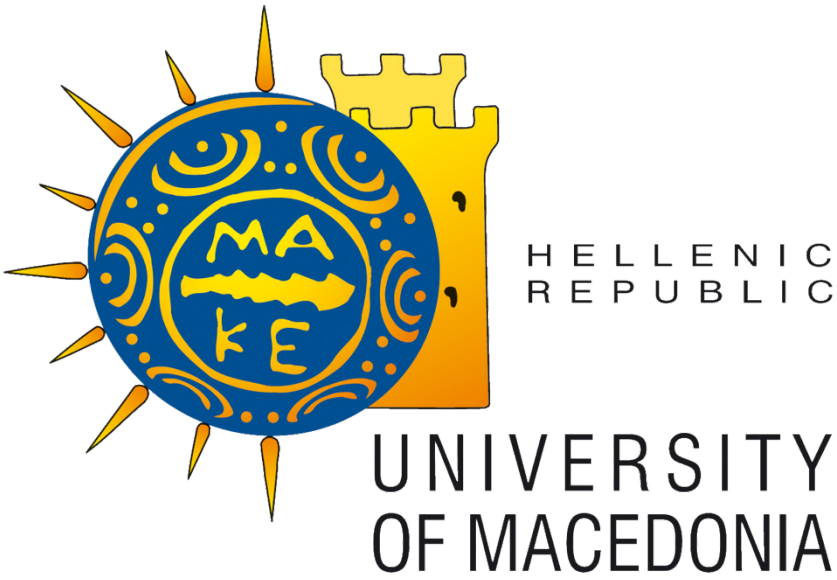
ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ ΠΛΗΡΟΦΟΡΙΑΣ
ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ



ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΘΕΜΑ
Συναισθηματική ανάλυση σε twitter post με χρήση στοίβας Elastic

Φοιτητής :
Καραμούλας Ελευθέριος – Α.Μ.1437

Επιβλέπον καθηγητής :
Ευαγγελίδης Γεώργιος

ΘΕΣΣΑΛΟΝΙΚΗ 2017-2018

Περίληψη

Στην εποχή μας τα δεδομένα που προέρχονται από τα μέσα κοινωνικής δικτύωσης μπορούν να κρύβουν σημαντικές πληροφορίες για διάφορα θέματα και ζητήματα. Αυτές μπορούν να σχετίζονται με πολιτικά θέματα και θέματα που αφορούν όλο τον κόσμο καθώς και γνώμες χρηστών για κάποιο προϊόν. Τα δεδομένα αυτά μετά από κατάλληλη προ επεξεργασία τόσο γλωσσολογική όσο και συναισθηματική μπορούν να αξιοποιηθούν κατάλληλα. Είναι πολύ πιο εύκολο και εύχρηστο να ληφθούν αποφάσεις προώθησης, ανάπτυξης και βελτίωσης των προϊόντων/υπηρεσιών αφού ληφθούν υπόψιν οι συσχετιζόμενες πληροφορίες για το συγκεκριμένο θέμα. Ο στόχος της πτυχιακής αυτής είναι η παρουσίαση των δυνατών επεξεργασιών στα διαθέσιμα δεδομένα. Καθώς και την σύγκριση των διάφορων μεθοδολογιών και των αποτελεσμάτων που η κάθε μια παρέχει. Όπως επίσης οι πληροφορίες που μπορούν να συλλεχθούν από αυτά τα δεδομένα και συμπεράσματα στα οποία μπορούμε να καταλήξουμε αν τις λάβουμε υπόψιν.

Λέξεις κλειδιά: big data, μέσα κοινωνικής δικτύωσης, python, NLTK, Scikit-learn, γλωσσική επεξεργασία, συναισθηματική ανάλυση, κατηγοριοποίηση, twitter, elk, elastic, logstash, elasticsearch, kibana

Abstract

In our day data coming from social media can hide important information on various topics and issues. These can be related to political and world-related issues around the globe as well as user opinions on a product. These data, after appropriate pre-processing, both linguistically and emotionally, can be appropriately exploited. It is much easier and handy to make decisions to promote, develop and improve products / services, taking into account the related information on the subject. The aim of this thesis is to present the possible processing on the available data as well as to compare the various methodologies and the results each provide. As well as the information that can be gathered from these data and the conclusions that we can reach if we take them into account.

Keywords: big data, social media, python, NLTK, Scikit-learn, natural language processing, sentiment analysis, classification, twitter, elk, elastic, logstash, elasticsearch, kibana

Περιεχόμενα

1.Εισαγωγή

1.1 Γενική Εισαγωγή

Όταν αναφερόμαστε σε ηλεκτρονικά δεδομένα τι ακριβώς εννοούμε και γιατί τα δεδομένα είναι τόσο σημαντικά για μικρές και μεγάλες εταιρίες στην εποχή μας; Πως μπορεί να αντλήσει κάποιος πληροφορίες από αυτά τα δεδομένα και πως μπορεί να τις χρησιμοποιήσει προς όφελος του; Τι είναι τα big data, ένα από τα πιο επίκαιρα θέματα των τελευταίων 2 ετών, για το οποίο μιλάνε όλοι; Data science και data analysis μια από τις πιο περιζήτητες και ακριβοπληρωμένες ειδικότητες των τελευταίων ετών. Αυτά τα ερωτήματα και αλλά σχετικά θέματα θα καλύψω στο 2ο κεφάλαιο της πτυχιακής. Στην συνέχεια στο τεχνικό κομμάτι θα περιγράψω αρχικά την προετοιμασία και το στήσιμο των τεχνολογιών που χρησιμοποιήθηκαν. Ακολουθεί η ροή εργασίας, η επεξεργασία και η οπτικοποίηση των δεδομένων. Τέλος θα παρουσιαστούν γενικά συμπεράσματα και ο σχολιασμός των αποτελεσμάτων.

1.2 Στόχοι της πτυχιακής

Το θέμα αυτής της πτυχιακής αναφέρεται σε πολλούς τομείς με μεγάλο υπόβαθρο. Επομένως είναι δύσκολο να αναλυθεί σε βάθος κάθε πτυχή και τεχνολογία που εμπλέκεται σε αυτή. Ωστόσο οι στόχοι είναι να γίνει μια πρώτη επαφή και εξοικείωση με τις κυριότερες έννοιες και τεχνολογίες. Όπως για παράδειγμα τι είναι τα big data, από που προέρχονται και αν τα εκμεταλλευθούμε κατάλληλα τι δυνατότητες και οφέλη μπορούμε να κερδίσουμε. Καθώς επίσης έννοιες όπως Natural Language Processing και sentiment analysis. Οι διάφορες δυνατότητες και συνδυασμοί που παρέχουν αν εφαρμοστούν κατάλληλα στα δεδομένα μας. Καθώς και η γνωριμία του αναγνώστη με τεχνολογίες όπως το twitter API και το ELK stack (Elasticsearch, Logstash, Kibana) της elastic. Όπως επίσης και βιβλιοθήκες της python οι NLTK και Scikit-learn που εξειδικεύονται στους τομείς της γλωσσικής επεξεργασίας και μηχανικής μάθησης αντίστοιχα. Όλα τα προαναφερθέντα θα καλυφθούν στα παρακάτω κεφάλαια μέχρι έναν βαθμό ώστε να αποκτήσει ο αναγνώστης μια θεωρητική καθώς και πιο πρακτική ιδέα επί του θέματος.

2.Δεδομένα, μεθοδολογίες και νέες τεχνολογίες

2.1Τα δεδομένα στην εποχή μας

Ας αρχίσουμε από την απλή ερώτηση «τι είναι τα δεδομένα;». Τα δεδομένα στην γενική τους μορφή θα μπορούσαν να χαρακτηριστούν ως πληροφορίες. Πιο συγκεκριμένα σαν γεγονότα ή αριθμοί τα οποία αποτελούνται από ποιοτικές και ποσοτικές μεταβλητές κάτι το οποίο υποστηρίζει την προηγουμένη παρατήρηση. Όταν μιλάμε για ποιοτικά δεδομένα αναφερόμαστε σε δεδομένα τα οποία προέκυψαν από παρατηρήσεις. Συνήθως πάνω σε ένα συγκεκριμένο θέμα ή από μια έρευνα που ως στόχο είχε να καταλήξει σε κάποια πιθανά αποτελέσματα. Τα ποσοτικά από την άλλη είναι πιο εύκολα να κατανοηθούν καθώς δηλώνουν ακριβώς ότι δείχνουν. Όπως αριθμοί από καταγραφές πωλήσεων, στατιστικά για τις αποδόσεις μιας εταιρείας μετά από την εφαρμογή μιας νέας πολιτικής ή κάποιου προωθητικού προγράμματος κλπ.

Η μέτρηση, συλλογή και ανάλυση των δεδομένων γίνεται για την ευκολότερη και αποτελεσματικότερη αποθήκευση, ανάκτηση και χρήση αυτών. Τα δεδομένα γίνονται κατάλληλα και αποτελεσματικά για την λήψη αποφάσεων εφόσον υποστούν κάποιου είδους επεξεργασία. Το άλλο πλεονέκτημα των δεδομένων και ταυτόχρονα ο λόγος που κάνει την επεξεργασία τους δύσκολη είναι ότι τα δεδομένα μπορούν να διαχωριστούν αρχικά ανάλογα με την μορφή τους σε ηλεκτρονικά ή μη. Καθώς είναι φυσικό η δεύτερη κατηγορία είναι κατανοήσιμη μόνο από τον άνθρωπο. Τα δεδομένα της πρώτης κατηγορίας ωστόσο μπορούν να είναι διαθέσιμα σε διαφορετικές καταστάσεις όπως για παράδειγμα δομημένα ή αδόμητα. Στην πρώτη κατηγορία έχουμε δεδομένα τα οποία είναι κατανοητά από ηλεκτρονικούς υπολογιστές. Ενώ στην δεύτερη κατηγορία έχουμε δεδομένα των οποίων η αξία είναι κατανοήσιμη μόνο από τον άνθρωπο. Τα δεδομένα σε μεγάλο όγκο και όταν θέλουμε ταχύτατη επεξεργασία και εξόρυξη πληροφορίας είναι κάτι το οποίο μπορεί να επιτευχθεί μόνο από μια μηχανή. Επομένως τα δεδομένα τα οποία έχουν αξία τελικά είναι αυτά που μπορεί να κατανοήσει ένας υπολογιστής. Ένας άλλος δυνατός διαχωρισμός των δεδομένων που καθιστά την επεξεργασία τους δύσκολη για έναν ηλεκτρονικό υπολογιστή καθώς απαιτείται διαφορετική προσέγγιση για κάθε κατηγορία είναι ανάλογα με τον τύπο τους. Όπως για παράδειγμα κείμενο, αριθμοί, εικόνες, ήχος κλπ. Αυτό έχει ως αποτέλεσμα ένας από τους κύριους στόχους της επιστήμης των δεδομένων να είναι η απαλοιφή αυτού του περιορισμού. Σε όσο το δυνατόν μεγαλύτερο βαθμό γίνεται ώστε όλο και περισσότερα δεδομένα να είναι διαθέσιμα προς επεξεργασία από ηλεκτρονικούς υπολογιστές.

