συστήματος.

5.1 Λεπτομέρειες υλοποίησης

5.1.1 Συλλογή δεδομένων από το Twitter

Τρόπος συλλογής Tweets: Twitter API - Search Queries

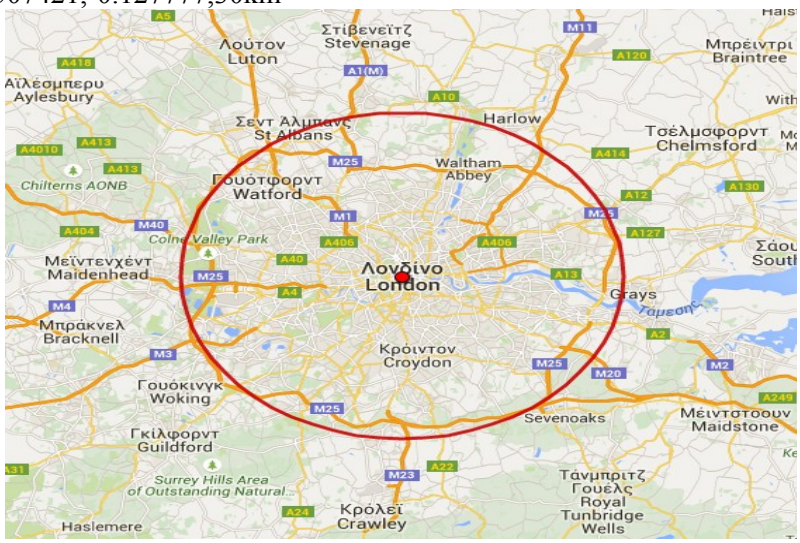
Χρησιμοποιήθηκε κώδικας από τη σελίδα Twitter4j.org. Γράφτηκε πρόγραμμα σε java που επικοινωνεί με το twitter API στέλνοντας ένα search query για statuses και λαμβάνοντας ως απάντηση τα statuses-tweets που ικανοποιούν τις συνθήκες του query. Συγκεκριμένα τα queries που χρησιμοποιήθηκαν ήταν της μορφής:

```
Query query = new Query("geocode:34.042763,-118.266936,10km");
query.since("2015-06-14");
query.lang("en");
query.until("2015-06-15");
```

Με το query αυτό δηλώνεται ότι ζητάμε tweets προερχόμενα από τη γεωγραφική περιοχή που περιέχεται στον κύκλο με κέντρο το 34.042763,-118.266936 και ακτίνα 10km. Επιπλέον, ζητάμε τα tweets αυτά να έχουν δημοσιευτεί σε ημερομηνίες που ξεκινούν από 2015-06-14 και τελειώνουν πριν τις 00:00 της 2015-06-15. Σαν responses λαμβάνουμε ένα μεγάλο αριθμό tweets. Επειδή το Twitter επιτρέπει τη λήψη μέχρι 18000 απαντήσεων σε διάστημα 15 λεπτών μέσω του API του, οι περιοχές ενδιαφέροντος χωρίστηκαν σε μικρότερες υποπεριοχές (μικρότερους κύκλους) ώστε να καλυφθούν ολόκληρες από διαδοχικά search queries. Στην εργασία αυτή ελήφθησαν tweets από τις πόλεις του Λονδίνου, της Νέας Υόρκης και του Los Angeles.

Οι περιοχές που ζητήθηκαν είναι οι ακόλουθες:

Λονδίνο: 51.507421,-0.127777,30km



Νέα Υόρκη:

staten island: 40.596972,-74.125164,6km

bronx: 40.850529,-73.877113,6km

lower manhattan: 40.713843,-73.999216,2km

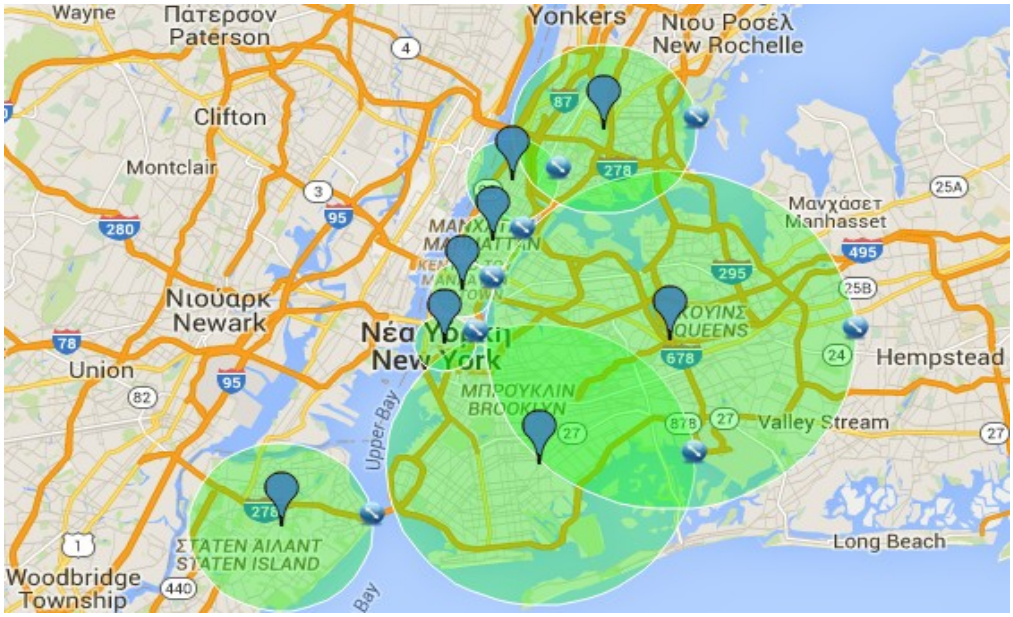
empire state building manhattan: 40.748531,-73.985750,2km

central park manhattan: 40.779570,-73.963352,2km

harlem manhattan: 40.817925,-73.947769,3km

brooklyn: 40.636520,-73.926380,10km

queens+jfk: 40.716032,-73.826216,12km

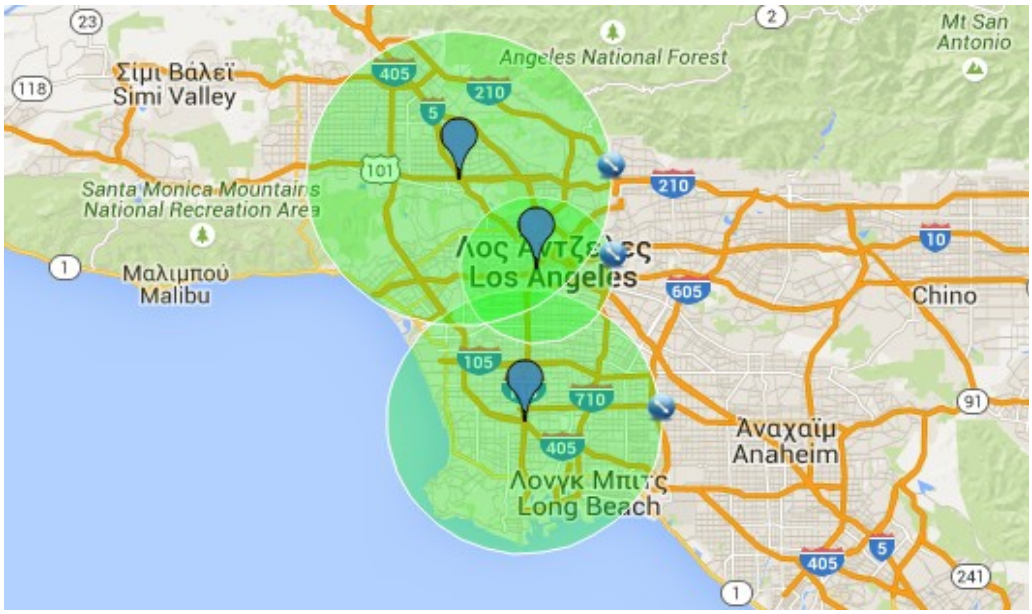


Los Angeles:

34.042763,-118.266936,10km

34.153849,-118.377231,20km

33.856717,-118.284877,18km



Πρώτη επεξεργασία και αποθήκευση των tweets

Αμέσως μετά τη λήψη του μέσω του Twitter API, κάθε tweet ελέγχεται για το αν περιέχει τις ακριβείς συντεταγμένες από τις οποίες δημοσιεύτηκε. Αυτό συμβαίνει γιατί χρειαζόμαστε την ακριβή τοποθεσία για να επεξεργαστούμε τη χωρική πληροφορία των tweets που συλλέγουμε. Αν κάποιο tweet δεν έχει συντεταγμένες, δίνεται αυτόματα το 0,0. Κάθε tweet αποθηκεύεται σε ξεχωριστό αρχείο .txt. Το κάθε αρχείο έχει τη μορφή:

date

coordinates

sender

content

και λαμβάνει σαν όνομα το status id που έχει από το Twitter.

Στο τέλος αυτής της διαδικασίας έχουμε συλλέξει περί τα 17000 tweets ανά μέρα και ανά πόλη για ένα διάστημα 7 ημερών (Δευτέρα-Κυριακή).

Η εβδομάδα που επιλέχθηκε είναι από 8 Ιουνίου 2015 έως 14 Ιουνίου 2015 και μαζεύτηκαν 334972 tweets συνολικά.

5.1.2 Επεξεργασία κειμένου

Όπως αναφέρθηκε παραπάνω, μετά τη συλλογή των tweets έπρεπε να γίνει ο καθαρισμός των "άχρηστων" όρων ώστε να γίνει η σημασιολογική και η συναισθηματική επεξεργασία τους. Έπρεπε δηλαδή στο αρχικό κείμενο του tweet να αναγνωριστεί τι μέρος του λόγου είναι η κάθε λέξη και να αφαιρεθούν όσες δεν χρειάζονταν.

Για τη σημασιολογική ανάλυση κρατήθηκαν μόνο όσες λέξεις ήταν ουσιαστικά ή #hashtags. Για τη συναισθηματική ανάλυση κρατήθηκαν όλες οι λέξεις εκτός από τα @-mentions (οι αναφορές σε χρήστες) και οι σύνδεσμοι URL.

ArkTweet NLP: Για να γίνει η κατηγοριοποίηση των λέξεων κάθε tweet και να επιλεγούν αυτές που κρατήθηκαν, χρησιμοποιήθηκε κώδικας για Επεξεργασία Φυσικής Γλώσσας από την ομάδα Noah's Ark του Language Technologies Institute, School of Computer Science του πανεπιστημίου Carnegie Mellon (http://www.ark.cs.cmu.edu/TweetNLP/). Κάθε λέξη ενός tweet κατατάσσεται σε μία από τις κατηγορίες που φαίνονται στην ακόλουθη εικόνα (POS tagset from Gimpel et al. 2011):

A Part-of-Speech Tagset

N	common noun
O	pronoun (personal/WH; not possessive)
^	proper noun
S	nominal + possessive
Z	proper noun + possessive
V	verb including copula, auxiliaries
L	nominal + verbal (e.g. *i'm*), verbal + nominal (*let's*)
M	proper noun + verbal
A	adjective
R	adverb
!	interjection
D	determiner
P	pre- or postposition, or subordinating conjunction
&	coordinating conjunction
T	verb particle
X	existential *there*, predeterminers
Y	X + verbal
#	hashtag (indicates topic/category for tweet)
@	at-mention (indicates a user as a recipient of a tweet)
~	discourse marker, indications of continuation across multiple tweets
U	URL or email address
E	emoticon
$	numeral
,	punctuation
G	other abbreviations, foreign words, possessive endings, symbols, garbage

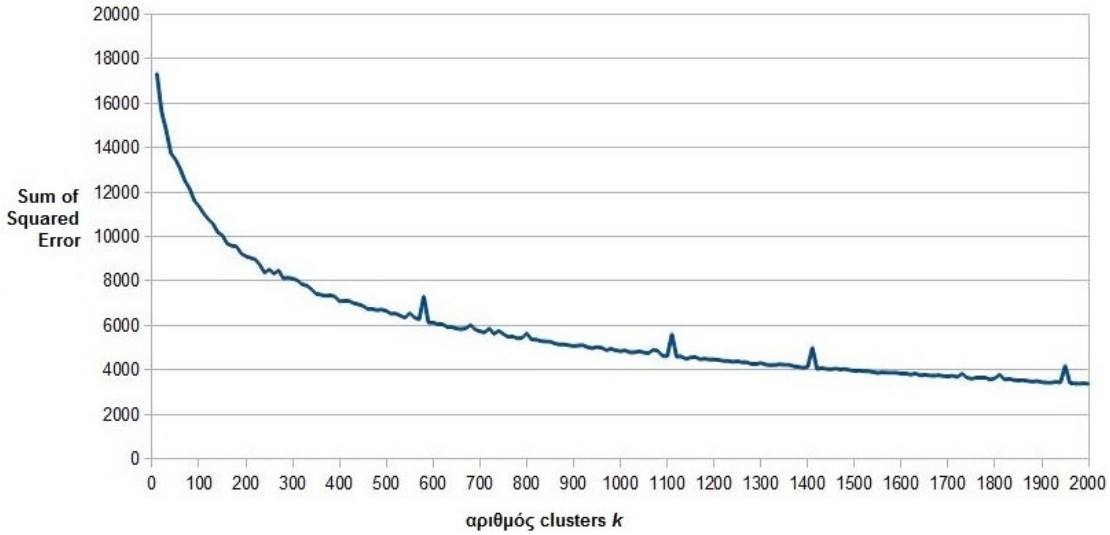
Αφού γίνει η αναγνώριση του τι μέρος του λόγου είναι η κάθε λέξη, αφαιρούμε όσες δεν χρειάζονται και τελικά από κάθε tweet παίρνουμε 2 καινούργια:

1. Ένα που περιέχει μόνο ουσιαστικά και #hashtags που θα χρησιμοποιηθεί για τη σημασιολογική ανάλυση
2. Ένα που περιέχει τα πάντα εκτός από @-mentions και URLs που θα χρησιμοποιηθεί για συναισθηματική ανάλυση

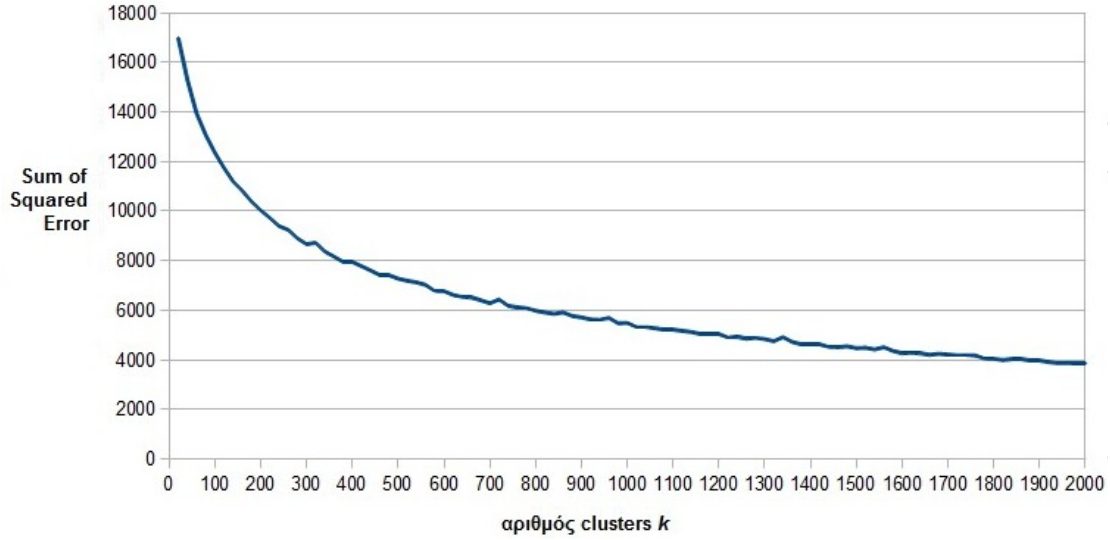
5.1.3 Υπολογισμός του αριθμού k των clusters

Για κάθε μέρα εξετάστηκε το άθροισμα τετραγώνων των αποκλίσεων και έγινε το διάγραμμα SSE (k). Η μορφή που είχαν οι γραφικές παραστάσεις ήταν όπως οι ακόλουθες:

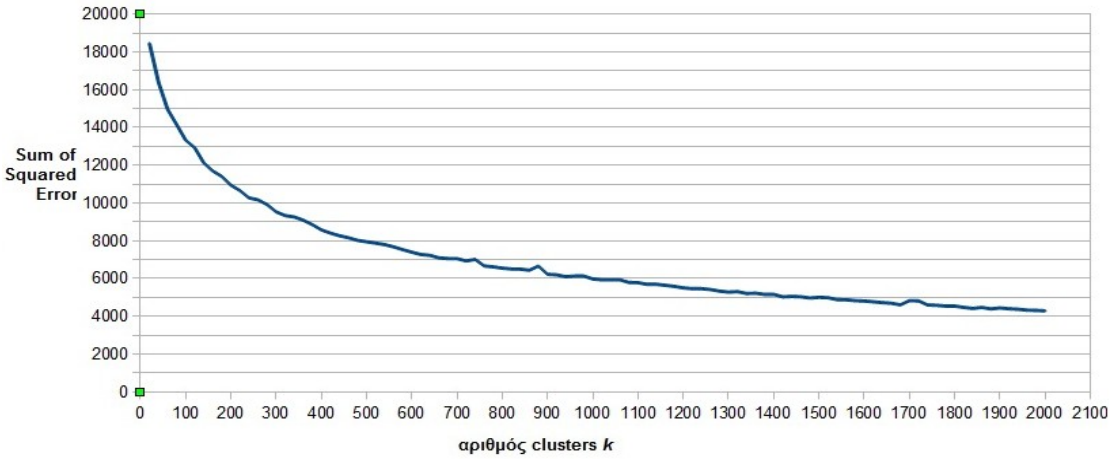
Νέα Υόρκη - Δευτέρα



Νέα Υόρκη – Τρίτη



Νέα Υόρκη – Τετάρτη



Παρατηρείται στις γραφικές παραστάσεις ότι για μικρά k, το άθροισμα των τετραγώνων των αποκλίσεων μειώνεται πιο γρήγορα. Όσο το k μεγαλώνει, ο ρυθμός μείωσης του SSE δεν είναι τόσο γρήγορος. Επιλέγεται το k=200, καθώς θεωρείται ότι για k>200 ο ρυθμός μείωσης του SSE είναι ικανοποιητικά αργός. Με άλλα λόγια δεχόμαστε ότι για k>200, η αύξηση του k δεν επιφέρει μεγάλες αλλαγές στα αποτελέσματα που παίρνουμε από τον kmeans. Σημειώνεται ότι κατά μέσο όρο έχουμε συλλέξει 17000 tweets ανά μέρα άρα αντιστοιχούν 85 tweets/cluster.

5.1.4 Mahout kmeans clustering

Εισαγωγή δεδομένων

Γίνονται Upload σε VM στον okeanos της ΕΔΕΤ τα επεξεργασμένα tweets όπου έχουν απομείνει μόνο τα ουσιαστικά. Τα tweets είναι κατανεμημένα σε φακέλους ανάλογα με τη μέρα που δημιουργήθηκαν(Δευτέρα, Τρίτη ..... Κυριακή).

Εκτέλεση script cluster-tweets.sh

Ύστερα τρέχει το script cluster-tweets.sh που κάνει το clustering. Το script εκτελεί τις ακόλουθες εντολές:

$HADOOP dfs -put ${WORK_DIR}/tweets-out/monday ${WORK_DIR}/tweets-out

Με την εντολή αυτή αντιγράφονται από το Local File System στο Hadoop Distributed File System τα tweets της Δευτέρας.

$MAHOUT seqdirectory -i ${WORK_DIR}/tweets-out -o ${WORK_DIR}/tweets-out-seqdir -c UTF-8 -chunk 64 -xm sequential

Για να γίνει η επεξεργασία, πρέπει οι πληροφορίες που περιέχουν τα tweets (τα οποία είναι αρχεία κειμένου) να αναπαρασταθούν με διανύσματα. Το πρώτο βήμα για να γίνει αυτό είναι τα αρχεία κειμένου να μετατραπούν σε Sequence Files. Ένα SequenceFile είναι μια κλάση του hadoop που μας επιτρέπει να γράψουμε αυθαίρετα ζεύγη από (κλειδί,τιμή) σε αυτήν. Για να μετατραπεί ύστερα το αρχείο-tweet σε διάνυσμα πρέπει το κλειδί να είναι ένα μοναδικό αναγνωριστικό (docID) του αρχείου, και η τιμή να είναι το περιεχόμενο του αρχείου σε κωδικοποίηση UTF-8.

**$MAHOUT seq2sparse **

**-i ${WORK_DIR}/tweets-out-seqdir/ **

-o ${WORK_DIR}/tweets-out-seqdir-sparse-kmeans --maxDFPercent 100 --namedVector

Η εντολή αυτή παίρνει τα SequenceFiles που δημιουργήθηκαν από το seqdirectory και δημιουργεί νέα SequenceFiles τα οποία έχουν τη μορφή (docID, TF-IDF Vector). Περιέχουν δηλαδή ένα αναγνωριστικό docID του αρχείου-tweet του οποίου την πληροφορία αναπαριστούν και ένα διάνυσμα TF-IDF των λέξεων του tweet.

Η seq2sparse δημιουργεί επίσης ένα dictionary Sequencefile που περιέχει όλους τους όρους που περιέχονται στη συλλογή των αρχείων μας και δίπλα ένα μοναδικό αριθμό που προσδιορίζει τον όρο αυτό: (wordIndex, word).

Η μορφή ενός διανύσματος TF-IDF είναι όπως η ακόλουθη: /522537387646537728.txt:{8290:7.1680192947387695}.Εδώ το αρχείο-tweet 522537387646537728.txt περιέχει τη λέξη με wordIndex 8290 η οποία έχει TFIDF=7.1680192947387695 για αυτό το αρχείο. Το 8290 μπορεί να αντιστοιχιστεί σε λέξη βρίσκοντας το ζευγάρι (wordIndex, word) στο dictionary file με wordIndex=8290.

Το maxDFPercent είναι ένα άνω όριο που προσδιορίζει το επιτρεπτό ποσοστό εμφανίσεων ενός όρου στα αρχεία που εξετάζουμε ώστε να μην αγνοηθεί. Δηλαδή αν κάποιος όρος εμφανίζεται σε πάνω από 90% των αρχείων μπορεί να θεωρηθεί ως μη χρήσιμος για να αντιπροσωπεύσει κάποιο αρχείο αφού εμφανίζεται σχεδόν σε όλα. Συνήθως αυτό το όριο χρησιμοποιείται για να αφαιρεθούν stop-words όπως "and, has" κ.ά. που ενώ εμφανίζονται συχνά δεν περιέχουν σημαντική πληροφορία που να διαφοροποιεί ένα αρχείο από κάποιο άλλο. Στην περίπτωσή μας επελέγη ως ανώτατο όριο το 100% διότι έχουμε επεξεργαστεί τα κείμενα των tweets σε προηγούμενο βήμα και δεν περιέχουν άχρηστες λέξεις.

**$MAHOUT kmeans **

**-i ${WORK_DIR}/tweets-out-seqdir-sparse-kmeans/tfidf-vectors/ **

**-c ${WORK_DIR}/tweets-kmeans-clusters **

**-o ${WORK_DIR}/tweets-kmeans **

**-dm org.apache.mahout.common.distance.CosineDistanceMeasure **

-x 20 -k 2000 -ow --clustering

Η εντολή που εφαρμόζει τον αλγόριθμο kmeans στα SequenceFiles που δημιουργήθηκαν από την seq2sparse. Παίρνει σαν είσοδο τα tfidf vectors και υπολογίζει τα κέντρα των clusters και κατανέμει τα αρχεία σε αυτούς.Οι επιλογές που έγιναν με αυτή την εντολή είναι:

-dm org.apache.mahout.common.distance.CosineDistanceMeasure: Επελέγη το CosineDistanceMeasure σαν τρόπος μέτρησης της απόστασης κάθε διανύσματος από το διάνυσμα που αντιπροσωπεύει το κέντρο του cluster. CosineDistanceMeasure = Cosine Similarity - 1.

-x 20: Ο ανώτατος επιτρεπτός αριθμός επανυπολογισμών των κέντρων των clusters μέχρι να συγκλίνουν. Από τα αποτελέσματα της επεξεργασίας και ύστερα από δοκιμές με μικρούς και μεγάλους αριθμούς από clusters διαπιστώθηκε ότι πάντα η διαδικασία συνέκλινε σε λιγότερες από 20 επαναλήψεις. Έτσι αφέθηκε το 20 ως ανώτατο όριο.

-k K: Ο αριθμός των clusters που επελέγησαν να υπολογιστούν για τη συγκεκριμένη μέρα.

Μετά το πέρας εκτέλεσης της εντολής kmeans θα έχουμε στο φάκελο /tweets-kmeans/clusters-*-final ένα αρχείο με τους clusters που υπολόγισε και στο φάκελο /tweets-kmeans/clusteredPoints ένα αρχείο που λέει ποιο διάνυσμα κατανεμήθηκε σε ποιο cluster.