Τα δεδομένα σε οποιαδήποτε μορφή μπορούν να επηρεάσουν διάφορες πτυχές του κόσμου και της καθημερινότητας μας. Για παράδειγμα τα τελευταία χρόνια τομείς όπως οι αγορές, οι διαφημίσεις και ο τρόπος λειτουργίας των εταιρειών έχει τροποποιηθεί και προσαρμοστεί με βάση τις συνήθειες των καταναλωτών και τις προτιμήσεις τους. Μεγάλες επιχειρήσεις συλλέγουν από τους καταναλωτές και επισκέπτες τους δεδομένα και στην συνέχεια με ανάλυση και εξόρυξη πληροφορίας ανακαλύπτουν προϊόντα τα οποία προτιμούνε περισσότερο, συνήθειες σε συνάρτηση με χρονικές περιόδους κα. Ωστόσο τα δεδομένα μπορούν να επηρεάσουν και πιο σημαντικούς τομείς όπως την εκπαίδευση και ζητήματα όπως η πολιτική και συσχετισμένα θέματα όπως οι πόλεμοι.

Το επόμενο ερώτημα που μπορούμε να κάνουμε για τα δεδομένα είναι «από που προέρχονται όλα αυτά τα δεδομένα;». Η απάντηση δεν μπορεί να είναι συγκεκριμένη και περιεκτική. Καθώς στην εποχή μας τα δεδομένα εκτός από διάφορες μορφές προέρχονται και από πολλές διαφορετικές πηγές. Βέβαια όταν αναφερόμαστε σε ηλεκτρονικά δεδομένα τα δεδομένα μπορούν να προέρχονται από πηγές που σχετίζονται με τους ηλεκτρονικούς υπολογιστές και τις υπηρεσίες που αυτοί φιλοξενούν και παρέχουν. Για παράδειγμα τα μέσα κοινωνικής δικτύωσης διαχειρίζονται τεράστια μεγέθη δεδομένων καθημερινά τα οποία

παράγονται από τους χρήστες τους. Εκ των οποίων άλλα είναι δημόσια, τα οποία ο χρήστης μοιράζεται με τους υπόλοιπους και άλλα είναι προσωπικά. Τα προσωπικά συνήθως παρέχονται στην υπηρεσία κατά την διαδικασία εγγραφής για παράδειγμα. Επίσης οι συναλλαγές που γίνονται μέσω ιστού καθημερινά αποτελούν τεράστιο ποσοστό των δεδομένων που παράγονται καθημερινά. Ωστόσο τι θεωρείτε ως συναλλαγή; Ως συναλλαγή ορίζεται οποιαδήποτε ενέργεια η οποία απαιτεί μια ενέργεια για συλλογή δεδομένων. Για παράδειγμα οι ηλεκτρονικές αγορές, η πλοήγηση μας στην ιστοσελίδα μιας εταιρίας επίσης αποτελεί μια συναλλαγή, οι διαφημίσεις που βλέπουμε στις ιστοσελίδες κα. Πιο γενικευμένο αλλά αποτελεί εξίσου μεγάλο τμήμα των δεδομένων είναι όλες οι ιστοσελίδες του ιστού που στεγάζουν πληροφορίες. Όπως για παράδειγμα ιστοσελίδες πληροφόρησης με άρθρα και βίντεο ή κυβερνητικές ιστοσελίδες με αποτελέσματα από καταγραφές, πληροφορίες για διάφορα κοινωνικά ζητήματα κα. Ένας νέος τομέας είναι τα δεδομένα που προέρχονται από αισθητήρες στους οποίους πολλές φορές αναφερόμαστε και ως IoT (Internet of Things).Τα συγκεκριμένα δεδομένα μπορούν να προέρχονται από καταγραφές καιρικών συνθηκών σε διάφορα σημεία του πλανήτη για πρόβλεψη ακραίων και μη καιρικών συνθηκών. Άλλο ένα παράδειγμα είναι τα έξυπνα συστήματα τα οποία εγκαθιστούμε στα σπίτια μας για τον αυτόματο έλεγχο διάφορων παροχών όπως ρεύμα, θέρμανση κα. Μέχρι και μικρότερες συσκευές όπως τα βραχιόλια χειρός. Τα οποία χρησιμοποιούνται όλο και από περισσότερο κόσμο για την καταγραφή και ανάλυση των καθημερινών φυσικών/σωματικών συνηθειών τους. Όπως η εκγύμναση και η ποιότητα του ύπνου.

Όλα τα παραπάνω συμβάλουν στην παραγωγή των big data. Ωστόσο δεν υπάρχει μια συγκεκριμένη τιμή μεγέθους μετά την οποία τα δεδομένα από «κανονικά» γίνονται big. Ο όρος big data θέλει κυρίως να τονίσει το μέγεθος των δεδομένων και το πλήθος των διαφορετικών τύπων που υπάρχουν. Στην ουσία η διαφορά είναι στον τρόπο που προσαρμόζουμε τα διάφορα εργαλεία να μπορούν να ανταπεξέλθουν στο μέγεθος και την πολυπλοκότητα για την συλλογή, αποθήκευση και ανάλυση των δεδομένων. Περισσότερα για τα big data στην επόμενη ενότητα.

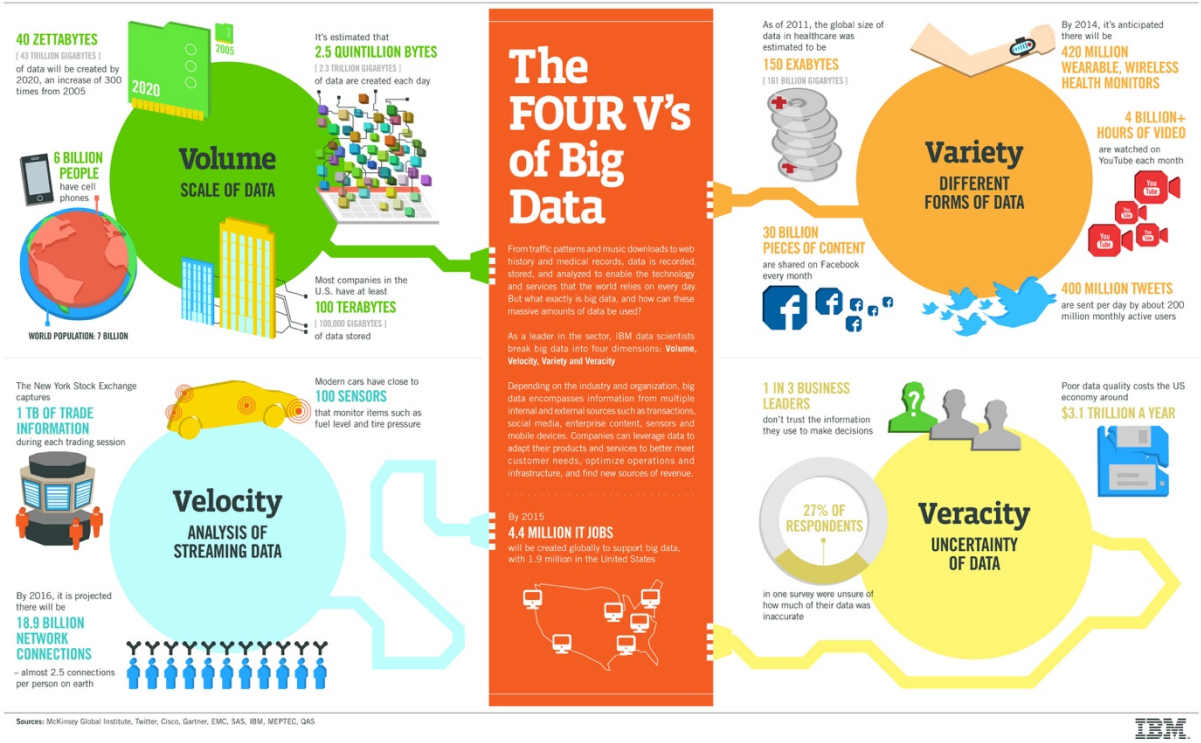
2.2Τι είναι το big data

Όπως προαναφέρθηκε big data είναι ο ορισμός που περιγράφει το τεράστιο μέγεθος των δεδομένων (δομημένων και μη) που παράγεται καθημερινά και κατακλύζει τον ηλεκτρονικό κόσμο. Ωστόσο σημασία δεν έχει το μέγεθος των δεδομένων αλλά η προστιθέμενη αξία. Τι πληροφορίες κερδίζει μια εταιρεία από αυτά τα δεδομένα για να βελτιώσει τις αποφάσεις και στρατηγικές/πολιτικές που ακολουθεί. Ενώ ο όρος big data είναι σχετικά νέος η διαδικασία συλλογής και ανάλυσης μεγάλων μεγεθών δεδομένων για τελική ανάλυση είναι κάτι που γίνεται για πολύ καιρό τώρα. Η έννοια αυτή κέρδισε το ενδιαφέρον στις αρχές του 2000 όταν διατυπωθήκαν τα χαρακτηριστικά τέσσερα V΄s (Volume, Velocity, Variety, Variability/ Veracity) των big data.

Όγκος (Volume): Η ποσότητα των παραγόμενων και αποθηκευμένων δεδομένων. Το μέγεθος των δεδομένων καθορίζει αν μπορούν να θεωρηθούν big data. Καθώς και η αξία και πιθανή απόκτηση διορατικότητας από αυτά. Ενώ η ποσότητα υποδηλώνει περισσότερα δεδομένα αυτό που τα καθιστά μοναδικά είναι η λεπτομερειακότητα τους. Οι οργανισμοί συλλέγουν δεδομένα από διάφορες πηγές όπως για παράδειγμα επιχειρηματικές συναλλαγές, μέσα κοινωνικής δικτύωσης, πληροφορίες από αισθητήρες και δεδομένα που μεταφέρονται από υπηρεσίες. Στο παρελθόν η αποθήκευση δεδομένων τέτοιου μεγέθους θα αποτελούσε πρόβλημα αλλά με τις νέες τεχνολογίες αυτά τα προβλήματα απαλείφονται όλο και σε μεγαλύτερο βαθμό.

Ταχύτητα (Velocity): Η ταχύτητα με την οποία παράγονται και επεξεργάζονται τα δεδομένα για να φτάσουν τις απαιτήσεις και προκλήσεις στην πορεία ανάπτυξης και παραγωγής τους. Υπάρχουν διάφοροι τρόποι επεξεργασίας των δεδομένων ανάλογα με την ταχύτητα

παραγωγής τους. Επίσης υπάρχουν εφαρμογές των οποίων τα δεδομένα πρέπει να υποστούν μετατροπές και να εκτιμηθούν πριν επιστραφούν στην πηγή προέλευσης τους.
Ποικιλία (Variety): Τα δεδομένα έρχονται σε διάφορες μορφές. Από δομημένα, αριθμητικά δεδομένα και δεδομένα σε βάσεις δεδομένων μέχρι αδόμητα δεδομένα όπως κείμενα, email, βίντεο, αρχεία ήχου και συναλλαγές. Τα αδόμητα και ημιδομημένα δεδομένα απαιτούν μεταδεδομένα για υποστήριξη και επιπλέον επεξεργασία για να αποδώσουν αξία. Μόλις επιτευχθεί αυτή η διεργασία τα δεδομένα αποκτούν ίδιες απαιτήσεις με τα δομημένα. Περαιτέρω πολυπλοκότητα προκύπτει όταν μια πηγή τροποποιεί την μορφή των δεδομένων που αυτή παράγει. Συχνές ή αλλαγές σε πραγματικό χρόνο στις μορφές δημιουργούν τεράστιες επιβαρύνσεις για τα συστήματα συναλλαγών και αναλύσεων των δεδομένων.
Μεταβλητότητα (Variability/Veracity): Η ποιότητα των συλλεγμένων δεδομένων μπορεί να διαφέρει σημαντικά και ως αποτέλεσμα να επηρεάσει την ακρίβεια της ανάλυσης. Η ασυνέπεια στα δεδομένα μπορεί να παρεμποδίσει διεργασίες διαχείρισης αυτών. Ειδικά οι ροές δεδομένων μπορούν να είναι υψηλά ασυνεπείς με περιοδικές κορυφές οι οποίες προκύπτουν χρονικά ή από γεγονότα πολλές φορές και είναι δύσκολο να διαχειριστούν. Όπως για παράδειγμα οι σχολιασμοί και τα διαθέσιμα άρθρα πάνω σε πολιτικά ζητήματα σε προεκλογική περίοδο.