**$MAHOUT clusterdump **

**-i `hadoop dfs -ls -d ${WORK_DIR}/tweets-kmeans/clusters-*-final | awk '{print $8}'` **

**-o ${WORK_DIR}/tweets-kmeans/clusterdumpNYsunday **

**-d ${WORK_DIR}/tweets-out-seqdir-sparse-kmeans/dictionary.file-0 **

**-dt sequencefile -b 30 -n 30 --evaluate -dm org.apache.mahout.common.distance.CosineDistanceMeasure -sp 0 **

--pointsDir ${WORK_DIR}/tweets-kmeans/clusteredPoints

Η εντολή clusterdump παίρνει σαν είσοδο:

1. τους clusters που υπολόγισε ο kmeans και έχουν αποθηκευτεί στο φάκελο clusters-*-final
2. ένα αρχείο που δείχνει τη σύνδεση του κάθε διανύσματος που δώσαμε ως είσοδο στον Kmeans με τον cluster στον οποίο κατανεμήθηκε (δηλαδή από τον οποίο απέχει τη μικρότερη απόσταση). Το αρχείο αυτό είναι αποθηκευμένο στο φάκελο /clusteredPoints
3. Το dictionary file που δημιούργησε η seq2sparse και αντιστοιχίζει κάθε wordIndex με τη λέξη την οποία αντιπροσωπεύει.

Οι επιλογές που έγιναν με αυτή την εντολή είναι:

-n 30: Επελέγησαν να επιστρέφονται οι 30 πιο σημαντικοί όροι κάθε cluster.

-dm org.apache.mahout.common.distance.CosineDistanceMeasure: Επελέγη το CosineDistanceMeasure σαν τρόπος μέτρησης της απόστασης κάθε διανύσματος από το διάνυσμα που αντιπροσωπεύει το κέντρο του cluster. CosineDistanceMeasure = Cosine Similarity - 1.

Η εντολή clusterdump δίνει σαν έξοδο ένα αρχείο όπου παρουσιάζονται οι clusters και οι πιο αντιπροσωπευτικοί-σημαντικοί όροι σε καθέναν.

Π.χ:

:VL-0{n=137 c=[a2b:0.212, actor

Top Terms:

film => 0.27162249070884537

affair => 0.2666899966497491

.....

Αυτό σημαίνει ότι ο cluster 0 έχει συνδεδεμένα σε αυτόν 137 tweets και η πιο αντιπροσωπευτική λέξη είναι film. Άρα μπορούμε να συμπεράνουμε ότι τα περισσότερα από αυτά τα 137 tweets έχουν ως θέμα τους ταινίες.

Συλλογή αποτελεσμάτων

Μετά την εκτέλεση του kmeans για κάθε μέρα παίρνουμε με την εντολή clusterdump το αρχείο clusterdump για τη συγκεκριμένη μέρα. Στη συνέχεια εκτελείται η εντολή hadoop fs -text /tmp/mahout-work-hduser/tweets-kmeans/clusteredPoints/part-m-00000 > clusteredpoints.txt και παίρνουμε για τη συγκεκριμένη μέρα το αρχείο clusteredpoints.txt που περιέχει την πληροφορία για το ποιο διάνυσμα κατανεμήθηκε σε ποιο cluster.

5.1.5 Mahout Naive Bayes classification

Συλλογή training set

Χρειάστηκε να συλλεγεί ένα σύνολο από θετικές και αρνητικές φράσεις/tweets ώστε να γίνει η εκπαίδευση του αλγορίθμου και να δημιουργηθεί το μοντέλο με βάση το οποίο θα ταξινομηθούν τα tweets.Αντλήθηκαν φράσεις από:

https://github.com/jperla/sentiment-data/tree/master/pang_lee_movie_reviews/rt-polaritydata

https://github.com/malavbhavsar/sentimentalizer/tree/master/lib/data

Κάθε φράση από αυτές τις σελίδες αποθηκεύτηκε σε ξεχωριστό αρχείο και αποθηκεύτηκε στο φάκελο positive, αν ανήκε στα positive, ή στο φάκελο negative.

Ύστερα οι δυο αυτοί φάκελοι έγιναν upload στο VM του okeanos.

Εκπαίδευση (Training)

Ακολούθως εκτελέστηκε το script train-classify-sentiment.sh

Με το script εκτελέστηκαν οι ακόλουθες εντολές:

$HADOOP dfs -put ${WORK_DIR}/sentiment-all ${WORK_DIR}/sentiment-all

Με την εντολή αυτή αντιγράφονται από το Local File System στο Hadoop Distributed File System τα tweets του training set.

**./bin/mahout seqdirectory **

**-i ${WORK_DIR}/sentiment-all **

-o ${WORK_DIR}/sentiment-seq -ow

Δημιουργούνται τα SequenceFiles. Για να γίνει η επεξεργασία, πρέπει οι πληροφορίες που περιέχουν τα αρχεία κειμένου να αναπαρασταθούν με διανύσματα. Το πρώτο βήμα για να γίνει αυτό είναι τα αρχεία κειμένου να μετατραπούν σε Sequence Files. Ένα SequenceFile είναι μια κλάση του hadoop που μας επιτρέπει να γράψουμε αυθαίρετα ζεύγη από (κλειδί,τιμή) σε αυτήν. Για να μετατραπεί ύστερα το αρχείο σε διάνυσμα πρέπει το κλειδί να είναι ένα μοναδικό αναγνωριστικό (docID) του αρχείου, και η τιμή να είναι το περιεχόμενο του αρχείου σε κωδικοποίηση UTF-8.

**./bin/mahout seq2sparse **

**-i ${WORK_DIR}/sentiment-seq **

-o ${WORK_DIR}/sentiment-vectors -lnorm -nv -wt tfidf

Η εντολή αυτή παίρνει τα SequenceFiles που δημιουργήθηκαν από το seqdirectory και δημιουργεί νέα SequenceFiles τα οποία έχουν τη μορφή (docID, TF-IDF Vector). Περιέχουν δηλαδή ένα αναγνωριστικό docID του αρχείου-tweet του οποίου την πληροφορία αναπαριστούν και ένα διάνυσμα TF-IDF των λέξεων του tweet.

**./bin/mahout split **

**-i ${WORK_DIR}/sentiment-vectors/tfidf-vectors **

**--trainingOutput ${WORK_DIR}/sentiment-train-vectors **

**--testOutput ${WORK_DIR}/sentiment-test-vectors **

--randomSelectionPct 10 --overwrite --sequenceFiles -xm sequential

Η εντολή split χωρίζει τυχαία τα διανύσματα TFIDF σε δύο ομάδες. Η μια (training set) θα χρησιμοποιηθεί για το training και η άλλη (testing set) για το testing.

Με την επιλογή **--randomSelectionPct 10** δηλώνεται ότι μόλις το 10% του dataset θα χρησιμοποιηθεί για testing.

**./bin/mahout trainnb **

**-i ${WORK_DIR}/sentiment-train-vectors -el **

**-o ${WORK_DIR}/model **

**-li ${WORK_DIR}/labelindex **

-ow $c

Με την εντολή αυτή γίνεται η "εκπαίδευση" του αλγορίθμου με χρήση του training set και δημιουργείται το μοντέλο με βάση το οποίο θα γίνει το testing.

**./bin/mahout testnb **

-i ${WORK_DIR}/sentiment-test-vectors

**-m ${WORK_DIR}/model **

**-l ${WORK_DIR}/labelindex **

-ow -o ${WORK_DIR}/sentiment-testing $c

Με την εντολή αυτή γίνεται testing στο 10% των αρχείων που κρατήσαμε για testing set ώστε να διαπιστωθεί κατά πόσο ο αλγόριθμος εκπαιδεύτηκε καλά. Δηλαδή κατά πόσο μπορεί να ταξινομήσει επιτυχώς κάποια αρχεία που ήδη ξέρουμε σε ποια κατηγορία ανήκουν.

Ταξινόμηση των tweets

Ύστερα ανεβάζουμε τα tweets που πρόκειται να ταξινομηθούν στο VM του okeanos. Εκτελείται το script classify-sentiment.sh που περιέχει τις ακόλουθες εντολές:

$HADOOP dfs -put ${WORK_DIR}/sentiment-all ${WORK_DIR}/sentiment-all

Με την εντολή αυτή αντιγράφονται από το Local File System στο Hadoop Distributed File System τα tweets του training set.

**./bin/mahout seqdirectory **

**-i ${WORK_DIR}/sentiment-all **

-o ${WORK_DIR}/sentiment-seq -ow

Δημιουργούνται τα SequenceFiles.

**./bin/mahout seq2sparse **

**-i ${WORK_DIR}/sentiment-seq **

-o ${WORK_DIR}/sentiment-vectors -lnorm -nv -wt tfidf

Η εντολή αυτή παίρνει τα SequenceFiles που δημιουργήθηκαν από το seqdirectory και δημιουργεί νέα SequenceFiles τα οποία έχουν τη μορφή (docID, TF-IDF Vector).

**./bin/mahout testnb **

**-i ${WORK_DIR}/sentiment-vectors/tfidf-vectors **

**-m ${WORK_DIR}/model **

**-l ${WORK_DIR}/labelindex **

-ow -o ${WORK_DIR}/sentiment-results $c

Η εντολή testnb παίρνει τώρα όλα τα tfidf vectors και τα ταξινομεί σε θετικά και αρνητικά με βάση το model που δημιουργήθηκε στη φάση της εκπαίδευσης.

Συλλογή αποτελεσμάτων

Από το terminal εκτελούνται οι ακόλουθες εντολές:

hadoop fs -text /tmp/mahout-work-hduser/sentiment-vectors/tfidf-vectors/part-r-00000 > Vectors.txt

Με την εντολή αυτή λαμβάνεται από το Hadoop File System ένα αρχείο **Vectors.txt** που περιέχει τα ονόματα των αρχείων-tweets με τη σειρά με την οποία έγιναν διανύσματα και το διάνυσμα TF-IDF του κάθε αρχείου.

Η μορφή του **Vectors.txt** είναι όπως η ακόλουθη:

/positive/521450711465623552.txt /positive/521450711465623552.txt:{1:1.4519851207733154,2:2.299283027648926}

/positive/521450762610958336.txt /positive/521450762610958336.txt:{1:1.4519851207733154,5:1.4519851207733154}

/positive/521450833217863681.txt /positive/521450833217863681.txt:{1:1.4519851207733154}

hadoop fs -text /tmp/mahout-work-hduser/sentiment-results/part-m-00000 > VResults.txt

Με την εντολή αυτή λαμβάνεται από το Hadoop File System ένα αρχείο **VResults.txt** που περιέχει την ταξινόμηση των αρχείων-tweets σε θετικά/αρνητικά. Το αρχείο αυτό έχει την ακόλουθη μορφή:

positive {0:45.99800442173515,1:39.37862904688699}

positive {0:30.783251097520118,1:31.182979617496986}

positive {0:14.256395362354375,1:14.585933020153863}

Κάθε γραμμή περιλαμβάνει το φάκελο στον οποίο βρισκεται το αρχείο που δόθηκε προς ταξινόμηση και στη συνέχεια τα βάρη που δηλώνουν πόσο αρνητικό (0:...) και πόσο θετικό (1:...) συναίσθημα κρίθηκε ότι έχει. Θεωρητικά αν το βάρος του 0 είναι μικρότερο από του 1, το tweet έχει ταξινομηθεί ως αρνητικού συναισθήματος. Στην πράξη αργότερα στην εργασία ορίζεται μια διαφορά μεταξύ 0 και 1 για την οποία τα tweets θεωρούνται neutral.

Το αρχείο **Vectors.txt** μας χρειάζεται για να ξέρουμε σε ποιο tweet αναφέρεται κάθε γραμμή του **VResults.txt**, καθώς η σειρά των tweets σε κάθε αρχείο txt είναι η ίδια.

Το ότι ο φάκελος στον οποίο βρίσκεται το tweet ονομάζεται positive δεν έχει καμία σημασία για την ταξινόμηση των tweets. Το όνομα του φακέλου στον οποίο αποθηκεύεται καθε αρχείο παίζει ρόλο μόνο κατά την εκπαίδευση καθώς τότε ταξινομείται σε θετικό ή αρνητικό ανάλογα με το σε ποιο φάκελο βρίσκεται.

5.1.6 Επεξεργασία των clusters και εύρεση ομοιοτήτων μεταξύ τους με χρήση cosine similarity

Γράφτηκε πρόγραμμα σε Java που διαβάζει τα αποτελέσματα του clustering, βρίσκει τους clusters, τα top terms και τα βάρη τους και τα αποθηκεύει τον καθένα σαν ξεχωριστό αρχείο txt σε φακέλους ανά πόλη. Πχ στο φάκελο με τους clusters του Λονδίνου περιέχονται 1400 αρχεία (200 για κάθε μέρα) το καθένα με όνομα "day+number.txt" Ενδεικτικά δίνεται το αρχείο fri24.txt από το φάκελο για τη Νέα Υόρκη:

```
terrace
2.451963354040075
star
2.205934171323423
hudson
1.6544993276949282
happiness
1.3197120560540094
park
0.8423583507537842
beginning
0.6452366157814309
projects
0.6328622323495371
```

```
bethesda
0.5401274893018935
latina
0.5277306238810221
tier
0.42468480710630063
couple
0.4196245140499539
river
0.4150557253095839
giveback
0.3707399015073423
whoistaller
0.3707399015073423
nyc
0.3683046720646046
ferry
0.3087407571298105
awards
0.29694617236102067
party
0.22634730515656648
future
0.19009235170152453
command
0.18536995075367116
prose
0.18536995075367116
intermission
0.18536995075367116
schwartzman
0.18536995075367116
trek
0.18536995075367116
glen
0.18536995075367116
rondo
0.18536995075367116
marcy
0.18536995075367116
liners
0.18536995075367116
regina
0.18536995075367116
comics
0.18004249643396447
```

Το όνομα του αρχείου δηλώνει ότι περιέχει τον cluster νούμερο 24 ο οποίος ανήκει στην Παρασκευή. Παρατηρούμε ότι ο cluster έχει 30 όρους και τα αντίστοιχα βάρη τους. Δημιουργούνται αρχεία για όλους τους clusters. Με άλλο πρόγραμμα java που γράφτηκε γίνεται η σύγκριση των clusters της κάθε μέρας με τους clusters των υπολοίπων ημερών χρησιμοποιώντας το cosine similarity. Πιο αναλυτικά, για κάθε cluster δημιουργείται διάνυσμα όπου οι διαστάσεις είναι οι λέξεις που περιέχει και το βάρος κάτω από την κάθε λέξη είναι συντελεστής σε αυτή τη διάσταση.