Εικόνα 1. Τα 4 Vs των Big Data

Τα big data είναι πολύ σημαντικά καθώς μπορούν να δώσουν απαντήσεις σε ζητήματα σημαντικά για τις εταιρείες. Οι απαντήσεις αυτές έχουν ως αποτέλεσμα οι εταιρείες να είναι σε θέση να πάρουν σημαντικές αποφάσεις. Μερικά χαρακτηριστικά παραδείγματα είναι η μείωση του κόστους και απαιτούμενου χρόνου παραγωγής, η παραγωγή νέων προϊόντων και προσφορών που καλύπτουν ανάγκες και απαιτήσεις των περισσότερων χρηστών καθώς και επίλυση και πρόληψη ανεπιθύμητων καταστάσεων. Μερικοί από τους τομείς που εκμεταλλεύονται την αξία των big data είναι οι τράπεζες, οι κυβερνήσεις, συστήματα υγείας και εκπαίδευσης και τομείς παραγωγής και εμπορίου προϊόντων. Μερικά ενδιαφέροντα γεγονότα για τα big data που αξίζουν να σημειωθούν είναι:

- Το μέγεθος των δεδομένων που έχει παραχθεί τα τελευταία 2 χρόνια ξεπερνάει το συνολικό μέγεθος των παραχθέντων δεδομένων από την αρχή της ανθρωπότητας.

- Υπολογίζεται πως μέχρι το 2020 για κάθε άνθρωπο η παραγωγή δεδομένων θα ισούται με 1,7 MB το δευτερόλεπτο.
- Και μέχρι τότε θα έχουμε συνολική αύξηση των 4.4 ZB με συνολικό μέγεθος περίπου 44 ZB.
- Κάθε δευτερόλεπτο γίνονται περίπου 40 χιλιάδες αναζητήσεις στο Google κάτι που ισούται με 1.2 τρισεκατομμύρια τον χρόνο.
- Το Facebook κατέγραψε ως μέγιστο αριθμό συνδεδεμένων χρηστών σε μια μέρα αριθμό που ξεπερνάει το 1 δισεκατομμύριο.
- Μέχρι το 2020 οι χρήστες έξυπνων τηλεφώνων θα ξεπεράσουν τον αριθμό των 6.1 δισεκατομμυρίων.
- 73% των εταιρειών έχει ή σκοπεύει να επενδύσει στο κοντινό μέλλον στα big data.
- Και το πιο συνταρακτικό από όλα είναι πως μόλις ένα 0.5% των συνολικών δεδομένων χρησιμοποιείται και έχει αναλυθεί αυτή τι στιγμή.

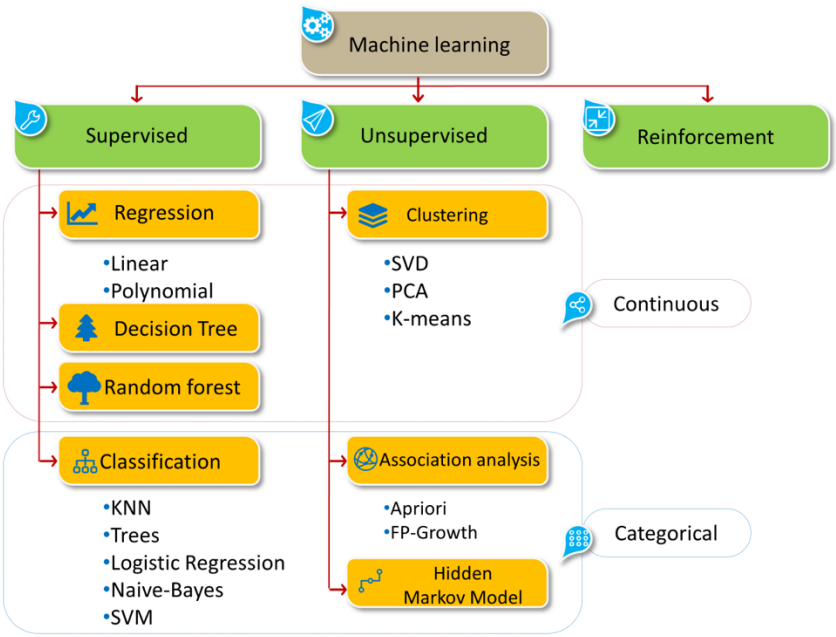
2.3 Τάσεις και νέες τεχνολογίες

Οι τάσεις στον τομέα των big data είναι κάτι που μπορεί να αλλάξει από την μια στιγμή στην άλλη. Οι αλλαγές στις επίκαιρες τεχνολογίες και εργαλεία γίνονται απροειδοποίητα και στιγμιαία. Η ζήτηση καθορίζεται από την ποιότητα των υπηρεσιών και τις δυνατότητες. Επομένως είναι πιθανό μια νέα τεχνολογία να κερδίσει γρήγορα ενδιαφέρον αν μπορεί να ανταπεξέλθει αποδοτικά στις τρέχουσες απαιτήσεις της αγοράς. Όπως προαναφέρθηκε τα ποσοστά των επιχειρήσεων που κάνουν χρήση των big data ή σκοπεύουν στο κοντινό μέλλον να επενδύσουν αυξάνονται συνεχώς. Κάτι που υποδηλώνει πως όλοι θέλουν την τελευταία τεχνολογία. Ενώ ο τομέας αυτός αναπτύσσεται καθημερινά και επιδέχεται αλλαγές υπάρχουν τεχνολογίες οι οποίες υπολογίζεται να παίξουν σημαντικό ρόλο σε αυτή την ανάπτυξη και να την επηρεάσουν σε μεγάλο βαθμό. Τέτοιες τεχνολογίες είναι η μηχανική μάθηση, προγνωστική ανάλυση, IoT και άλλες τις οποίες θα δούμε περιληπτικά παρακάτω.

Πρώτο και από τα πιο σημαντικά χαρακτηριστικά είναι ότι στην αγορά υπάρχουν πάρα πολλά εργαλεία ανοιχτού λογισμικού τα οποία προσφέρουν πολλές δυνατότητες και έχουν κερδίσει την προσοχή και το ενδιαφέρον σε αυτόν τον τομέα. Από τα πιο γνωστά και πολυχρησιμοποιημένα είναι το Hadoop και το Spark και τα δυο προϊόντα της Apache. Υπολογίζεται πως το ποσοστό των επιχειρήσεων που κάνουν χρήση του Hadoop θα είναι περίπου 60% και οι χρήστες του γενικά αυξάνονται κατά 33% κάθε χρόνο. Αναμένεται να επεκταθεί η χρήση τόσο του Hadoop όσο και τεχνολογιών NoSQL. Καθώς και αρχιτεκτονικές που θα επιταχύνουν την επεξεργασία των big data και θα παρέχουν την δυνατότητα αλληλεπίδρασης με αυτά σε πραγματικό χρόνο. Πολλές επιχειρήσεις επενδύουν σε μια τεχνολογία που ονομάζεται in-memory με σκοπό την επιτάχυνση της επεξεργασίας των δεδομένων. Συνήθως οι βάσεις δεδομένων αποθηκεύουν τα δεδομένα σε σκληρούς ή SSD δίσκους. Με την τεχνολογία in-memory η αποθήκευση γίνεται στην RAM κάτι που είναι αρκετά ταχύτερο να γίνει τόσο για εγγραφή όσο και για ανάκτηση.

Ένα άλλο μεγάλο πεδίο στο οποίο επενδύουν πολλές εταιρείες είναι η μηχανική μάθηση. Είναι μια υποκατηγορία της τεχνητής νοημοσύνης που ως στόχο έχει να δώσει την δυνατότητα στους υπολογιστές να αποκτήσουν γνώση χωρίς να είναι προγραμματισμένοι αποκλειστικά για μια λειτουργία. Η επιστήμη αυτή έχει ως στόχο την κατασκευή αλγορίθμων οι οποίοι είναι σε θέση να καταλήξουν σε μια τελική κατάσταση με βάση τα δεδομένα που τους παρουσιάζονται (data-driven). Όπως για παράδειγμα να προβλέψουν μια κατάσταση ή να πάρουν μια απόφαση. Οι αλγόριθμοι αυτοί για να μπορούν να λειτουργήσουν προαπαιτούν εκπαίδευση. Η εκπαίδευση μπορεί να χωριστεί σε 2 μεγάλες κατηγορίες επιβλεπόμενη και μη επιβλεπόμενη. Η διαφορά των 2 μεθοδολογιών είναι ότι στην $1^{η}$ περίπτωση για κάθε δεδομένο

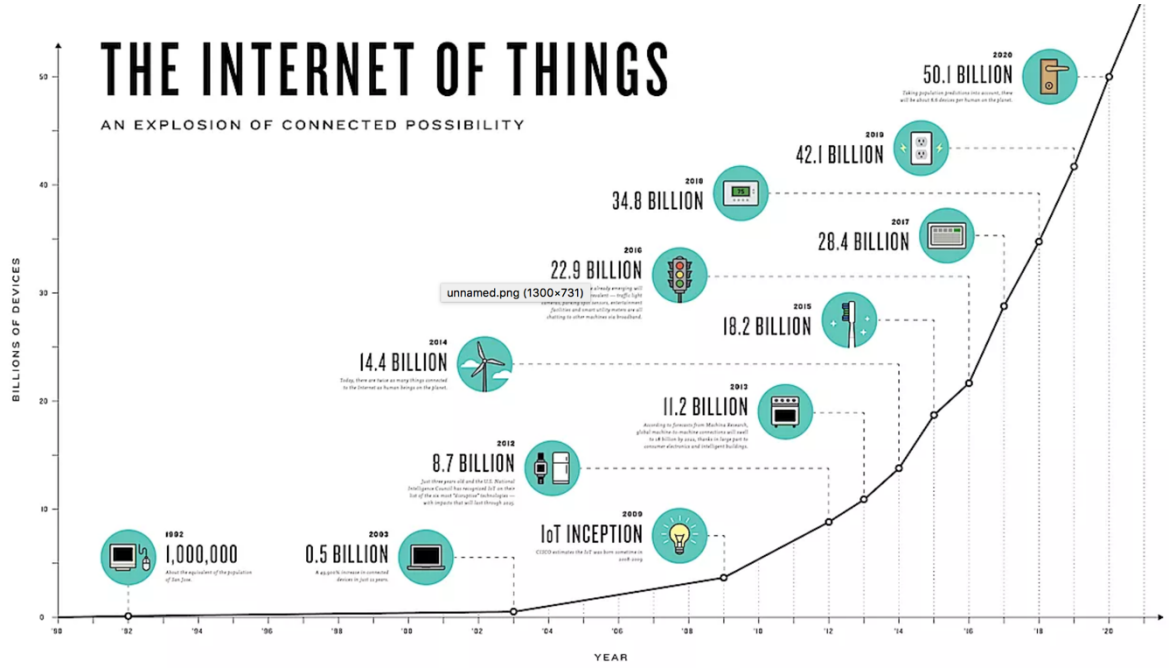
εισόδου εκπαίδευσης δίνουμε στον αλγόριθμο και το επιθυμητό αποτέλεσμα που πρέπει να δώσει ενώ στην $2^{η}$ περίπτωση ο αλγόριθμος δεν γνωρίζει αυτή την πληροφορία.