Έτσι πχ για τον αποπάνω cluster στη διάσταση terrace ο συντελεστής θα είναι 2.451963354040075.

Για ευκολία στην ανάλυση ενός συγκεκριμένου παραδείγματος θεωρούμε τους εξής clusters οι οποίοι αντί για 30 έχουν 3 όρους ο καθένας:

fri24: `terrace`
`2`
`star`
`1.5`
`hudson`
`1`

mon35: `life`
`3`
`terrace`
`2`
`hudson`
`0.5`

Παίρνουμε τον cluster *fri24* και αναγνωρίζεται ως cluster ημέρας Παρασκευής. Για όλους τους clusters των υπόλοιπων ημερών εξετάζουμε αν έχουν κοινούς όρους. Αν δεν έχουν, cosine similarity=0. Αν έχουν (όπως συμβαίνει στο παράδειγμά μας), παίρνουμε το διάνυσμα του fri24: 2terrace+1.5star+1hudson και το διάνυσμα του Mon35: 3life+2terrace+0.5hudson και τα μετατρέπουμε ως εξής: fri24: 2terrace+1hudson+1.5star+0+0+0 (0 μέχρι την 6η διάσταση) και mon35: 2terrace+0.5hudson+0+3life+0+0. Δηλαδή οι κοινοί όροι μπαίνουν στις πρώτες ίδιες θέσεις και των δύο διανυσμάτων. Ύστερα στο πρώτο διάνυσμα μπαίνουν οι υπόλοιποι δικοί του όροι μέχρι τη θέση 3 και συμπληρώνεται με μηδενικά μέχρι το τέλος (θέση 6). Το δεύτερο συμπληρώνεται με μηδενικά από εκεί που τελείωσαν οι κοινοί όροι μέχρι και τη θέση 3, ύστερα μπαίνουν οι υπόλοιποι δικοί του όροι και όταν τελειώσουν αυτοί μπαίνουν μηδενικά μέχρι το τέλος. Στην πραγματική εφαρμογή αντί για 3 και 6 έχουμε αντίστοιχα 30 και 60.

Παίρνουμε έπειτα τα δύο νέα διανύσματα και βρίσκουμε το συνημίτονο της μεταξύ τους γωνίας. Για κάθε μέρα (από τις 6 που απομένουν εκτός από τη μέρα του διανύσματος που έχει επιλεγεί αρχικά) αναζητείται το διάνυσμα με το οποίο η γωνία είναι η μικρότερη και το ζεύγος των διανυσμάτων αποθηκεύεται σε αρχείο **clusterrelations.csv**. Αυτή η διαδικασία γίνεται για όλα τα διανύσματα. Για το παράδειγμα που εξετάστηκε, στο αρχείο csv η γραμμή που θα γραφόταν θα ήταν: "20150612","24","20150608","35", που σημαίνει ότι **ο cluster της 12ης Ιουνίου 2015 (που είναι Παρασκευή) μοιάζει περισσότερο με τον cluster 35 της Δευτέρας 8 Ιουνίου**. Λίγο μετά υπολογίζονται και αποθηκεύονται οι clusters Τρίτης έως Κυριακής (εκτός Παρασκευής γιατί από Παρασκευή είναι ο αρχικός cluster fri24) που μοιάζουν περισσότερο με τον fri24. Η διαδικασία αυτή γίνεται για όλους τους clusters.

5.1.7 Δημιουργία των αρχείων csv και txt με τα δεδομένα προς εισαγωγή στη βάση

Στο προηγούμενο βήμα δημιουργήθηκε το αρχείο clusterrelations.csv που περιέχει τα ζευγάρια όμοιων clusters διαφορετικών ημερών.Δημιουργούνται επίσης τα εξής αρχεία:

clustercypher.txt

Για τη δημιουργία του αρχείου αυτού διαβάζονται με εφαρμογή που γράφτηκε σε Java τα αποτελέσματα του clustering και εξάγονται τα δεδομένα των clusters: ημερομηνία, id, σημαντικότεροι όροι και τα αντίστοιχα βάρη τους. Ύστερα για κάθε cluster δημιουργείται ένα Cypher Query που εισάγει τα δεδομένα του cluster στη βάση δεδομένων. Όλα τα cypher queries για τους clusters κάθε πόλης βρίσκονται στο clustercypher.txt της πόλης. Παράδειγμα γραμμής του nyclustercypher.txt:

```
MERGE (c:Cluster {id:10527,date:20150608,city:"NY",terms:
["comingsoon","shopaus","laptop","magic","hand"],weights:
[9.544743537902832,4.772371768951416,4.425797939300537,4.170385360717773,3.67375946044
92188]})
```

relationcypher.csv

Για τη δημιουργία του αρχείου αυτού από την ίδια εφαρμογή διαβάζονται και πάλι τα αποτελέσματα που πήραμε από το clustering και τα οποία περιέχουν το σε ποιον cluster έχει κατανεμηθεί το κάθε tweet. Έτσι σε κάθε γραμμή του relationcypher.csv γράφεται το id του tweet και το id του cluster στον οποίο έχει κατανεμηθεί. Παράδειγμα γραμμής: `"521450270606503936","5622".`

tweetcypher.csv

Για τη δημιουργία του αρχείου αυτού από την ίδια εφαρμογή διαβάζονται όλα τα αρχικά tweets και εξάγονται οι πληροφορίες για το id του, την ημερομηνία του, γεωγραφικές συντεταγμένες, ώρα δημοσίευσης, το Username του χρήστη που έκανε τη δημοσίευση και το περιεχόμενο του tweet. Χρησιμοποιείται επίσης το clusteredpoints.txt από τα αποτελέσματα του clustering και εξάγεται το cosine distance κάθε tweet-αρχείου από το κεντρικό διάνυσμα του cluster στον οποίο κατανεμήθηκε. Παράδειγμα γραμμής tweetcypher.csv:
"521450276444966913","20141013","1030008","40.75150004","-73.99878471","Snappleyard","Let\'s talk about Maggie wearing Daryl\'s poncho","0.7857568303063941"

Με τη σειρά οι πληροφορίες είναι:"ID","Date","Time(αφαιρώντας το '1' από την αρχή της συβολοσειράς),"latitude","longitude","χρήστης","περιεχόμενο","απόσταση από το κεντρικό διάνυσμα του cluster".

sentimentcypher.csv

Άλλη εφαρμογή που γράφτηκε επίσης σε java εξετάζει για κάθε πόλη και μέρα τα αρχεία Vectors.txt και Vresults.txt που είναι αποτελέσματα του classification, συγκρίνει τις γραμμές τους και για κάθε id ενός tweet παίρνει τα βάρη του negative και positive και τα γράφει στο sentimentcypher.csv ως εξής:
`"521450270606503936","373.73354904980806","399.6912322611348"`

Στο αποπάνω παράδειγμα παρατηρείται ότι το tweet με id 521450270606503936 έχει βάρος positive 373.73354904980806 και negative 399.6912322611348. Θα μπορούσαμε να πούμε ότι το classification συμπέρανε ότι το συγκεκριμένο tweet έχει αρνητικό συναίσθημα.

clusterrelations.csv

Το αρχείο αυτό περιέχει τα ζευγάρια όμοιων clusters διαφορετικών ημερών και δημιουργήθηκε στο 5.1.6.

hyperclusterrelations.csv

Το αρχείο αυτό περιέχει τα ζευγάρια όμοιων clusters διαφορετικών πόλεων.

5.1.8 Εισαγωγή δεδομένων στη βάση neo4j

Η εισαγωγή δεδομένων στη βάση neo4j γίνεται με δύο τρόπους:

- Εισαγωγή από το web console της neo4j: Χρησιμοποιήθηκε η έκδοση "Community" της βάσης neo4j. Με το που "σηκώνεται" η βάση, προσφέρεται ένα web interface το οποίο είναι προσβάσιμο μέσω του browser και από το οποίο μπορούμε να εισαγάγουμε στη βάση μεγάλο όγκο δεδομένων χρησιμοποιώντας cypher queries και αρχεία csv.
- Εισαγωγή με πρόγραμμα γραμμένο σε java: Το neo4j προσφέρει βιβλιοθήκες που σηκώνουν/δημιουργούν μια βάση και επικοινωνούν μαζί της εκτελώντας cypher queries.

Τα δεδομένα τελικώς εισάγονται στη βάση ακολουθώντας τα εξής βήματα:

1.
```
USING PERIODIC COMMIT LOAD CSV FROM
'file:C:/Users/giorgis/Desktop/diplwmatikh/raw_tweets/London/lond
ontweetcypher.csv' AS line CREATE (a:Tweet { id: line[0], date:
toInt(line[1]),time: toInt(line[2]),latitude: toFloat(line[3]),
longitude: toFloat(line[4]), user:line[5], text: line[6],
distance: toFloat(line[7])});
```
Με την εντολή αυτή δηλώνουμε ότι διαβάζεται το csv που περιέχει τις πληροφορίες των Tweets (εδώ για το Λονδίνο) **ανά γραμμή** και δημιουργείται στη βάση ένα node που είναι Tweet και περιέχει id,date,time,latitude,longitude,user,text,distance. Οι πληροφορίες αυτές περιέχονται με την ίδια σειρά σε κάθε γραμμή του αρχείου csv όπου η κάθε πληροφορία χωρίζεται από τις υπόλοιπες με κόμμα ,. Κάθε γραμμή αναφέρεται και σε διαφορετικό tweet.

2. Αφού εισαχθούν τα tweets στη βάση, εισάγουμε με εφαρμογή γραμμένη σε java η οποία χρησιμοποιεί βιβλιοθήκες του neo4j τα δεδομένα για τους clusters που υπάρχουν στο αρχείο clustercypher.txt Όπως έχει ειπωθεί, το αρχείο αυτό περιέχει cypher queries και το μόνο που κάνει η εφαρμογή μας είναι να εκτελεί όλά αυτά τα queries.

3. Δημιουργούμε στη βάση indexes ως προς τα id και date των tweets και των clusters, για να γίνεται ταχύτερα η αναζήτηση. Αυτό γίνεται πάλι από το web interface εκτελώντας τις εντολές:
```
CREATE INDEX ON :Cluster(id)
CREATE INDEX ON :Cluster(date)
CREATE INDEX ON :Tweet(id)
CREATE INDEX ON :Tweet(date)
```

4.
```
USING PERIODIC COMMIT LOAD CSV FROM
'file:C:/Users/giorgis/Desktop/diplwmatikh/raw_tweets/London/lond
onrelationcypher.csv' AS line MATCH (t:Tweet { id:line[0] }),
(c:Cluster {id:toInt(line[1]) }) WHERE t.date=c.date CREATE (t)-
[r:BELONGS_TO]->(c);
```
Με την εντολή αυτή διαβάζεται το csv που περιέχει τις πληροφορίες που συνδέουν το κάθε tweet με τον cluster στον οποίο έχει τοποθετηθεί κατά το clustering. Και πάλι ανά γραμμή διαβάζεται το id του tweet και το id του cluster. Αναζητείται στη βάση tweet με το συγκεκριμένο id (line[0]) και cluster με το επίσης συγκεκριμένο id (line[1]) και ζητείται tweet και cluster να έχουν την ίδια ημερομηνία (γιατί μπορεί clusters διαφορετικών ημερών να έχουν ίδιο id. Έπειτα δημιουργείται σχέση BELONGS_TO από το tweet προς τον cluster.

5.
```
USING PERIODIC COMMIT LOAD CSV FROM
'file:C:/Users/giorgis/Desktop/diplwmatikh/raw_tweets/London/lond
onsentimentcypher.csv' AS line MATCH (t:Tweet { id:line[0] }) SET
t.negative=toFloat(line[1]), t.positive=toFloat(line[2]);
```
Με την εντολή αυτή διαβάζεται ανά γραμμή το csv που περιέχει τις πληροφορίες για το συναίσθημα του κάθε tweet. Αναζητείται στη βάση το tweet που έχει το συγκεκριμένο id (line[0]) και του προσθέτουμε properties negative (line[1]) και positive(line[2]).

6.
```
USING PERIODIC COMMIT LOAD CSV FROM
'file:C:/Users/giorgis/Desktop/diplwmatikh/raw_tweets/London/clus
terrelations_London.csv' AS line MATCH (a:Cluster
{ id:toInt(line[1]) , date:toInt(line[0] ,  city: "London") }),
(b:Cluster {id:toInt(line[3]) , date:toInt(line[2]) ,  city:
"London" }) CREATE (a)-[r:SIMILAR_TO]->(b);
```
Με την εντολή αυτή διαβάζεται ανά γραμμή το csv που περιέχει τις πληροφορίες για τους όμοιους clusters διαφορετικών ημερών της ίδιας πόλης.

Αναζητούνται clusters με βάση το Id, την ημερομηνία και την πόλη και δημιουργείται σχέση μεταξύ τους με όνομα SIMILAR_TO

```
7. USING PERIODIC COMMIT LOAD CSV FROM
   'file:/hyperclusterrelations.csv' AS line MATCH (a:Cluster
   { date:toInt(line[0]), id:toInt(line[1]), city:line[2]}),
   (b:Cluster { date:toInt(line[3]), id:toInt(line[4]),
   city:line[5]}) MERGE (a)-[r:OTHER_CITY]->(b);
```

Με την εντολή αυτή διαβάζεται ανά γραμμή το csv που περιέχει τις πληροφορίες για τους όμοιους clusters διαφορετικών πόλεων. Αναζητούνται clusters με βάση το Id, την ημερομηνία και την πόλη και δημιουργείται σχέση μεταξύ τους με όνομα OTHER_CITY.

Κάθε ένα από αυτά τα βήματα 1,2,4,5,6 εκτελείται μία φορά για κάθε πόλη πριν προχωρήσουμε στο επόμενο.

5.1.9 Δημιουργία web εφαρμογής που επικοινωνεί με τη βάση και παρουσιάζει τα δεδομένα στο χρήστη με κατανοητό τρόπο

Χρησιμοποιώντας την HTML φτιάχτηκαν δύο σελίδες: Η clusters.html και η index.html. Η εμφάνιση των σελίδων αυτών ελέγχεται με τη χρήση CSS.

CLUSTERS.HTML

Στη σελίδα αυτή ο χρήστης μπορεί να δει τους clusters μιας πόλης. Με το που φορτώνει η σελίδα εμφανίζονται στα δεξιά οι 20 clusters της πόλης οι οποίοι έχουν συνδεδεμένα πάνω τους τα περισσότερα tweets. Αυτό γίνεται με χρήση javascript, ώστε να αλλάζει το περιεχόμενο της σελίδας δυναμικά ανάλογα με τις επιθυμίες του χρήστη. Συγκεκριμένα για την εμφάνιση των 20 clusters με τα περισσότερα tweets, η λειτουργία υλοποιείται με τον ακόλουθο κώδικα javascript:

```
var startingstatement = {
       "statements" : [ {
             "statement" : "MATCH (n:Tweet)-[r]->(x:Cluster) WHERE
length(x.terms)>1 RETURN x, COUNT ORDER BY COUNT DESC LIMIT 20"
       } ]
};

$.ajax({
        headers: {
            'Accept': 'application/json',
            'Content-Type': 'application/json'
        },
        'type': 'POST',
       'url': 'http://83.212.116.97:7474/db/data/transaction/commit',
        'data': JSON.stringify(startingstatement),
        'dataType': 'json',
        'beforeSend': function() {
                loadingdiv.innerHTML='LOADING...';
            },
        'complete': function() {
                loadingdiv.innerHTML='';
            },
          'success': εδώ ορίζεται συνάρτηση που θα εκτελεστεί αν δοθεί απάντηση 200 OK στο
HTTP request,
        'error':function(jqXHR, textStatus, errorThrown){
                    alert(errorThrown);
                            console.log("error: "+JSON.stringify(textStatus)
+JSON.stringify(jqXHR));
        }
    });
```

Ο παραπάνω κώδικας κάνεις τα εξής:

- Αρχικά ορίζουμε μια μεταβλητή startingstatement στην οποία καταχωρούμε το cypher query MATCH (n:Tweet)-[r]->(x:Cluster) WHERE length(x.terms)>1 RETURN x, COUNT(r) ORDER BY COUNT(r) DESC LIMIT 20. Αυτό ζητάει να βρεθούν όλα τα tweets n που συνδέονται με clusters x μέσω μιας σχέσης r όπου κάθε cluster x έχει πάνω από έναν όρο (term). Ζητάει να επιστραφούν όλοι οι clusters x που βρέθηκαν, και να παρουσιαστούν οι 20 πρώτοι ανάλογα με το πόσες σχέσεις r βρέθηκε ότι έχουν (δηλαδή με πόσα tweets συνδέονται). Η μορφή του startingstatement είναι JSON.
- Έπειτα χρησιμοποιώντας τη συνάρτηση ajax της βιβλιοθήκης jquery στέλνουμε έννα HTTP request με τα κατάλληλα headers και options. Πιο συγκεκριμένα, με τα headers δηλώνουμε στη βάση ότι στελνουμε και δεχόμαστε σαν απάντηση δεδομένα τύπου json. Η μέθοδος που χρησιμοποιούμε είναι η POST και το request γινεται στο url που ορίζεται.
- Στη συνέχεια το αντικείμενο startingstatement μετατρέπεται σε JSON text και αποθηκεύεται σε ένα string με τη μέθοδο JSON.stringify(). Αυτό το string περνάμε σαν data μαζί με το HTTP request.
- Ο τύπος 'dataType' των δεδομένων ορίζεται ότι είναι json.
- Σε περίπτωση που το request γίνει επιτυχώς, εκτελείται η συνάρτηση που ορίζεται στο 'success', αλλιώς εκτελείται η συνάρτηση που ορίζεται στο 'error'.

Ο κώδικας της συνάρτησης που ορίζεται στο success είναι ο ακόλουθος:

```
function(singledata, xhr, textStatus){
	var response=JSON.stringify(singledata ,null, 4);
	var contenttoput='';
	var json_obj1 = $.parseJSON(response);
	for (var i = 0; i < json_obj1.results[0].data.length; ++i) {
		var important = json_obj1.results[0].data[i].row[0];
		clusterids[i]=important.id;
		contenttoput = contenttoput + '<div class="cluster"
id="'+important.id+'" onclick="changeContent ('+important.id+')"><div
class="clustertitle"><u>Cluster nr. '+important.id+' date:
'+important.date+'</u></div><div class="topterms">';
		for (var j = 0; j < important.terms.length; ++j) {
			if (important.weights[j]>1 || j<12){
				contenttoput = contenttoput + '<div
class="term">'+important.terms[j]+'</div>';
			}
				else {break;}
		}
			contenttoput = contenttoput + '</div></div>';
	}
		clusterlist.innerHTML = contenttoput;
}
```

Παίρνουμε το singledata που επιστρέφεται με το HTTP response, το μετατρέπουμε σε JSON text και αποθηκεύεται στη μεταβλητή response. Η μεταβλητή contenttoput θα περιέχει html κώδικα με τα στοιχεία που θα παρουσιαστούν στο χρήστη. Η parseJSON μετατρέπει το response από JSON text σε μορφή που είναι επεξεργάσιμη με javascript. Αποθηκεύεται στο json_obj1.

Ένα παράδειγμα response είναι:

```
{
    "results": [
        {
            "columns": [
```

```
            "x",
            "COUNT(r)"
        ],
        "data": [
            {
                "row": [
                    {
                        "city": "NY",
                        "id": 15161,
                        "date": 20150611,
                        "weights": [
                            3.532086961919611,
                            3.494650563326749,
                            ...
                        ],
                        "terms": [
                            "york",
                            "new",
                            ...
                        ]
                    },
                    1038
                ]
            },
            {
                "row": [
                    {
                        "city": "NY",
                        "id": 931,
                        "date": 20150613,
                        "weights": [
                            3.322412851474268,
                            3.3192167547966935,
                            ...
                        ],
                        "terms": [
                            "new",
                            "york",
                            ...
                        ]
                    },
                    1032
                ]
            },
            {
                "row": [...
                            ]
            },
            { κι άλλα 18 rows...
            }
        ]
    }
],
```

```
    "errors": []
}
```

Η επεξεργασία του response γίνεται μέσα στο for loop του success function:

1. Για κάθε στοιχείο του data παίρνουμε το row[0] που είναι ένα javascript object και το αποθηκεύουμε στη μεταβλητή important.
2. Ύστερα στο contenttoput βάζουμε με τη σειρά ανά cluster τα top terms (σημαντικότερους όρους). Ο HTML κώδικας που τα περιβάλλει φτιάχνει με συγκεκριμένο τρόπο τα divs ώστε να παρουσιαστεί σωστά ο κάθε cluster με τα δικά του terms. Επειδή κάθε cluster έχει 30 terms, κρατάμε τουλάχιστον 12. Κρατάμε επίσης όσα από το 13ο term και μετά έχουν βάρος μεγαλύτερο του 1. Τα δεδομένα αποθηκεύονται στο contenttuput.
3. Στο clusters.html έχει οριστεί συγκεκριμένο div στα δεξιά της σελίδας με id="clusterlist" του οποίου το περιεχόμενο αλλάζει με την εντολή **clusterlist.innerHTML = contenttoput**. Έχει οριστεί στον html κώδικα του contenttoput η συνάρτηση **onclick="changeContent ('+important.id+')** η οποία εκτελείται όταν ο χρήστης κάνει κλικ στο div του συγκεκριμένου cluster.

Μια άλλη λειτουργία που προσφέρει το clusters.html είναι η αναζήτηση σε όλους τους clusters μιας πόλης για μια συγκεκριμένη λέξη. Στα αποτελέσματα επιστρέφονται 20 clusters που περιέχουν τη λέξη στα top terms τους με πρώτο cluster αυτόν που περιέχει περισσότερα tweets.

Το statement για τη συγκεκριμένη αναζήτηση είναι:

```
"statement" : "MATCH (t:Tweet)-[r]->(c:Cluster) where
c.city=\""+city+ "\" AND \"" +
document.getElementById('searchbox').value + "\" in c.terms return c,
COUNT(r) ORDER BY COUNT(r) DESC LIMIT 20"
```

Οι clusters που επιστρέφονται από τις αναζητήσεις εμφανίζονται στα δεξιά της σελίδας. Και πάλι στον html κώδικα ορίζεται η συνάρτηση **onclick="changeContent ('+important.id+')** η οποία εκτελείται όταν ο χρήστης κάνει κλικ στο div ενός συγκεκριμένου cluster. Πατώντας πάνω σε έναν cluster εμφανίζονται στο χάρτη τα tweets που είναι συνδεδεμένα με αυτόν. Εμφανίζονται μόνο όσα tweets έχουν αποθηκευμένες γεωγραφικές συντεταγμένες.

Η συνάρτηση changeContent(id): Η changeContent βρίσκει όλα τα tweets του cluster με id=id και τα παρουσιάζει σε χάρτη στις συντεταγμένες τους. Παρουσιάζει επίσης το γενικό sentiment του συγκεκριμένου cluster μετρώντας πόσα αρνητικά και πόσα θετικά tweets έχει. Ύστερα πατώντας σε κάποιο tweet ο χρήστης μπορεί να δει το περιεχόμενό του.