Εικόνα 2. Κατηγορίες της μηχανική μάθηση

Άλλη μια τάση η οποία είναι στενά συνδεδεμένη με την μηχανική μάθηση είναι η προγνωστική ανάλυση (predictive analytics). Πολλά συστήματα μηχανικής μάθησης παρέχουν τα κατάλληλα χαρακτηριστικά για να μπορεί να υποστηριχθεί λογισμικό προγνωστικής ανάλυσης. Ο στόχος αυτού του τομέα πάει ένα βήμα παραπέρα από την ισχύουσα κατάσταση. Όπου με τα παραγόμενα δεδομένα και την χρήση εργαλείων ανάλυσης γίνεται η προσπάθεια να κατανοήσουν οι εταιρείες γιατί προκύπτουν τα αποτελέσματα που προκύπτουν. Επομένως με τον συνδυασμό και την εφαρμογή αυτών των μεθοδολογιών θα γίνεται ανάλυση των big data για να προβλεφθεί μια μελλοντική κατάσταση. Υπολογίζεται πως μόλις ένα 29% των επιχειρήσεων κάνει χρήση τέτοιων εργαλείων παρ' όλα αυτά αναμένεται μεγάλη αύξηση τα επόμενα χρόνια. Ωστόσο υπάρχουν και άλλοι τομείς στους οποίους οι εταιρείες προσπαθούν να ενσωματώσουν κάποια λειτουργία η οποία βασίζεται σε τεχνητή νοημοσύνη. Όπως έξυπνες εφαρμογές οι οποίες για παράδειγμα διαχειρίζονται τα αποτελέσματα από την ανάλυση των big data και προηγούμενες συμπεριφορές των χρηστών και παράγουν εξατομικευμένα αποτελέσματα για κάθε χρήστη. Ένα άλλο παράδειγμα είναι η έξυπνη ασφάλεια. Οι εταιρείες αναλύουν τα αρχεία καταγραφής ασφαλείας τους με σκοπό την πρόληψη ηλεκτρονικών επιθέσεων.

Ο τομέας των IoT αναπτύσσεται ραγδαία στην εποχή μας παράλληλα σε αυτόν των big data. Όπως είναι αναμενόμενο πολλές φορές αυτές οι δυο τεχνολογίες συσχετίζονται μεταξύ τους καθώς μεγάλο τμήμα των big data προέρχεται από αυτά. Με όλες τις καινούργιες συσκευές και εφαρμογές των IoT οι επιχειρήσεις θα έρθουν αντιμέτωπες με ακόμα ταχύτερη αύξηση των δεδομένων τους. Με αποτέλεσμα πολλές να χρειαστούν νέες τεχνολογίες και συστήματα για να είναι σε θέση να αναλύσουν αυτό το μέγεθος δεδομένων και να ανακτήσουν πληροφορίες που θα προσφέρουν κέρδος σε αυτές.



Εικόνα 3. Η αύξηση των IoT

Αυτό έχει ως αποτέλεσμα οι εταιρείες να στρέφονται σε μια τεχνολογία η οποία ονομάζεται edge computing. Το προτέρημα αυτής της τεχνολογίας είναι πως παρέχει την επεξεργασία των δεδομένων «κοντά» στην συσκευή παραγωγής τους. Με αποτέλεσμα οι επιχειρήσεις να έχουν λιγότερες μεταφορές στο δίκτυο τους κάτι που αυξάνει την επίδοση του και την εξοικονόμηση. Αυτό επιτυγχάνετε επειδή οι διάφορες επεξεργασίες δεν γίνονται σε κάποια υπηρεσία νεφους και το μέγεθος των δεδομένων μειώνεται σημαντικά πριν αυτά αποθηκευτούν. Μια ακόμα δυνατότητα που παρέχει αυτή η τεχνολογία είναι η επιτάχυνση της επεξεργασίας των δεδομένων με αποτέλεσμα οι υπεύθυνοι για την λήψη αποφάσεων να είναι σε θέση να προβούν σε ενέργειες ταχύτερα με βάση τα παραγόμενα αποτελέσματα.

2.4Μέθοδοι ανάλυσης και επεξεργασίας

Η διαδικασία της κατηγοριοποίησης κειμένου σε θετικό/αρνητικό μπορεί να επιτευχθεί με δυο τεχνικές, αυτή της μηχανικής μάθησης και βασισμένη σε λεξικό. Υπάρχει και τρίτη επιλογή η οποία ονομάζεται υβριδική και είναι ο συνδυασμός των δυο πρώτων. Οι δυο αυτές κύριες κατηγορίες διαχωρίζονται σε 2 υποκατηγορίες η κάθε μια. Μια από τις υποκατηγορίες της μηχανικής μάθησης διαχωρίζεται σε ακόμα τέσσερεις. Παρακάτω θα αναφερθώ περιληπτικά στην κάθε ομάδα για να δούμε τα κύρια χαρακτηριστικά τους και στην εικόνα 4 συνοψίζεται αυτός ο διαχωρισμός.

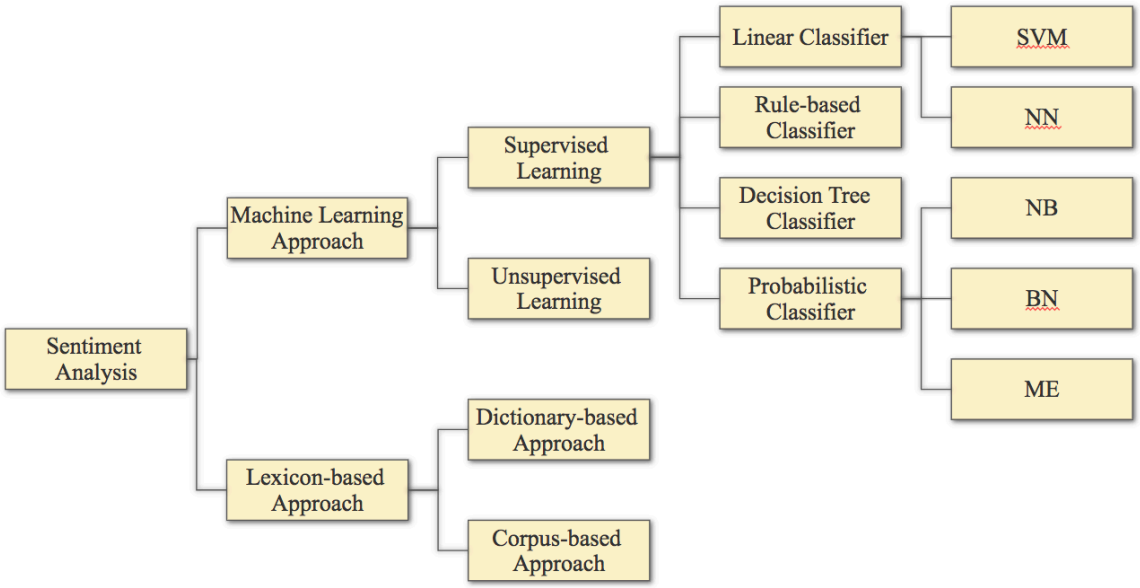
Μηχανική μάθηση: αυτή η τεχνική κάνει χρήση αλγορίθμων της μηχανική μάθηση για την επίτευξη του στόχου μας. Αντιμετωπίζει το πρόβλημα της συναισθηματικής ανάλυσης ως ένα κανονικό πρόβλημα κατηγοριοποίησης κειμένου με την βοήθεια συντακτικών και γλωσσικών χαρακτηριστικών. Το μοντέλο που θα προκύψει από αυτήν τη μέθοδο είναι σε θέση όταν του παρουσιάζονται τα χαρακτηριστικά από μια άγνωστη «κατάσταση» να μπορεί να προβλέψει την ομάδα στην οποία αυτή ανήκει. Αυτή η μέθοδος χωρίζεται σε δυο υποκατηγορίες ανάλογα με τον τρόπο εκπαίδευσης του μοντέλου.

Επιβλεπομένη: με τον όρο αυτόν εννοούμε πως τα δεδομένα που παρουσιάζουμε στο μοντέλο μας κατά την εκπαίδευση είναι γνωστή η κατηγορία στην οποία ανήκει η κάθε εισαγόμενη περίπτωση. Οι αλγόριθμοι που ανήκουν σε αυτή την κατηγορία χωρίζονται σε τέσσερις κατηγορίες ανάλογα με τις τεχνικές που εφαρμόζουν.

Πιθανοτική: Η μεθοδολογία αυτής της κατηγορίας βασίζεται σε ένα πιθανοτικό μοντέλο το οποίο αποτελείται από όλες τις πιθανές καταστάσεις ή αλλιώς μια πιθανοτική κατανομή. Ως αποτέλεσμα έχει, αφού αναλυθεί η περίπτωση μας, τις πιθανότητες να ανήκει για κάθε ομάδα και όχι μόνο μια την πιο πιθανή.
Γραμμική: Οι αλγόριθμοι αυτής της κατηγορίας λειτουργούν με διανύσματα. Ένα διάνυσμα Α αποτελούν οι κανονικοποιημένες τιμές συχνοτήτων για τους όρους ενός κειμένου, ένα δεύτερο διάνυσμα Β περιέχει τους γραμμικούς συντελεστές με διαστάσεις ίδιες με το πρώτο και μια τιμή χ που είναι ένας βαθμωτός συντελεστής. Το αποτέλεσμα του γραμμικού ταξινομητή προκύπτει από τον τύπο AB+χ και αποτελεί ένα υπερεπίπεδο (επίπεδο πολλών διαστάσεων) το οποίο διαχωρίζει τον χώρο στις δυνατές κλάσεις.
Δέντρο αποφάσεων: η τεχνική της ταξινόμησης με δέντρα αποφάσεων παρέχει έναν ιεραρχικό διαμοιρασμό των δεδομένων εκπαίδευσης με βάση μια συνθήκη για τις τιμές των χαρακτηριστικών. Η συνθήκη αυτή είναι η ύπαρξη ή απουσία ενός ή περισσοτέρων χαρακτηριστικών στο σύνολο μας. Ο διαχωρισμός αυτός επαναλαμβάνεται αναδρομικά μέχρις ότου τα φύλλα του δέντρου περιέχουν έναν ελάχιστο αριθμό χαρακτηριστικών που θα συμβάλουν στην κατηγοριοποίηση.
Κατά κανόνα: Σε αυτή την τεχνική ο χώρος των δεδομένων διαχωρίζεται με βάση κάποιο σετ κανόνων. Το αριστερό μέρος του κανόνα αποτελεί τις συνθήκες ή προϋποθέσεις προκειμένου να καταλήξουμε στο δεξί μέρος του κανόνα που είναι το αποτέλεσμα, στην περίπτωση μας η κατηγορία στην οποία ανήκει η περίπτωση που αναλύεται. Συνήθως οι κανόνες βασίζονται στην παρουσία κάποιου χαρακτηριστικού και όχι απουσίας. Η δεύτερη περίπτωση αποφεύγεται γιατί δεν αποτελεί πηγή πληροφόρησης ειδικά σε δεδομένα μικρού μεγέθους.
Μη επιβλεπόμενη: αυτή είναι η ακριβώς αντίθετη κατάσταση από την επιβλεπόμενη εκπαίδευση στην οποία η εκπαίδευση του μοντέλου μας γίνεται χωρίς να γνωρίζει τα επιθυμητά αποτελέσματα κατά την διαδικασία αυτή. Συνήθως η επιλογή του μη επιβλεπόμενη εκπαίδευση γίνεται γιατί η απόκτηση μεγάλων δεδομένων είναι εύκολη και γρήγορη αλλά η διαδικασία κατηγοριοποίησης αυτών των δεδομένων αρχικά πρέπει να γίνει από κάποιον άνθρωπο.
Βασισμένη σε λεξικό (Lexicon-based): αυτή η μεθοδολογία έχει ως στόχο να κατηγοριοποίηση το κείμενο με βάση λέξεις, εκφράσεις και ιδιωματισμούς, που εκφράζουν γνώμη/άποψη, που αυτό περιέχει. Η διαδικασία εύρεσης τέτοιων χαρακτηριστικών όμως μπορεί να γίνει με τρεις δυνατούς τρόπους. Ο πρώτος τρόπος είναι χειρωνακτικός και απαιτεί πολύ χρόνο η μοναδική του χρήση είναι να βοηθήσει στην επαλήθευση των αποτελεσμάτων. Οι άλλοι δυο είναι αυτοματοποιημένοι.
Βασισμένη σε λεξικό (Dictionary-based): με αυτή την μεθοδολογία επιλέγονται μερικοί όροι των οποίων είναι γνωστή η διάθεση από τον άνθρωπο και στην συνέχεια αρχίζει μια αυτοματοποιημένη αναζήτηση συνώνυμων και αντώνυμων ορών από μεγάλες πηγές. Οι νέοι όροι που ανακαλύφθηκαν προσθέτονται σε μια λίστα και ξεκινάει η ίδια διαδικασία για αυτούς τους όρους. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να μην εισαχθούν νέοι όροι μετά από κάποια επανάληψη. Στο τέλος μπορούν να γίνουν βελτιώσεις με ανθρώπινη παρέμβαση για την αφαίρεση εσφαλμένων ορών. Ένα μεγάλο μειονέκτημα αυτής της μεθόδου είναι ότι δεν μπορεί να καταλάβει την διάθεση σε καταστάσεις όπου το περιεχόμενο και ο τομέας του θέματος παίζουν ρόλο.
Βασισμένη σε σώμα(Corpus-based): η μέθοδος αποτελεί επέκταση της προηγούμενης καθώς έχει ως στόχο την επίλυση του προβλήματος που προαναφέρθηκε. Η μέθοδος αυτή ακολουθεί το ίδιο σκεπτικό με την προηγουμένη για την δημιουργία μιας λίστας με χαρακτηριστικά με την διαφορά πως για την εύρεση των ορών η μέθοδος αυτή λαμβάνει υπόψιν συντακτικά μοτίβα και άλλα μοτίβα που προκύπτουν κατά την εύρεση των ορών.

Ωστόσο πέρα από τις μεθόδους εκπαίδευσης πολύ σημαντικό ρόλο παίζουν και τα στάδια προ επεξεργασίας του κειμένου (Natural Language Processing) και επιλογής των

χαρακτηριστικών (feature extraction) από αυτό. Στο στάδιο προ επεξεργασίας μπορούμε να ακολουθήσουμε διάφορες τεχνικές οι οποίες θα βελτιώσουν τα αποτελέσματα. Για παράδειγμα ένα από τα πρώτα βήματα είναι ο διαχωρισμός του κειμένου σε προτάσεις ή λέξεις. Στην συνέχεια στα σύνολα (προτάσεων ή λέξεων) που προκύπτουν μπορούμε να εφαρμόσουμε περαιτέρω επεξεργασία όπως αφαίρεση stop words δηλαδή λέξεων που δεν προσδίδουν αξία/πληροφορία όπως άρθρα, συνδετικές λέξεις κα. παρόμοια. Χρήσιμες μέθοδοι είναι αυτές του stemming και lemmatisation οι οποίες συμβάλουν κατά κάποιο τρόπο στην μείωση του μεγέθους της πληροφορίας χωρίς να χάσουμε σημαντικά δεδομένα. Το stemming είναι η διαδικασία αφαίρεσης των καταλήξεων από λέξεις για την ανακάλυψη της ρίζας. Το lemmatization είναι η διαδικασία που ανάγει παράγωγες λέξεις στα ριζικά τους. Τα αποτελέσματα αυτής της λειτουργίας επηρεάζονται ανάλογα με το τι μέρος του λόγου είναι ο όρος που επεξεργαζόμαστε. Κάτι που μας οδηγεί στην επόμενη δυνατή επεξεργασία που είναι η σημείωση των μερών του λόγου δηλαδή η αναγνώριση σε ποιο μέρος του λόγου ανήκουν οι λέξεις του κειμένου μας. Με βάση τα όσα προαναφέρθηκαν μπορούμε να καταλήξουμε σε πιο περίπλοκες επεξεργασίες και να ανακαλύψουμε περισσότερες πληροφορίες με την αναζήτηση μοτίβων με κανονικές εκφράσεις όπως και αν λαμβάνουμε υπόψιν οντότητες όπως ονόματα ανθρώπων, πόλεων/χωρών, γνωστών κτιρίων κλπ. Το επόμενο στάδιο είναι ο τρόπος με τον οποίο θα διαλέξουμε τα χαρακτηριστικά που θα λάβουν μέρος στην εκπαίδευση. Και σε αυτό το στάδιο έχουμε 2 κατηγορίες μεθόδων εκ των οποίων η μια είναι μη-αυτοματοποιημένη (Lexicon based) και οι πρώτοι όροι επιλέγονται από κάποιον και στην συνέχεια με συνώνυμες λέξεις και άλλα μεγάλα σετ παράγεται το τελικό σετ μας. Οι άλλες μέθοδοι είναι στατιστικές και πλήρως αυτοματοποιημένες. Αν και οι διαφορές τους είναι στον τρόπο που διαχειρίζονται τις πληροφορίες και οι τύποι που τις ορίζουν, η γενική προσέγγιση είναι ίδια. Τα αρχεία μπορούν να διαχειριστούν ως ένα σύνολο λέξεων (bag of words) ή σαν ένα αλφαριθμητικό που διατηρεί την ακολουθία των λέξεων ως έχουν στο κείμενο. Η πρώτη προσέγγιση προτιμάται τις περισσότερες φορές λόγο της απλότητας της. Μερικές από τις πιο συχνά χρησιμοποιημένες στατιστικές μεθόδους είναι οι PMI (Point-wise Mutual Information), χ-τετράγωνο (χ^2) και LSI (Latent Semantic Indexing).



Εικόνα 4. Κατηγορίες τεχνικών για συναισθηματική ανάλυση και οι υποκατηγορίες τους.

3.Στοίβα Elastic

3.1Εισαγωγη

Η στοίβα elastic ή αλλιώς ELK (από τα πρώτα γράμματα των 3 κύριων προϊόντων Elasticsearch, Logstash και Kibana) είναι ένα σύνολο εργαλείων ανοιχτού λογισμικού που ως στόχο έχουν την εύκολη συλλογή, αποθήκευση, ανάκτηση και οπτικοποίηση δεδομένων. Καθώς και την ανακάλυψη πληροφοριών από αυτά. Πλέον στην συλλογή τους υπάρχουν και άλλα εργαλεία όπως το Beats ,X-Pack και Cloud τα οποία προσδίδουν ακόμα περισσότερες δυνατότητες στα υπάρχοντα και τα επεκτείνουν για περισσότερες και πιο εξειδικευμένες περιπτώσεις χρήσης. Τα εργαλεία αναπτύχθηκαν με στόχο τη δημιουργία επεκτάσιμων λύσεων για να μπορούν να είναι κατανεμημένα χωρίς να χρειάζονται πολύ μεγάλες αλλαγές και το υλοποίησαν κάνοντας χρήση μιας κοινά χρησιμοποιούμενης διεπαφής JSON μέσο HTTP.

Τα εργαλεία αυτά είναι πολύ ευέλικτα όσων αφορά την συμβατότητα τους με άλλες τεχνολογίες και γλώσσες προγραμματισμού καθώς τα ίδια είναι γραμμένα σε διάφορες γλώσσες για να εξυπηρετήσουν ανάγκες όπως φορητότητα και αποδοτική διανομή των μεταγλωττισμένων δυαδίκων αρχείων ή ανάπτυξη ενιαίων τμημάτων frontend και backend. Επίσης σχεδιάζονται και επεκτείνονται πολλά πρόσθετα τακτικά για διάφορες άλλες τεχνολογίες και γλώσσες και ο κάθε χρήστης επίσης μπορεί να συμβάλλει στην δημιουργία και βελτίωση των προσθέτων με βάση τις ανάγκες του. Τα παραπάνω χαρακτηριστικά έχουν καταστήσει τις παραπάνω τεχνολογίες ελκυστικές για πολλές μεγάλες εταιρίες και επιχειρήσεις όπως Adobe, Facebook, CERN, GitHub, Mozilla, Netflix, Quora και άλλες πολλές.

3.2Logstash

3.2.1Εισαγωγή

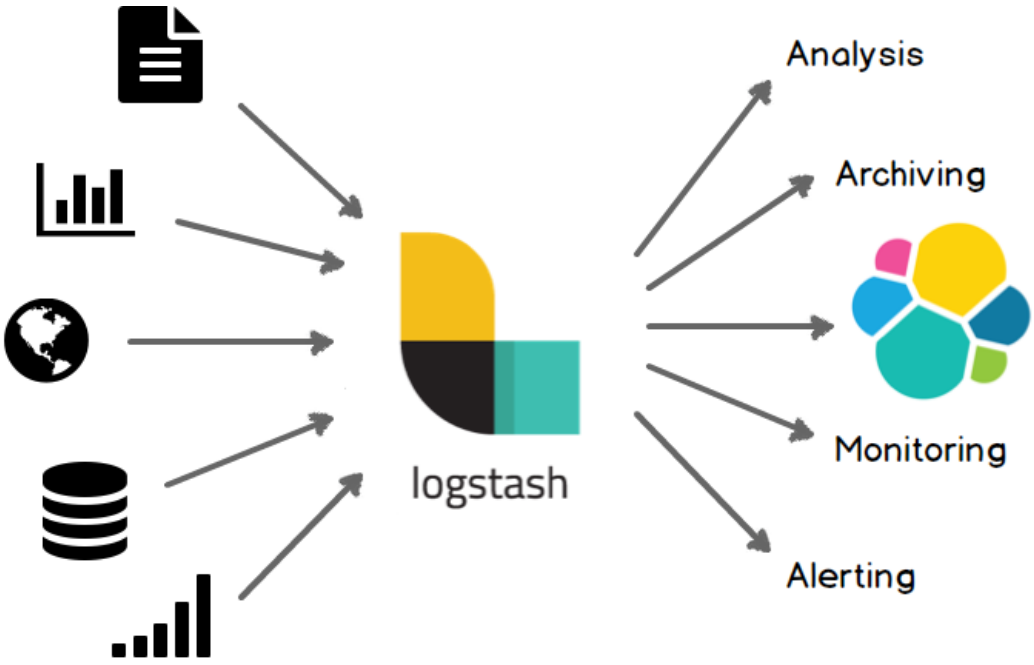
Το logstash είναι μια μηχανή συλλογής δεδομένων ανοιχτού κώδικα με δυνατότητες σωλήνωσης σε πραγματικό χρόνο. Μπορεί να ενοποιήσει δυναμικά τα δεδομένα από διαφορετικές πηγές και να τα κανονικοποιήσει σε προορισμούς τις επιλογής μας. Παρέχει τη δυνατότητα «καθαρισμού» και οργάνωσης των δεδομένων πριν προχωρήσουμε σε εξειδικευμένες αναλύσεις και την οπτικοποίηση τους. Ενώ η αρχική ιδέα ήταν καθαρά η συλλογή αρχείων καταγραφής στην συνέχεια επεκτάθηκε παραπάνω από αυτό. Οποιοσδήποτε τύπος γεγονότος μπορεί να εμπλουτισθεί και να μετατραπεί από μια πληθώρα προσθέτων εισόδου, εξόδου και φιλτραρίσματος όπως επίσης και πολλούς κωδικοποιητές οι οποίοι συμβάλουν στην περαιτέρω απλοποιημένη διαδικασία «αποδοχής» του αρχείου καταγραφής. Τα κύρια χαρακτηριστικά του logstash τα οποία θα δούμε και παρακάτω είναι η οριζόντια επεκτάσιμη σωλήνωση επεξεργασίας δεδομένων με μεγάλη στήριξη από τα άλλα 2 εργαλεία elasticsearch και kibana και η δυνατότητα συνδυασμού, ταιριάσματος και οργάνωσης των διάφορων προσθέτων τα οποία θα χρησιμοποιηθούν κατά την διαδικασία. τα διαθέσιμα πρόσθετα είναι πάνω από 200 τα οποία συντηρούνται από την ίδια την εταιρεία όπως και την κοινότητα προγραμματιστών. Επίσης ο κάθε χρήστης έχει την δυνατότητα σχεδιασμού και επεκτάσεις αυτών ανάλογα με τις ανάγκες και την περίπτωση χρήσης.