Τα μέρη της changeContent αναλυτικά:

- Μέρος 1ο: Cypher query για αναζήτηση των tweets από συγκεκριμένο cluster:

```
var data = {
      "statements" : [ {
            "statement" : "MATCH (a:Tweet)-[r]->(b:Cluster {id:"+id+"})
WHERE b.city=\""+city+"\" RETURN a"
      } ]
};
```

- Μέρος 2ο: HTTP request:

```
$.ajax({
headers: {
      'Accept': 'application/json',
        'Content-Type': 'application/json'
    },
    'type': 'POST',
   'url': 'http://83.212.116.97:7474/db/data/transaction/commit',
    'data': JSON.stringify(data),
    'dataType': 'json',
    'beforeSend': function() {
               loadingdiv.innerHTML='LOADING...';
           },
   'complete': function() {
               loadingdiv.innerHTML='';
           },
   'success': function(singledata, xhr, textStatus){ Συνάρτηση που εκτελείται αν το
HTTP response είναι 200 OK
        },
    'error':function(jqXHR, textStatus, errorThrown){
                alert(errorThrown);
      }
});}
```

Αν το HTTP request έγινε επιτυχώς, πήραμε απάντηση και εκτελείται το **success function**.
Το success function αποτελείται από τα ακόλουθα μέρη:

- Μέρος 1ο: Επεξεργασία των αποτελεσμάτων και εξαγωγή συνολικού συναισθήματος του cluster:

```
function(data, xhr, textStatus){
             var response=JSON.stringify(data ,null, 4);
             var json_obj = $.parseJSON(response);
             var positive=0;
             var negative=0;
             var neutral=0;
             var contenttoput='';
             features = new Array(json_obj.results[0].data.length);
             for (var i = 0; i < json_obj.results[0].data.length; ++i) {
                    var important = json_obj.results[0].data[i].row[0];
                    if (important.positive+50<important.negative){ positive+
+;}
                    else if (important.positive>important.negative+50)
{ negative++;}
                    else{neutral++;}
                    var coordinates =
ol.proj.transform([json_obj.results[0].data[i].row[0].longitude,json_obj.res
ults[0].data[i].row[0].latitude], 'EPSG:4326', 'EPSG:3857');
                    features[i] = new ol.Feature(new
ol.geom.Point(coordinates));

features[i].setId=(json_obj.results[0].data[i].row[0].id);
             }
             contenttoput='<div id="clustersentimenttotal">Total Tweets: '+
(positive+negative+neutral)+'</div>';
```

```
            contenttoput=contenttoput+'<div id="positive">positive:
'+100*positive/(positive+negative+neutral)+'%</div>';
            contenttoput=contenttoput+'<div id="negative">negative:
'+100*negative/(positive+negative+neutral)+'%</div>';
            contenttoput=contenttoput+'<div
id="neutral">neutral/irrelevant: '+100*neutral/(positive+negative+neutral)
+'%</div>';
            clustersentiment.innerHTML=contenttoput;
}
```

Στο response παίρνουμε όλα τα tweets που περιέχει ο cluster. Στο for loop, κάθε tweet αποθηκεύεται στη μεταβλητή important, εξετάζεται αν είναι positive, negative ή neutral/irrelevant και αυξάνεται ο αντίστοιχος μετρητής. Ένα tweet είναι neutral αν η διαφορά μεταξύ των βαρών positive και negative που πήρε από το classification είναι το πολύ 50. Η τιμή αυτή είναι εμπειρική και επελέγη ως μια τιμή που εξασφαλίζει ότι αν ένα tweet χαρακτηριστεί τελικά positive ή negative, ο χαρακτηρισμός έχει μεγάλη πιθανότητα να είναι επιτυχής. Στον πίνακα **features** αποθηκεύονται οι συντεταγμένες του κάθε tweet αφού μετατραπεί το σύστημα συντεταγμένων τους από EPSG:3857 (projected) σε EPSG: 4326 (geographic). Η μετατροπή υπαγορεύεται από τη βιβλιοθήκη χαρτών που χρησιμοποιείται πιο κάτω. Ύστερα στο contenttoput αποθηκεύεται html με τα συγκεντρωτικά δεδομένα του cluster που έχει επιλεγεί από το χρήστη. Παρουσιάζονται σε ποσοστά το πόσα tweets είναι θετικά, πόσα αρνητικά και πόσα neutral/irrelevant. Έτσι ο χρήστης αποκτά μια γενική εικόνα για το συναίσθημα του cluster. Στο clusters.html έχει οριστεί συγκεκριμένο div στα δεξιά της σελίδας με id="clustersentiment" του οποίου το περιεχόμενο αλλάζει με την εντολή clustersentiment.innerHTML = contenttoput.

- Μέρος 2ο: Σχεδίαση χάρτη:

```
var source = new ol.source.Vector({
      features: features,
});
if (drawnmap==1){ clusterSource.clear();}
clusterSource = new ol.source.Cluster({
      distance: 40,
      source: source
});
var styleCache = {};
var clusters = new ol.layer.Vector({
      source: clusterSource,
      style: function(feature, resolution) {
            var size = feature.get('features').length;
            var style = styleCache[size];
            if (!style) {
                  style = [new ol.style.Style({
                        image: new ol.style.Circle({
                              radius: 10,
                              stroke: new ol.style.Stroke({
                                    color: '#fff'
                              }),
                              fill: new ol.style.Fill({
                                    color: '#3399CC'
                              })
                        }),
                        text: new ol.style.Text({
```

```
                                text: size.toString(),
                                fill: new ol.style.Fill({
                                      color: '#fff'
                                })
                          })
                    })];
                    styleCache[size] = style;
              }
              return style;
       }
});
var raster = new ol.layer.Tile({
source: new ol.source.MapQuest({layer: 'osm'})
});
var raw = new ol.layer.Vector({
source: source,
});
if (drawnmap==1){
map.removeLayer(clusters);
map.removeLayer(raster);
map.addLayer(raster);
map.addLayer(clusters);
}
else{
       map = new ol.Map({
              layers: [raster, clusters],
              renderer: 'canvas',
              target: 'map',
              view: new ol.View({
                    center: ol.proj.transform(
                    [ -73.976863, 40.753043], 'EPSG:4326', 'EPSG:3857'),
                    zoom: 10
              })
       });
}
map.on('click', function(evt) {
displayFeatureInfo(evt.pixel);
})
```

Ο παραπάνω κώδικας προέρχεται από τη σελίδα openlayers.org και έχει υποστεί ορισμένες μετατροπές. Με την εντολή **var clusters = new ol.layer.Vector(...)** εισάγονται στο αντικείμενο clusters όλα τα ζεύγη συντεταγμένων κάθε tweet από τον πίνακα *features* όπου είχαν αποθηκευτεί προηγουμένως. Ύστερα ελέγχεται αν είναι ήδη σχεδιασμένος ο χάρτης **if (drawnmap==1)**. Αν ναι, αφαιρούνται τα προηγούμενα tweets και σχεδιάζονται τα tweets του cluster που μόλις έχει επιλέξει ο χρήστης με τις εντολές **map.removeLayer(clusters); map.removeLayer(raster); map.addLayer(raster); map.addLayer(clusters);**. Αλλιώς σχεδιάζεται από την αρχή ο χάρτης **map = new ol.Map** και εισάγονται τα tweets με την εντολή **layers: [raster, clusters]**. Ο χάρτης είναι αρχικά κεντραρισμένος στις συντεταγμένες -73.976863, 40.753043 (Νέα Υόρκη). Με την εντολή **map.on('click', function(evt) { displayFeatureInfo(evt.pixel);})** ορίζεται event trigger κάθε φορά που ο χρήστης κάνει κλικ στο χάρτη και καλείται η συνάρτηση **displayFeatureInfo**.

- Μέρος 3ο: Η συνάρτηση **displayFeatureInfo**:

```
var displayFeatureInfo = function(pixel) {
      var selected_features =
map.forEachFeatureAtPixel(pixel,function(feature, layer) {
            var specialcoords =
ol.proj.transform(map.getCoordinateFromPixel(pixel), 'EPSG:3857',
'EPSG:4326');
            var helpy1=specialcoords[0]-0.0015;
            var helpy2=specialcoords[0]+0.0015;
            var helpy3=specialcoords[1]-0.0015;
            var helpy4=specialcoords[1]+0.0015;
            var singledata = {
                  "statements" : [ {
                        "statement" : "MATCH (a:Tweet)-[r]-(b) WHERE
a.longitude>= "+helpy1+" AND a.longitude<= "+helpy2+" AND a.latitude>=
"+helpy3+" AND a.latitude<= "+helpy4+" AND b.id= "+selectedid+" RETURN a"
                  } ]
            };
      $.ajax({
            headers: {
                  'Accept': 'application/json',
                  'Content-Type': 'application/json'
            },
            'type': 'POST',
            'url': 'http://83.212.116.97:7474/db/data/transaction/commit',
            'data': JSON.stringify(singledata),
            'dataType': 'json',
            'beforeSend': function() {
              loadingdiv.innerHTML='LOADING...';
           },
       'complete': function() {
              loadingdiv.innerHTML='';
           },
            'success': function(singledata, xhr, textStatus){
            var response=JSON.stringify(singledata ,null, 4);
            var contenttoput='';
            var json_obj2 = $.parseJSON(response);
            tweetcontent.innerHTML = '';
            for (var i = 0; i < json_obj2.results[0].data.length; ++i){
                  var important = json_obj2.results[0].data[i].row[0];
                  var tweetsentiment='unknown';
                  if (important.positive+50<important.negative )
                  { tweetsentiment='positive';}
                  else if (important.positive>important.negative+50)
                        { tweetsentiment='negative';}
                  else{tweetsentiment='neutral/irrelevant';}
                  contenttoput = contenttoput + '<div id="tweetid"> Status
id: ' + json_obj2.results[0].data[i].row[0].id +'</div><div id="tweettext">'
+ json_obj2.results[0].data[i].row[0].text+'</div>'+'<div
id="sentiment">Tweet classified as: <b>'+tweetsentiment+'</b> pos vs neg:
'+important.positive+' , '+important.negative+'</div><hr />';
            }
            tweetcontent.innerHTML = contenttoput;
            },
            'error':function(jqXHR, textStatus, errorThrown){
```

```
                                    alert(errorThrown);
                }
        });
    return feature;
  });
};
```

Η συνάρτηση αυτή καλείται όταν ο χρήστης κάνει κλικ στο χάρτη. Επιστρέφονται οι συντεταγμένες του pixel και μετατρέπονται από EPSG: 4326 (geographic) σε EPSG:3857 (projected) σύστημα. Στη βάση no4j οι αποθηκευμένες συντεταγμένες είναι σε προβολικό σύστημα.

Από το ζεύγος συντεταγμένων στο οποίο έχει κλικάρει ο χρήστης, φτιάχνεται ένα τετράγωνο με πλευρά 0.003 και κέντρο τις συντεταγμένες του χρήστη.

var helpy1=specialcoords[0]-0.0015;

var helpy2=specialcoords[0]+0.0015;

var helpy3=specialcoords[1]-0.0015;

var helpy4=specialcoords[1]+0.0015;

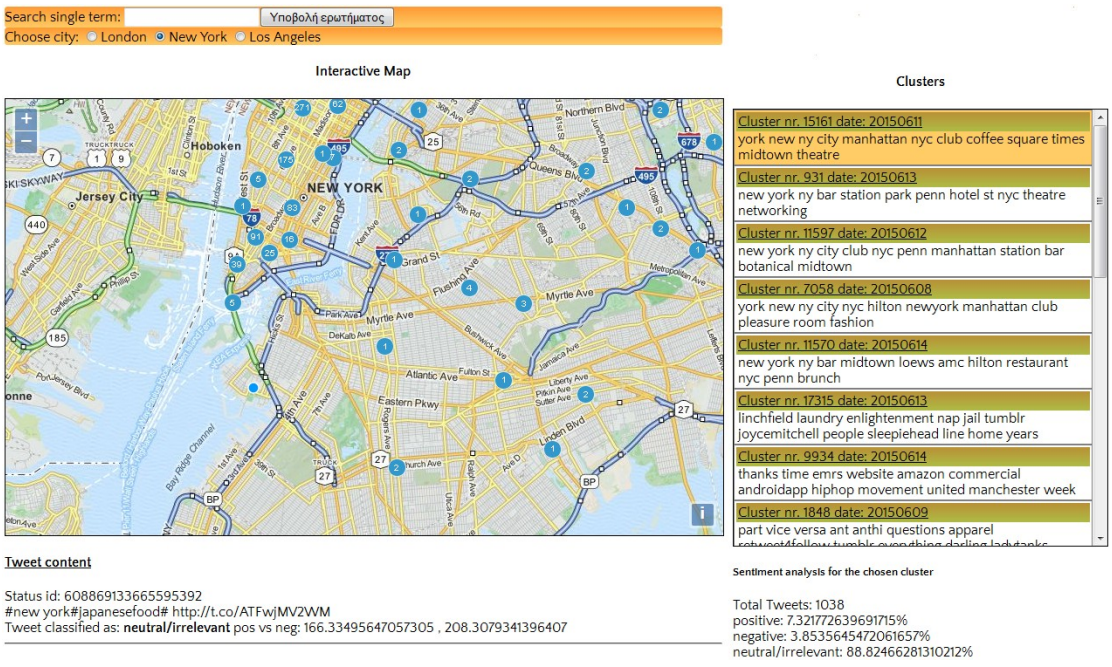
Το τετράγωνο αυτό δίνεται σαν περιορισμός στο cypher query του HTTP request:

"statement" : "MATCH (a:Tweet)-[r]-(b) WHERE a.longitude>= "+helpy1+" AND a.longitude<= "+helpy2+" AND a.latitude>= "+helpy3+" AND a.latitude<= "+helpy4+" AND b.id= "+selectedid+" RETURN a"

Γίνεται αναζήτηση για tweets που έχουν δημοσιευθεί από συντεταγμένες μέσα σε αυτό το τετράγωνο (δηλαδή πολύ κοντα στο σημείο που έχει κάνει κλικ ο χρήστης). Στο response επιστρέφονται τα tweets που πλοιρούν τα κριτήρια και εξάγονται οι πληροφορίες τους (id, περιεχόμενο, βάρος positive, βάρος negative) και χαρακτηρίζεται το tweet ως θετικό, αρνητικό ή ουδέτερο όπως και πιο πάνω. Οι πληροφορίες αυτές εισάγονται με html στη μεταβλητή contenttoput.Στο clusters.html έχει οριστεί συγκεκριμένο div στα δεξιά της σελίδας με id="tweetcontent" του οποίου το περιεχόμενο αλλάζει με την εντολή **tweetcontent.innerHTML = contenttoput**.