Ας δούμε τώρα λίγο πιο αναλυτικά τις δυνατότητες και τρόπους χρήσης του logstash. Όπως προαναφέρθηκε η αρχική ιδέα ήταν η συλλογή αρχείων καταγραφής κάτι το οποίο καταφέρνει με επιτυχία καθώς μπορεί να διαχειριστεί οποιοδήποτε τύπο αρχείου καταγραφής με ευκολία και σε μεγάλους αριθμούς όπως για παράδειγμα αρχείων καταγραφής ιστού από έναν Apache διακομιστή, αρχεία καταγραφής από εφαρμογές καταγραφής όπως το log4j και άλλες πολλές μορφές όπως καταγραφές συστημάτων, δικτύων και τοίχων προστασίας. Επίσης είναι δυνατή η καταγραφή μετρικών από διάφορες υποδομές μέσω TCP ή UDP και όλες αυτές

οι καταγραφές πραγματοποιούνται με πλήρη ασφαλή προώθηση αυτών. Άλλη σημαντική δυνατότητα είναι η μετατροπή HTTP αιτήσεων σε «γεγονότα» όπως και η δημιουργία γεγονότων από δικιά μας αίτηση κατά παραγγελία σε HTTP endpoint για παράδειγμα η κατανάλωση twitter posts από υπηρεσία ιστού τροφοδοσίας ή η καταγραφή κατάστασης, απόδοσης και μετρικών από διεπαφές εφαρμογών ιστού ιδανικό για σενάρια οπού η απαίτηση της κατάστασης κατά παραγγελία προτιμάται από την αποδοχή των δεδομένων ανά τακτά χρονικά διάστημα από την εφαρμογή. Με αυτόν τον τρόπο καταγραφής των συμβάντων και γεγονότων γίνεται ευκολότερη η παρατήρηση και εξόρυξη πληροφορίας από τα δεδομένα που έχουμε στην διάθεση μας. Μια ακόμα αποδοτική και με μεγάλες δυνατότητες είναι η καταγραφή πληροφοριών από αισθητήρες και IoT συσκευές και συστήματα. Είναι δυνατή η αποστολή πληροφοριών από κινητές εφαρμογές που επικοινωνούν με έξυπνα σπίτια, οχήματα, μηχανήματα καταγράφεις υγείας και πολλές άλλες εξειδικευμένες εφαρμογές βιομηχανίας.

Για καλυτέρα αποτελέσματα ανάλυσης και πληροφορίας χρειαζόμαστε τα «ιδανικά» δεδομένα. Οπότε ο καθαρισμός και η μετατροπή των δεδομένων κατά την εισαγωγή μπορεί να οδηγήσει σε εμφανείς πληροφορίες κατά την ευρετηριοποίηση ή έξοδο των δεδομένων. Το logstash έρχεται έτοιμο με πολλές δυνατότητες συσσωμάτωσης, μετατροπών, αναγνώρισης προτύπων και γεωγραφικής τοποθεσίας καθώς και δυνατότητες δυναμικής αναζήτησης.

Όσα προαναφέρθηκαν για τις δυνατές πηγές από τις οποίες μπορούμε να αντλήσουμε αρχεία καταγραφής και γεγονότα για αποθήκευση και επεξεργασία καθώς και τη δυνατότητα προωθήσεις των δεδομένων σε διάφορους προορισμούς της επιλογής μας ανάλογα με την περαιτέρω επεξεργασία που θέλουμε να κάνουμε ή τις δυνατότητες ευρετηριοποίησης και ανάλυσης δεδομένων που χρειαζόμαστε. Μπορούν να συνοψισθούν στην παρακάτω εικόνα.



Εικόνα 5. Πηγές εισόδου και προορισμοί εξόδου των δεδομένων

3.2.2Τρόπος λειτουργίας

Η σωλήνωση επεξεργασίας των γεγονότων έχει 3 στάδια : είσοδος, φιλτράρισμα και έξοδος. Κατά το στάδιο εισόδου γίνεται η δημιουργία των γεγονότων από τις διάφορες πηγές, τα φίλτρα κάνουν την τροποποίηση των γεγονότων και στην έξοδο γίνεται ο διαμοιρασμός τους. Οι είσοδοι και έξοδοι υποστηρίζουν κωδικοποιητές που δίνουν την δυνατότητα

κωδικοποίησης και αποκωδικοποίησης των δεδομένων που εισάγονται ή εξάγονται από την σωλήνωση χωρίς να χρειάζεται η εφαρμογή έξτρα φίλτρου υπεύθυνου καθαρά για αυτή την λειτουργία.

Οι είσοδοι χρησιμοποιούνται για να εισάγουμε γεγονότα και δεδομένα στο logstash. στην ιστοσελίδα με την τεκμηρίωση (document) υπάρχει μια πλήρης λίστα με όλα τα πρόσθετα εισόδου που είναι διαθέσιμα απευθείας για χρήση μετά την εγκατάσταση τους. Τα πιο συχνά χρησιμοποιημένα είναι

- file
- syslog
- redis
- beats

Τα φίλτρα είναι ενδιάμεση επεξεργασία των δεδομένων στην όλη διαδικασία του logstash. Είναι δυνατός ο συνδυασμός των φίλτρων με λογικές εκφράσεις για την εκτέλεση μια ενέργειας πάνω στα δεδομένα μόνο αν πληρούν συγκεκριμένες προϋποθέσεις. Τα πιο συχνά χρησιμοποιημένα είναι

- grok
- mutate
- drop
- geoip

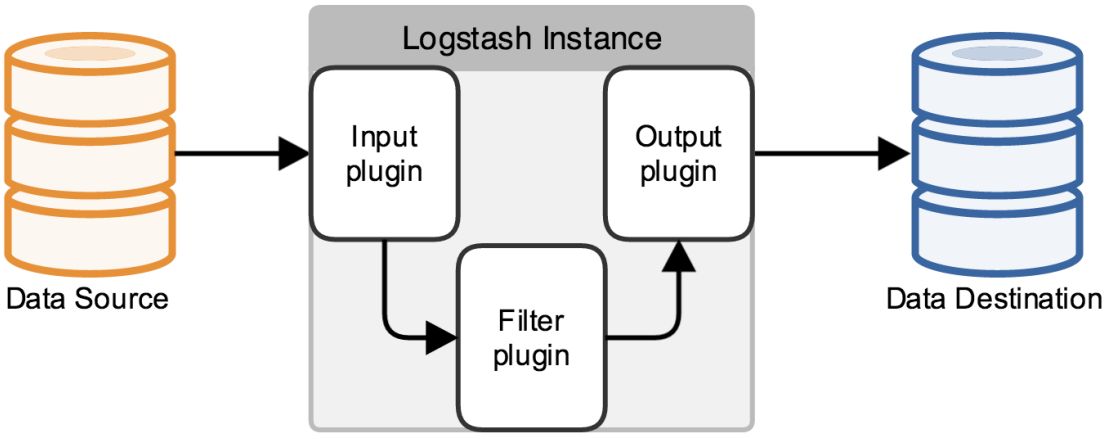
Οι έξοδοι είναι το τελευταίο στάδιο της σωλήνωσης. Τα δεδομένα μπορούν να περάσουν σε περισσότερες από μια εξόδους. Όμως όταν ολοκληρωθούν όλες οι έξοδοι ενός γεγονότος αυτό τελειώνει και η επεξεργασία περνάει στο επόμενο γεγονός. Οι πιο συχνά χρησιμοποιημένες είναι

- elasticsearch
- file
- graphite
- hadoop

Οι κωδικοποιητές είναι φίλτρα ροών που μπορούν να εφαρμοστούν κατά τις διαδικασίες εισόδου ή εξόδου. Ουσιαστικά συμβάλουν στον διαχωρισμό της μεταφοράς και της στειροποίησης. Δυο δημοφιλείς και συχνά χρησιμοποιουμένοι κωδικοποιητές είναι

- json
- plain

Η παραπάνω αναλυτική περιγραφή του τρόπου λειτουργίας του logstash φαίνεται συνοπτικά στην παρακάτω εικόνα.



Εικόνα 6. Σωλήνωση του Logstash και τα 3 στάδια της διαδικασίας.

3.2.3Μοντέλο Εκτέλεσης

Η σωλήνωση επεξεργασίας του logstash συντονίζει την εκτέλεση των 3ων σταδίων. Κάθε στάδιο εισόδου στην σωλήνωση εκτελείται σε ξεχωριστώ νήμα. Τα δεδομένα εγγράφονται από την είσοδο σε μια κοινή συγχρονισμένη ουρά της Java (SynchronousQueue). Η λειτουργία της ουράς δεν είναι η αποθήκευση των δεδομένων αλλά η προώθηση αυτών σε ένα ελεύθερο νήμα. Σε περίπτωση που όλα τα νήματα είναι σε λειτουργία μπλοκάρει την είσοδο καινούργιων δεδομένων. Κάθε νήμα της σωλήνωσης παίρνει ένα τμήμα των εισαγόμενων δεδομένων από την ουρά, περνάει τα δεδομένα από κάθε ορισμένο φίλτρο και τέλος τα επεξεργασμένα πλέον δεδομένα τα περνάει στα σημεία εξόδου. Το πλήθος των νημάτων και το μέγεθος του τμήματος το οποίο αναλαμβάνει το κάθε ένα για επεξεργασία μπορούν να καθοριστούν στο αρχείο διαμόρφωσης. Το SynchronousQueue είναι η προεπιλεγμένη ουρά που χρησιμοποιείται. Στην περίπτωση αυτή ωστόσο αν το logstash τερματίσει απροσδόκητα τα δεδομένα ,που είχαν αποθηκευτεί στην προσωρινή μνήμη πριν επεξεργαστούν, θα χαθούν.