- Μέρος 4ο: Αφού έχει σχεδιαστεί ο χάρτης, θέτεται και η μεταβλητή **drawnmap = 1**.

Στιγμιότυπο οθόνης του clusters.html:



INDEX.HMTL

Χρησιμοποιώντας κώδικα παρόμοιο με του clusters.html, στη σελίδα index παρέχονται στο χρήστη οι εξής δυνατότητες:

1. Ο χρήστης μπορεί να εισαγάγει το ID, την πόλη και την ημερομηνία ενός cluster και να δε τους clusters των υπόλοιπων ημερών της εβδομάδας από την ίδια πόλη οι οποίοι παρουσιάζουν τη μεγαλύτερη ομοιότητα με τον cluster που πληκτρολογήθηκε. Το cypher query που στέλνεται με HTTP request στη βάση για να γίνει η αναζήτηση είναι:

```
var searchstatement = {
      "statements" : [ {
            "statement" : "MATCH (c:Cluster)-[r:SIMILAR_TO]->(b:Cluster)
where c.id= "+document.getElementById('clustersearchbox').value+" AND
c.date= "+document.getElementById('clusterdate').value.replace(/-/g, '')+"
AND c.city=\""+city+"\" return c,b"
      } ]
};
```

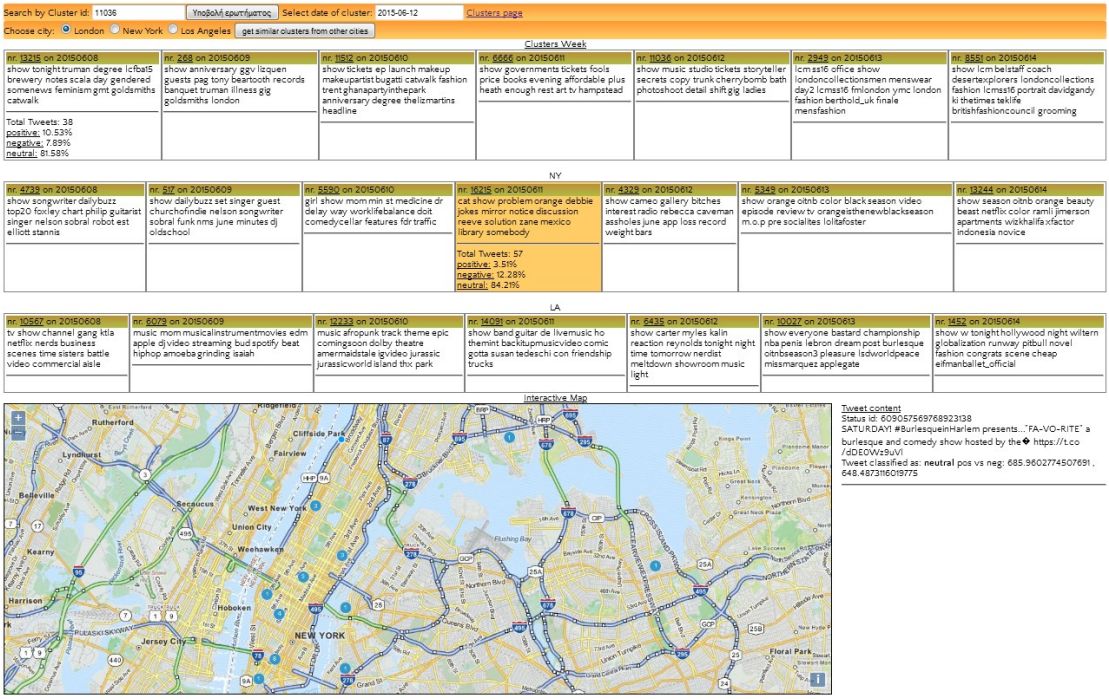
2. Ο χρήστης μπορεί να ζητήσει να δει τους clusters από άλλες πόλεις που μοιάζουν περισσότερο με τον cluster του οποίου τις πληροφορίες εισάγει. Τα cypher queries που στέλνονται με HTTP request στη βάση για να γίνει η αναζήτηση είναι:

```
var searchstatement = {
      "statements" : [ {
            "statement" : "MATCH (c:Cluster {city:\""+city+"\"})-
[r:OTHER_CITY]->(b:Cluster {city:\""+city1+"\"}) where c.id=
"+document.getElementById('clustersearchbox').value+" AND c.date=
"+document.getElementById('clusterdate').value.replace(/-/g, '')+" return
c,b"
      } ]
};
var searchstatement = {
      "statements" : [ {
            "statement" : "MATCH (c:Cluster {city:\""+city+"\"})-
[r:OTHER_CITY]->(b:Cluster {city:\""+city2+"\"}) where c.id=
"+document.getElementById('clustersearchbox').value+" AND c.date=
"+document.getElementById('clusterdate').value.replace(/-/g, '')+" return
c,b"
      } ]
};
```

3. Πατώντας πάνω σε ένα cluster εμφανίζονται πληροφορίες για το γενικό συναίσθημα και σχεδιάζονται τα tweets του στις συντεταγμένες του καθενός στο χάρτη με παρόμοιο τρόπο όπως στο clusters.html, με τη συνάρτηση showContent(id,date,cityz). Τα cypher queries που στέλνονται με HTTP request στη βάση για να γίνει η αναζήτηση είναι:

```
var searchstatement = {
      "statements" : [ {
            "statement" : "MATCH (a:Tweet)-[r]->(b:Cluster
{id:"+id+",date:"+date+",city:\""+cityz+"\"}) RETURN a"
      } ]
};
var singledata = {
      "statements" : [ {
            "statement" : "MATCH (a:Tweet)-[r]-(b) WHERE a.longitude>=
"+helpy1+" AND a.longitude<= "+helpy2+" AND a.latitude>= "+helpy3+" AND
a.latitude<= "+helpy4+" AND b.id= "+selectedid+" RETURN a"
            } ]
      };
```

Στιγμιότυπο οθόνης του index.html:



5.2 Πλατφόρμες και προγραμματιστικά εργαλεία

Για την ανάπτυξη των προγραμμάτων java χρησιμοποιήθηκε το περιβάλλον Eclipse.

Η web εφαρμογή αναπτύχθηκε με HTML,CSS, JAVASCRIPT. Είναι δηλαδή δύο ιστοσελίδες. Το Index.hmtl και το clusters.html. Τα αρχεία HTML CSS JS ανέβηκαν σε ένα server και είναι προσβάσιμα στις διευθύνσεις http://giorgis.eu/neo και http://giorgis.eu/neo/clusters.html . Η βάση δεδομένων τρέχει σαν service σε VM στο okeanos του ΕΔΕΤ.

6 Έλεγχος

Θα ακολουθήσει η αξιολόγηση του συστήματος.

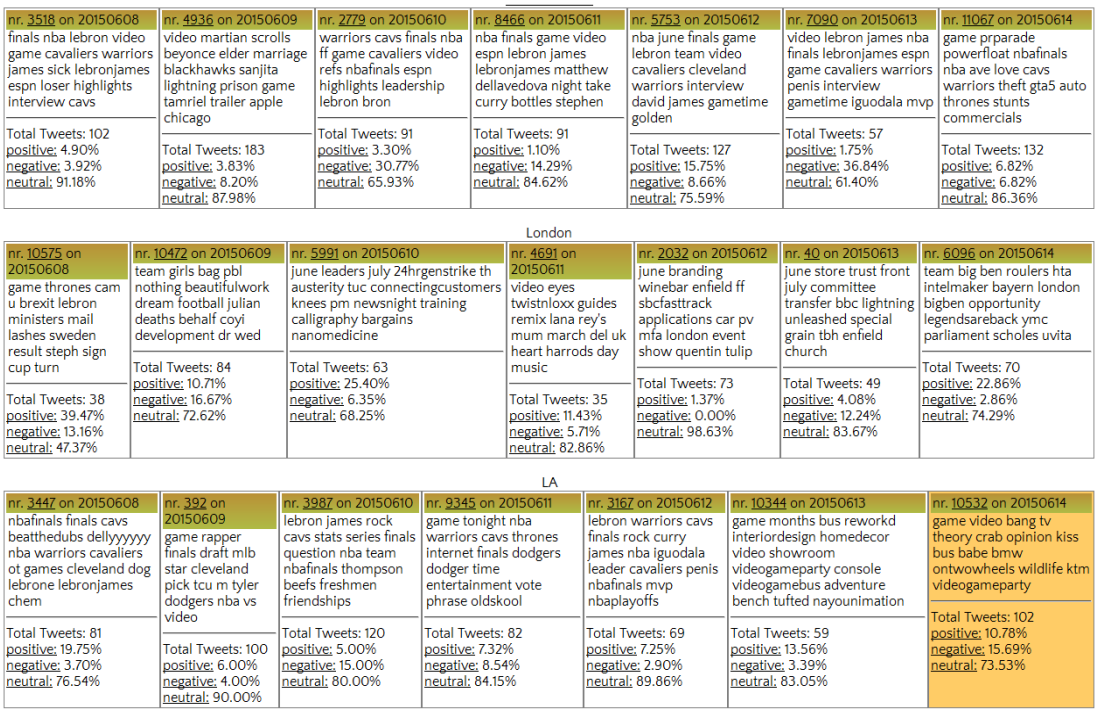
6.1 Μεθοδολογία ελέγχου

Για τον έλεγχο του συστήματος έγινε προσπάθεια να βρεθούν σενάρια της πραγματικότητας τα οποία έχουν αποτυπωθεί και στα δεδομένα που έχουν αποθηκευτεί στη βάση. Με άλλα λόγια εξετάζουμε αν αποτελέσματα που παρουσιάζει η εφαρμογή μας, έχουν λογική και ανταποκρίνονται σε πραγματικά γεγονότα / θέματα.

6.2 Αναλυτική παρουσίαση ελέγχου

Βρέθηκαν τα ακόλουθα σενάρια:

- Σενάριο 1 – Λονδίνο: Στο clusters.html κάνουμε αναζήτηση με τη λέξη entourage. Στα δεξιά επιστρέφονται οι clusters που περιέχουν τη λέξη στους σημαντικότερους όρους τους (top terms). Παρατηρούμε ότι ο δεύτερος περιέχει τα εξής top terms: *entouragemovie film premiere stars year morning's five european week technology movietheme bluecarpet*. Και ο τρίτος περιέχει τα εξής: *friend square leicester movie premier entourage chelsea football league londonroadfilm vail furious*. Επιλέγουμε clusters που περιέχουν τη λέξη που ψάχνουμε όσο πιο "μπροστά" γίνεται. Όπως έχει ειπωθεί, όσο πιο πάνω είναι μια λέξη στα top terms, τόσο πιο σημαντική είναι για τον cluster. Επιλέγεται ο 3ος cluster, παίρνουμε τις 6 πρώτες λέξεις του (*friend square leicester movie premier entourage)* και τις εισάγουμε στην αναζήτηση του google. Αμέσως επιστρέφεται ως αποτέλεσμα άρθρο για την πρεμιέρα της ταινίας entourage στο Leicester square του Λονδίνου (http://markmeets.com/past-london-film-premiere/ - **Entourage** UK film premiere was at the Vue West end cinema, Leicester Square, London on Tuesday 9th June 2015. Jeremy Piven, Adrian Grenier, Jerry Ferrara, Kevin Connolly & Kevin Dillon attended.) Κοιτώντας την ημερομηνία του cluster, παρατηρούμε ότι ανήκει πράγματη στην 9η Ιουνίου 2015. Άρα τα αποτελέσματα της εργασίας συμβαδίζουν με την πραγματικότητα.
- Σενάριο 2 – Νέα Υόρκη: Στο clusters.html κάνουμε αναζήτηση με τη λέξη nba. Στα δεξιά επιστρέφονται οι clusters που περιέχουν τη λέξη στα top terms τους. Παρατηρούμε ότι ο 4ος περιέχει τα εξής top terms: *nba june finals game lebron team video cavaliers cleveland warriors interview david*. Με μια αναζήτηση στο google καταλαβαίνουμε ότι πράγματι την εβδομάδα που εξετάζουμε έγιναν οι τελικοί του **nba** μεταξύ των ομάδων **Cleveland Cavalier**s και Golden State **Warriors**. Το μεγάλο αστέρι των cavaliers ήταν ο Lebron **James**. Επιλέγεται λοιπόν ο 4ος cluster και πάμε στο index.html και εισάγουμε το id του (5753), την πόλη (NY) και την ημερομηνία του (12-06-2015). Παίρνουμε τους clusters που μοιάζουν περισσότερο με αυτόν για κάθε ημέρα και κάθε πόλη:



Παρατηρούμε ότι για Νέα Υόρκη και Los Angeles η εφαρμογή επέστρεψε clusters που πράγματι μιλούν για τους τελικούς του NBA. Εξαίρεση αποτελεί ο cluster 4936 της 09-06-2015. Για τη μέρα αυτή η εφαρμογή μας δεν βρήκε cluster ικανοποιητικά όμοιο με τον αρχικό. Παρατηρούμε επίσης ότι οι clusters που εμφανίζονται από το Λονδίνο μιλούν για **June, game, lebron, team, event** αλλά οι εμφανίσεις αυτών των λέξεων είναι ελάχιστες και όχι σημαντικές. Μπορούμε να συμπεράνουμε ότι στο Λονδίνο οι χρήστες του facebook δεν ενδιαφέρθηκαν ιδιαίτερα για τους τελικούς του NBA.