Για την αποφυγή αυτής της ανεπιθύμητης περίπτωσης μπορεί να γίνει χρήση μιας PersistentQueue η οποία γραφεί τα δεδομένα στον σκληρό δίσκο. Όταν τα δεδομένα γράφονται στην ουρά επιτυχώς το πρόσθετο εισόδου στέλνει μια επιβεβαίωση στην πηγή των δεδομένων και στην συνέχεια το logstash κρατάει ένα αρχείο με τα επιτυχώς επεξεργασμένα δεδομένα. Τα δεδομένα καταγράφονται ως επιτυχή όταν ολοκληρώσουν τα στάδια φιλτραρίσματος και εξόδου με επιτυχία. Μετά από περίπτωση βλάβης όταν γίνει επανεκκίνηση του συστήματος θα συνεχιστεί η επεξεργασία των δεδομένων από την ουρά καθώς και η λήψη νέων. Ένας ακόμα προνομιακός λόγος χρήσης αυτής της ουράς είναι η αποφυγή εγκατάστασης και διαχειρίσεις ενός message broker (π.χ. RabbitMQ, Apache Kafka) στην περίπτωση που αναμένουμε δεδομένα μεγάλου μεγέθους και η χρήση μεγάλου ρυθμιστή είναι απαραίτητη. Ακόμα και σε αυτήν την περίπτωση όμως δεν είναι εγγυημένη η 100% προστασία των δεδομένων από κάποιου είδους απώλειες. Για παράδειγμα πρόσθετα εισόδου τα οποία δεν υποστηρίζουν πρωτόκολλο δημοσίευσης - εγγραφής και σφάλματα ή βλάβες στον ίδιο τον σκληρό δεν καλύπτονται. Επίσης μπορεί να γίνει χρήση και του dead letter queue. Ουρά στην οποία γίνεται εγγραφή δεδομένων των οποίων η διαδικασία επεξεργασίας απέτυχε για λόγους όπως να μην ταιριάζουν τα δεδομένα ή να μην είναι γνωστός ο τρόπος επεξεργασίας.

3.2.4Σύντομη περιγραφή εκκίνησης

Πρώτη προϋπόθεση πριν την εγκατάσταση του logstash είναι να έχουμε εγκατεστημένη JAVA έκδοση 8 (9 δεν υποστηρίζεται) και να ορίσουμε την μεταβλητή περιβάλλοντος (environment variable) JAVA_HOME η οποία έχει το μονοπάτι στον φάκελο όπου έχουμε εγκατεστημένη την JAVA. Στην συνέχεια επιλέγουμε να κατεβάσουμε την διανομή (distribution) που μας ταιριάζει και ακολουθούμε τα αντίστοιχα βήματα για την εγκατάσταση. Στην συνέχεια πριν ξεκινήσουμε το logstash μπορούμε να διαμορφώσουμε τον τρόπο λειτουργίας του ορίζοντας ρυθμίσεις σε συγκεκριμένα αρχεία ανάλογα με τις ανάγκες μας σε περίπτωση που δεν θέλουμε να χρησιμοποιήσουμε τις προκαθορισμένες. Το logstash έχει δυο ειδών αρχεία διαμόρφωσης. Το αρχείο διαμόρφωσης σωλήνωσης, το οποίο είναι απαραίτητο να οριστεί πριν την εκκίνηση με τουλάχιστον μια είσοδο και έξοδο, και το αρχείο ρυθμίσεων στο οποίο μπορούμε να καθορίσουμε τον τρόπο λειτουργίας του εργαλείου. Για παράδειγμα τα αρχεία logstash.yml, jvm.options, startup.options υπάρχουν ήδη και είναι προκαθορισμένα αλλά είναι δυνατή η τροποποίηση τους. Μερικά παραδείγματα ρυθμίσεων που μπορούν να καθοριστούν είναι το πλήθος των πυρήνων που θα χρησιμοποιηθούν (pipeline.workers), το πλήθος των γεγονότων που θα δέχεται κάθε νήμα (pipeline.batch.size), ο τύπος ουράς

(queue.type) και αν αλλαγές στα αρχεία διαμόρφωσης σωλήνωσης να εφαρμόζονται επιτόπου χωρίς να χρειάζεται να κάνουμε επανεκκίνηση του logstash (config.reload.automatic) κα.

Μετά από αυτά τα βήματα μπορούμε τώρα να ξεκινήσουμε το logstash από το τερματικό. Η απλούστερη εντολή είναι bin/lostash –f pipeline.conf όπου η παράμετρος είναι το όνομα του αρχείου διαμόρφωσης σωλήνωσης που να χρησιμοποιήσει. Κατά την εκκίνηση από το τερματικό μπορούμε να ορίσουμε πάλι πλήθος παραμέτρων και συγκεκριμένα ρυθμίσεις λειτουργίας σε περίπτωση που θέλουμε να κάνουμε αντικατάσταση κάποιες από τις ρυθμίσεις στα αρχεία ρυθμίσεων. Το logstash κατά την λειτουργία του αποθηκεύει επίσης και εσωτερικά αρχεία καταγραφής συνήθως στον φάκελο logs κάτι το οποίο μπορεί να ρυθμιστεί όπως και το επίπεδο των αρχείων καταγραφής που αποθηκεύονται (debug, info, error κα.). Τώρα για τον τερματισμό του υπάρχουν διάφοροι τρόποι. Σε Mac (λογικά και άλλα Unix συστήματα) με control + C γίνεται τερματισμός. Δηλαδή σταματάει η είσοδος, τα γεγονότα που δεν έχουν ολοκληρωθεί ακόμα περνάνε τα στάδια της σωλήνωσης και τέλος γίνεται πλήρης ασφαλής τερματισμός της λειτουργίας. Άλλη επιλογή είναι πάλι από το τερματικό, αν κατά την εκκίνηση κρατήσουμε το PID της διεργασίας να εκτελέσουμε kill –TERM {PID}. Αυτή είναι μια συνοπτική και γρήγορη εκκίνηση του logstash.

3.3Elasticsearch

3.3.1Εισαγωγή

Το elasticsearch είναι μια μηχανή υψηλά κλιμακούμενη ανοιχτού κώδικα που παρέχει αναζήτηση σε πλήρες κείμενο και ανάλυση. Παρέχει την δυνατότητα αποθήκευσης, αναζήτησης και ανάλυσης δεδομένων μεγάλου όγκου πολύ γρήγορα και σχεδόν σε πραγματικό χρόνο. Και αυτό γιατί υπάρχει μια καθυστέρηση από την στιγμή που θα γίνει ευρετηριοποίηση των δεδομένων μέχρι να γίνουν διαθέσιμα προς αναζήτηση. Το elasticsearch κάνει χρήση του Lucene και επομένως αντεστραμμένων ευρετηρίων για τις αναζητήσεις. Μπορεί να κλιμακωθεί έως και petabytes δομημένων και μη δεδομένων. Στην γενική του χρήση χρησιμοποιείται ως τεχνολογία που ενισχύει εφαρμογές που παρέχουν αναζήτηση με πολύπλοκα χαρακτηριστικά και πολύπλοκες απαιτήσεις ή για εξόρυξη πληροφοριών από δεδομένα μεγάλου όγκου.

Οι παρακάτω έννοιες είναι βασικές για την όλη τεχνολογία και είναι στενά συνδεδεμένες μεταξύ τους για την λειτουργία του προγράμματος.

Συστοιχία: είναι η ομάδα ενός ή περισσότερων κομβψν που μαζί περιέχουν όλα τα δεδομένα μας και παρέχει ομαδοποιημένη ευρετηριοποίηση και δυνατότητες αναζήτησης σε όλους τους κόμβους. Κάθε συστοιχία χαρακτηρίζεται μοναδικά από μια ιδιότητα/όνομα, το προκαθορισμένο όνομα κατά την δημιουργία μιας συστοιχίας είναι elasticsearch. Το όνομα παίζει σημαντικό ρόλο καθώς κατά την δημιουργία των κόμβων τους δίνεται η πληροφορία σε ποια συστοιχία ανήκουν με βάση το όνομα αυτών. Για αυτό το λόγο καλό είναι να αποφεύγεται η επαναχρησιμοποίηση ίδιων ονομάτων καθώς αυτό μπορεί να οδηγήσει σε εσφαλμένες αναθέσεις κόμβων σε συστοιχίες. Διάφοροι συνδυασμοί είναι δυνατοί μπορούμε να έχουμε συστοιχία με μόνο έναν κόμβο ή πολλές ανεξάρτητες συστοιχίες που η κάθε μια έχει το δικό της πλήθος κόμβων.

Κόμβος: είναι ένας μεμονωμένος διακομιστής και αποτελεί κομμάτι της συστοιχίας. Κάθε κόμβος αποθηκεύει δεδομένα και συμμετέχει στην ευρετηριοποίηση και τις αναζητήσεις. Κάθε κόμβος όπως και η συστοιχία χαρακτηρίζεται μοναδικά με ένα όνομα που αρχικά είναι ένα τυχαίο αλφαριθμητικό κατά την εκκίνηση και στην συνέχεια μπορεί να τροποποιηθεί. Το όνομα αυτό είναι σημαντικό για λόγους διαχείρισης. Οι κόμβοι κατά την εκκίνηση τους είναι ρυθμισμένα να μπουν σε ομάδα συστοιχίας με το όνομα elastisearch ωστόσο μπορούμε να ρυθμίσουμε και να «καθοδηγήσουμε» ένα κόμβο σε ποια ομάδα να μπει. Δεν υπάρχει