- Σενάριο 3 – Los Angeles: Στο clusters.html γίνεται η αναζήτηση της λέξης movie. Στα δεξιά εμφανίζονται διάφοροι clusters, πολλοί με γενικό περιεχόμενο όπως ο cluster 745 που περιέχει λέξεις όπως *night saturday house hollywood date fun pasadena movie*. Ένας τέτοιος cluster περιμένουμε να έχει θετικό συναίσθημα, καθώς αναφέρεται σε διασκέδαση. Πράγματι πατώντας πάνω του βλέπουμε τα στοιχεία:

  Total Tweets: 204

  positive: 17.647058823529413%

  negative: 1.9607843137254901%

  neutral/irrelevant: 80.3921568627451%

  Άρα το συναίσθημα του 745 μπορεί να χαρακτηριστεί θετικό.

  Αμέσως κάτω από τον 745 εμφανίζεται ο 2320 με ημερομηνία 10-06-2015 και όρους: *world jurassic park jurassicworld premiere dolby worldpremiere years chrispratt theatre g movie*. Με αναζήτηση στο google βλέπουμε ότι πράγματι στις 9 **Ιουνίου** έγινε στο **Los Angeles** η επίσημη **πρεμιέρα** της **ταινίας jurassic world** στην οποία πρωταγωνιστεί ο ηθοποιός **Chris Pratt**. Λογικό η συζήτηση να γίνει την 10η Ιουνίου – το πρωί μετά την πρεμιέρα, οπότε πιθανότατα τα ειδησεογραφικά sites έγραψαν για το γεγονός στις κοσμικές στήλες. Υπενθυμίζεται ότι καινούργια μέρα ξεκινάει για την εφαρμογή μας τα μεσάνυχτα.

Εισάγοντας τα στοιχεία του 2320 στο index.html ψάχνουμε να δούμε με ποιους clusters άλλων ημερών της ίδιας πόλης μοιάζει ο 2320. Παίρνουμε τα εξής αποτελέσματα:

nr. 12293 on 20150608	nr. 3697 on 20150609	nr. 2320 on 20150610	nr. 11175 on 20150611	nr. 4818 on 20150612	nr. 3043 on 20150613	nr. 15144 on 20150614
world cup course person day someone happiness lighteth john match usa lbsu aus fifawwc light	jurassic park	world jurassic park jurassicworld premiere dolby worldpremiere years chrispratt theatre g movie mothafuckaaaas labcoats wagons	park jurassic griffith memorial highland echo neighbors golf runyon basketball kanye court play lake kim	world jurassic 3d imax ca amc grove theatres pacific tcl angeles experience los cinemas things	jurassic world park cinemas arclight ca cinemark los movie angeles matinee seats vino pratt amc	world jurassic nintendo ca angeles championships los championship advertisement banner highland cinerama amc dome hollywood

Η ορθότητα των αποτελεσμάτων επαληθεύεται από το ότι στις 8 και 9 Ιουνίου (Δευτέρα και Τρίτη) δεν έχουμε συζητήσεις για το jurassic world αφού η ταινία δεν έχει κάνει ακόμα πρεμιέρα. Την Τετάρτη όμως υπάρχει ένας cluster που μιλά για την πρεμιέρα (η οποία όπως είπαμε έγινε στις 9 Ιουνίου). Παρατηρούμε ότι δεν υπάρχει η λέξη **premiere** σε cluster άλλης ημέρας. Στις 11 Ιουνίου βλέπουμε μόνο τις λέξεις **jurassic** και **park** που σημαίνει ότι δεν ήταν δημοφιλές θέμα η ταινία που εξετάζουμε. Από την Παρασκευή και μετά όμως έχουμε clusters με πολλά terms για **cinema, 3d, imax, jurassic, world** και άλλα... Αυτό συμβαίνει γιατί η πρεμιέρα της ταινίας για το κοινό έγινε στις ΗΠΑ την Παρασκευή 12 Ιουνίου.

Στη συνέχεια ψάχνουμε να βρούμε clusters όμοιους με τον 2320 από άλλες πόλεις. Παίρνουμε τα εξής αποτελέσματα:

nr. 11077 on 20150608	nr. 6844 on 20150609	nr. 9967 on 20150610	nr. 9080 on 20150611	nr. 11621 on 20150612	nr. 3891 on 20150613	nr. 5162 on 20150614
park hyde london victoria heaton olympic battersea hill regent's limmo tonnes holland parklife peninsula farringdon	world months cup engvnz follow lots standard stitch blessings difference zealand brand yorkshire tea eniolabadmus	park hyde london richmond victoria chiswick londonlife brockwell regents battersea nofilter olympic stairs tech greenwich	jurassic world jurassicworld cineworld minutes fire dinosaurs greater london park greenwich vue asap floyd must	jurassic world imax cineworld 3d west greater london park quay experience bfi waterloo cricklewood movies	world jurassic abbey baby cup westminster star london sis caves cineworld greater chislehurst imax arch	jurassic world chris imax pratt bullshit greater puppy london cinema dinosaur robinson isla owen bfi

NY

nr. 3254 on 20150608	nr. 11892 on 20150609	nr. 3541 on 20150610	nr. 2974 on 20150611	nr. 9284 on 20150612	nr. 7978 on 20150613	nr. 15013 on 20150614
world trade center one observatory newyork oneworldtradecenter nyc lot view ny new york floor set	central park centralpark nyc zoo ny brews tigers lions photo lake bryant manhattan york event	world trade center one luck nyc m's m oneworldtradecenter york new photo oneworldobservatory skyline iloveny	world trade center war raf crimes intelligence jurassic christopher lee amc loews imax nyc photo	jurassic world amc loews ny new 3d york cinemas union square imax regal lincoln experience	jurassic world amc bakery loews cookie ny dinosaurs 3d imax cinemas lego new york experience	jurassic world cinemas san juan regal non jurassicpark walk jurassicworld review movie ny startup francisco

Με αναζήτηση στο imdb.com βρίσκουμε ότι η πρεμιέρα της ταινίας για το κοινό στο Ηνωμένο Βασίλειο έγινε στις 11 Ιουνίου. Γι αυτό μέχρι και τις 10 Ιουνίου δεν έχουμε στους clusters του Λονδίνου κάποια αναφορά στην ταινία.

Στη Νέα Υόρκη αντίστοιχα ο κόσμος αρχίζει να μιλά για την ταινία την Παρασκευή 12 Ιουνίου, αφού τότε ήταν η πρεμιέρα στις ΗΠΑ.

7 Επίλογος

7.1 Σύνοψη και συμπεράσματα

Όπως φάνηκε και στο προηγούμενο κεφάλαιο, τα αποτελέσματα της επεξεργασίας των δεδομένων που παρουσιάζει η web εφαρμογή μας μπορούν πράγματι να αντιστοιχούν σε πραγματικές συζητήσεις και συναισθήματα. Πραγματοποιώντας μια αναζήτηση για μια λέξη ο χρήστης μπορεί να λάβει πληροφορίες για διάφορα θέματα συζήτησης που έχουν σχέση με αυτή τη λέξη (θέμα = cluster), να δει πώς μεταβάλλονται στο χρόνο σε περίοδο μιας εβδομάδας και στο χώρο - σε διαφορετικές πόλεις. Πολλές φορές μπορεί ένας cluster να περιέχει πάνω από ένα θέματα. Μένει στο χρήστη να μπορεί να ξεχωρίσει τα διαφορετικά θέματα (όπου υπάρχουν) εξετάζοντας τους σημαντικότερους όρους του cluster κατα προτεραιότητα. Παρατηρείται ότι χρήστες σε διαφορετικές πόλεις της ίδιας χώρας μπορεί να συζητούν για ίδια θέματα την ίδια μέρα, ή και για διαφορετικά θέματα. Η διαφοροποίηση είναι ακόμα πιο εμφανής όταν εξετάζουμε πόλεις διαφορετικών χωρών. Θέματα που ενδιαφέρουν μια πόλη ή χώρα μπορεί να μην ενδιαφέρουν κάποια άλλη. Ή κάποια πόλη ή χώρα να επηρεάζεται από ένα γεγονός (όπως είναι ακόμα και η πρεμιέρα μιας ταινίας) μεταγενέστερα από μια άλλη. Όλα αυτά είναι εμφανή στα αποτελέσματά μας.

Ο χρήστης έχει επίσης τη δυνατότητα να εξετάσει το συναίσθημα των ανθρώπων (χρηστών του twitter) σε συγκεκριμένες πόλεις και μέρες για ένα θέμα. Η συναισθηματική ανάλυση είναι δύσκολο να γίνεται πάντα επιτυχώς καθώς κάθε άνθρωπος μπορεί να σχολιάζει, να χρησιμοποιεί smileys και τις καθημερινές εκφράσεις με σκοπό την ειρωνεία, το χλευασμό, την επιδοκιμασία κ.ά. Είναι πολύ δύσκολο για έναν αλγόριθμο να ταξινομήσει σωστά ένα tweet σε θετικό ή αρνητικό όταν οι ίδιες λέξεις μπορούν να χρησιμοποιηθούν και θετικά και αρνητικά.

7.2 Μελλοντικές επεκτάσεις

Οι δυνατότητες για επέκταση της διπλωματικής είναι πολλές.

- Θα μπορούσαν να λαμβάνονται υπ όψιν κατηγορικά χαρακτηριστικά όπως το λειτουργικό σύστημα ή το είδος της συσκευής από την οποία γίνεται μια δημοσίευση. Έτσι θα υπήρχε η δυνατότητα να μελετηθούν τα χαρακτηριστικά αυτά και η αποδοχή τους από το καταναλωτικό κοινό εξετάζοντας τον αριθμό των εμφανίσεών τους και το συναίσθημα των δημοσιεύσεων. Επιπροσθέτως, θα μπορούσαν να συσχετιστούν πληροφορίες όπως το μοντέλο μιας συσκευής (το οποίο συχνά συνδέεται με την οικονομική ευχέρεια και τον τρόπο ζωής ενός χρήστη) με τα θέματα για τα οποία μιλάει.
- Στο μέλλον θα μπορούσε να αυτοματοποιηθεί η όλη διαδικασία συλλογής και επεξεργασίας των tweets και η εισαγωγή τους στη βάση με τελικό στόχο η βάση να περιέχει σήμερα τα επεξεργασμένα tweets της χτεσινής ημέρας.

8 Βιβλιογραφία

[1] Alec Go, Richa Bhayani, Lei Huang. 2009. Twitter Sentiment Classification using Distant Supervision. Technical report, Stanford.

[2] Anders Skovsgaard, Darius Sidlauskas, Christian S. Jensen. 2014. Scalable Top-k Spatio-Temporal Term Querying. In Data Engineering (ICDE), 2014 IEEE 30th International Conference on March 31 2014-April 4 2014.

[3] Apoorv Agarwal, Jasneet Singh Sabharwal. 2012. End-to-End Sentiment Analysis of Twitter Data. In COLING 2012

[4] Ceren Budak Theodore Georgiou Divyakant Agrawal Amr El Abbadi. 2014. GeoScope: Online Detection of GeoCorrelated Information Trends in Social Networks. In proceedings of the International Conference on Very Large Data Bases, VLDB , 2014

[5] Hamed Abdelhaq, Christian Sengstock, and Michael Gertz. 2013. EvenTweet: Online Localized Event Detection from Twitter. In Proceedings of the International Conference on Very Large Data Bases, VLDB , 2013

[6] Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, David R. Karger. 2003. Tackling the Poor Assumptions of Naive Bayes Text Classifiers. In ICML 2003

[7] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith. 2011. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In Proc. of ACL.

[8] Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alex Smola, Kostas Tsioutsiouliklis. 2012. Discovering Geographical Topics In The Twitter Stream.In WWW 2012

[9] Michael Mathioudakis, Nick Koudas. 2010. TwitterMonitor: Trend Detection over the Twitter Stream. In SIGMOD Conference 2010

[10] Thanaa M. Ghanem, Amr Magdy, Mashaal Musleh, Sohaib Ghani, Mohamed F. Mokbel. 2014. VisCAT: Spatio-Temporal Visualization and Aggregation of Categorical Attributes in Twitter Data. In SIGSPATIAL/GIS 2014

[11] Zhijun Yin, Liangliang Cao, Jiawei Han, Chengxiang Zhai, Thomas Huang. 2011. Geographical Topic Discovery and Comparison. In WWW 2011.

[12] https://mahout.apache.org/

[13] http://twitter4j.org

[14] http://neo4j.org

[15] http://www.ark.cs.cmu.edu/TweetNLP/

[16] https://dev.twitter.com/