περιορισμός πόσοι κόμβοι μπορούν να μπουν σε μια συστοιχία. Αν στο σύστημα μας δεν έχουμε κανένα κόμβο η εκκίνηση ενός θα δημιουργήσει μια συστοιχία με έναν κόμβο με το όνομα elasticsearch.
Ευρετήριο: είναι μια συλλογή από αρχεία που χαρακτηρίζονται από όμοια ή σχεδόν όμοια χαρακτηριστικά. Και αυτό γιατί μπορούμε να έχουμε αρχεία τα οποία δεν έχουν πληροφορίες για κάποια συγκεκριμένα πεδία. Όπως οι συστοιχίες και οι κόμβοι έτσι και τα ευρετήρια χαρακτηρίζονται μοναδικά από το όνομα τους, το οποίο πρέπει να είναι με πεζά γράμματα, και για κάθε λειτουργία που θέλουμε να εκτελέσουμε πάνω στο ευρετήριο όπως αναζήτηση, διαγραφή, ευρετηριοποίηση κλπ. πρέπει να χρησιμοποιήσουμε το όνομα που το προσδιορίζει μοναδικά. Σε μια συστοιχία μπορούμε να ορίσουμε όσα ευρετήρια θέλουμε. Για παράδειγμα ευρετήρια που κρατάνε δεδομένα σχετικά με ένα σύστημα όπως χρήστες, πελάτες, συναλλαγές κα.
Χαρτογράφηση: σε ένα index μπορούμε να ορίσουμε ένα ή και περισσότερα types. Η χαρτογράφηση είναι στην ουσία λογική κατηγοριοποίηση/διαχωρισμός του index μας του οποίου η σημασιολογία είναι καθαρά δικιά μας απόφαση. Γενικά η χαρτογράφηση ορίζεται για αρχεία που έχουν κοινά πεδία. Σκοπός της χαρτογράφησης είναι να κάνουμε γνωστό στο elasticsearch τι είδους αρχεία και με τι πεδία να περιμένει για εισαγωγή ώστε και εμείς στην συνέχεια να μπορούμε να τα επεξεργαστούμε και ανακτήσουμε με ευκολία και χωρίς απρόσμενα αποτελέσματα.
Αρχείο: το αρχείο είναι η βασική μονάδα πληροφορίας/δεδομένων που μπορεί να ευρετηριοποιηθεί. Τα documents αναπαρίστανται σε μορφή JSON που είναι μια ευρέως χρησιμοποιουμένη μορφή ανταλλαγής δεδομένων ιστού. Σε ένα index μπορούμε να αποθηκεύσουμε όσα documents θέλουμε. Ενώ τα documents εννοιολογικά τοποθετούνται σε ένα index, στην πραγματικότητα το document πρέπει να ευρετηριοποιηθεί/ανατεθεί σε ένα type μέσα σε ένα index.
Θραύσματα & Αντίγραφα: Ένα ευρετήριο μπορεί να περιέχει τέτοιο μέγεθος δεδομένων το οποίο ξεπερνάει την χωρητικότητα της μνήμης ενός μεμονωμένου κόμβου. Παραδείγματος χάριν ένα μεμονωμένο ευρετήριο μπορεί να περιέχει δισεκατομμύρια αρχεία συνολικού όγκου μεγαλύτερα του 1 TB με πιθανά αποτελέσματα να μην χωράει στον δίσκο ενός μεμονωμένου κόμβου ή η ανταπόκριση του συστήματος σε αιτήματα αναζήτησης να είναι πολύ αργή. Για την λύση αυτού του προβλήματος το elasticsearch παρέχει την δυνατότητα διαχωρισμού ενός ευρετήριου σε τμήματα τα οποία ονομάζονται θραύσματα. Κατά την δημιουργία του ευρετηρίου μπορούμε να προσδιορίσουμε σε πόσα θραύσματα θέλουμε να το διαχωρίσουμε. Κάθε θραύσμα μπορεί να θεωρηθεί σαν ένα αυτόνομο και πλήρες λειτουργικό ευρετήριο το οποίο αποτελεί μέρος του κόμβου σε μια συστοιχία. Το elasticsearch παρέχει την δυνατότητα να δημιουργήσουμε αντίγραφα των θραυσμάτων μας, τα αντίγραφα αυτά ονομάζονται αντίγραφα και ο σκοπός τους είναι να καλύπτουν περιπτώσεις βλαβών αν για οποιονδήποτε λόγο κάποιο από τα ευρετήρια μας ή κόμβος δεν είναι διαθέσιμο για να παρέχει τα δεδομένα που είναι αποθηκευμένα σε αυτά.
Τα θραύσματα είναι χρήσιμα για δυο λόγους:

1. Δίνουν την δυνατότητα διαχωρισμού του μεγέθους των δεδομένων
2. Παρέχουν την δυνατότητα κατανομής και παραλληλοποίησης των διεργασιών κάτι που αυξάνει την απόδοση τους.

Τα αντίγραφα είναι σημαντικές για δυο κύριους λόγους:

1. Παρέχει διαθεσιμότητα των δεδομένων σε περίπτωση βλάβης. Για αυτό είναι σημαντικό τα αντίγραφα να τοποθετούνται σε διαφορετικό κόμβο από αυτό του οποίου έγιναν αντίγραφα.
2. Δίνουν την δυνατότητα κλιμάκωσης της αναζήτησης καθώς η διεργασίες μπορούν να γίνουν παράλληλα.

3.3.2Τρόπος λειτουργίας

Ενώ το elasticsearch παρέχει APIs τα οποία είναι πολύ εύκολα στην χρήση και χωρίς πολλές γνώσεις και κόπο μπορούμε να αρχίσουμε να δημιουργούμε φοβερά πράγματα με αυτό καλό είναι να έχουμε ιδέα τι κρύβεται κάτω από αυτή την τεχνολογία, τους αλγορίθμους και δομές που χρησιμοποιεί και γενικές πρακτικές. Με αυτές τις γνώσεις μπορούμε να πετύχουμε αποτελέσματα τα οποία θα καλύπτουν όλες τις δυνατότητες της τεχνολογίας αυτής και θα βελτιώσουν την εμπειρία του χρήστη ως προς την αναζήτηση και παράλληλα το σύστημα μας θα είναι αποδοτικό και αξιόπιστο. Ας αρχίσουμε με την βασική μορφή του ευρετηρίου, το αντεστραμμένο ευρετήριο. Ενώ είναι μια πολύπλευρη δομή δεδομένων, είναι εύκολη στην κατανόηση και χρήση, παρακάτω θα δούμε την χρήση και την κατασκευή του. Τώρα που έχουμε ορίσει την τεχνολογία πάνω στην οποία βασίζονται πολλά από αυτά που θα αναφερθούν παρακάτω είναι :

- Πως πραγματοποιούνται απλές αναζητήσεις
- Ποιοι τύποι αναζητήσεων μπορούν να εκτελεστούν αποδοτικά και γιατί είναι απαραίτητη η τροποποίηση προβλημάτων στην μορφή string-prefix
- Γιατί η προ επεξεργασία του κειμένου είναι σημαντική
- Πως τα ευρετήρια δημιουργούνται σε τμήματα και πως αυτό επηρεάζει τι αναζητήσεις
- Τα ευρετήρια και η τμηματοποίηση αυτών στο elasticsearch

Ας προχωρήσουμε στην περιγραφή του αντεστραμμένου ευρετηρίου. Το αντεστραμμένο ευρετήριο αντιστοιχίζει όρους σε αρχεία τα οποία περιέχουν τον αντίστοιχο όρο(όπως και την θέση τους μέσα στα αρχεία). Οι όροι αποθηκεύονται σε λεξιλόγιο κάτι που καθιστά την εύρεση τους γρήγορη όπως και την ύπαρξη τους στα διαθέσιμα αρχεία. Αυτή η προσέγγιση είναι το ακριβώς αντίθετο του προς τα εμπρός ευρετηρίου (forward index) που αντιστοιχίζει αρχεία σε όρους. Στην συνέχεια πραγματοποιείται μια απλή αναζήτηση με πολλαπλούς όρους αναζητώντας την ύπαρξη των αντίστοιχων όρων και παίρνουμε την ένωση ή τομή των σετ των αρχείων στα οποία υπάρχουν οι επιθυμητοί όροι για να καταλήξουμε με την τελική λίστα των αρχείων που ταιριάζουν στην αναζήτηση μας. Σύνθετες αναζητήσεις φυσικά είναι πιο περίπλοκες αλλά η προσέγγιση για την εκτέλεση τους είναι ακριβώς ίδια με αυτήν που μόλις περιγράφηκε.

Συνεπώς μπορούμε να συμπεράνουμε πως οι όροι του ευρετηρίου είναι η μονάδα αναζήτησης μας. Επομένως οι όροι που παράγουμε και τοποθετούμε στο ευρετήριο μας καθορίζουν τα είδη των αναζητήσεων που μπορούμε να διεξάγουμε αποτελεσματικά. Στην γενική μορφή μπορούμε αποτελεσματικά να διεξάγουμε αναζήτηση όρων των οποίων έχουμε το πρόθεμα. Σε άλλες περιπτώσεις με πιο απαιτητικές αναζητήσεις απαραίτητη είναι η προ επεξεργασία των κειμένων όπως αντιστροφή των όρων, δημιουργία n-grams δηλαδή διαχωρισμό του όρου σε συλλαβές, διαχωρισμό σύνθετων λέξεων κα. Κατά την κατασκευή των αντεστραμμένων ευρετηρίων προτεραιότητα έχουν η ταχύτητα αναζήτησης, πόσο συμπαγή είναι τα ευρετήρια και η ταχύτητα ευρετηριοποίησης καθώς και ο χρόνος απόκρισης. Η ταχύτητα αναζήτησης και η σύμπτυξη του ευρετηρίου είναι δυο μεγέθη συσχετισμένα καθώς όταν έχουμε μικρά ευρετήρια λιγότερα δεδομένα πρέπει να ελεγχθούν και περισσότερα θα χωρέσουν στην μνήμη.

Για την σύμπτυξη των ευρετηρίων χρησιμοποιούνται διάφορες τεχνικές συμπίεσης. Ωστόσο η συμπίεση έχει ως αποτέλεσμα την μη αποτελεσματική ενημέρωση των υπαρχόντων δεδομένων. Κάτι που στην περίπτωση μας δεν γίνεται ποτέ. Το Lucene δημιουργεί αρχεία ευρετηρίου των οποίων η τροποποίηση δεν είναι δυνατή σε αντίθεση με τα B-trees. Η λύση στην ανανέωση είναι η διαγραφή. Όταν διαγράφουμε ένα αρχείο από το ευρετήριο μας, το αρχείο μπαίνει σε ένα ειδικό αρχείο διαγραφής (στην ουσία ένα bitmap) που είναι εύκολη η ανανέωση του σε αντίθεση με τα ευρετήρια. Επομένως η πράξη της ανανέωσης είναι διαγραφή ακολουθούμενη από επανεισαγωγή του αρχείου. Κατά την εισαγωγή αρχείων οι αλλαγές

αρχικά αποθηκεύονται σε έναν ρυθμιστής και στην συνέχεια γράφονται στον δίσκο. Το πότε θα γίνει αυτό εξαρτάται από διάφορους παράγοντες όπως πόσο γρήγορα θέλουμε οι αλλαγές να γίνουν ορατές, από την διαθέσιμη μνήμη, την διαθεσιμότητα εισόδων/εξόδων κα. Τα εισαχθέντα αρχεία αποτελούν ένα τμήμα (segment) του ευρετηρίου η επεξήγηση των οποίων ακολουθεί παρακάτω. Τα ευρετήρια αποτελούνται από ένα ή περισσότερα αμετάβλητα τμήματα. Κατά την διεργασία της αναζήτησης, το Lucene πραγματοποιεί την αναζήτηση σε όλα τα διαθέσιμα τμήματα και συγχωνεύει όλα τα αποτελέσματα από το καθένα. Όπως είναι αναμενόμενο η αύξηση του πλήθους των τμημάτων έχει ως αποτέλεσμα την αύξηση του χρόνου αναζήτησης και για αυτό το λόγο το Lucene κάνει συγχώνευση μερικών με βάση κάποιες αρχές κατά την δημιουργία νέων τμημάτων.

Κατά την συγχώνευση αρχεία που έχουν σημειωθεί ως διαγραμμένα απορρίπτονται ολοκληρωτικά. Πριν τα τμήματα εγγραφούν στον δίσκο, οι αλλαγές αποθηκεύονται στον ρυθμιστή. Σε προηγούμενες εκδόσεις του Lucene κάθε αρχείο που εισάγονταν αποτελούσε ένα μεμονωμένο τμήμα και όλα συγχωνεύονταν κατά την εγγραφή. Τώρα υπάρχει η δυνατότητα δημιουργίας μεγαλύτερων τμημάτων από μεγαλύτερο πλήθος αρχείων και αυτά μπορούν να υπάρχουν ένα για κάθε νήμα αυξάνοντας έτσι την απόδοση της ευρετηριοποίησης λόγω της δυνατότητας παραλληλοποίησης της εγγραφής. Κατά την δημιουργία νέων τμημάτων συγκεκριμένα τμήματα της κρυφής μνήμης καταργούνται κάτι το οποίο μπορεί να επηρεάσει αρνητικά την αναζήτηση. Πληροφορίες στην κρυφή μνήμη όπως πεδία και φίλτρα είναι διαφορετικά για κάθε τμήμα. Οι εγγραφές κατά κύριο λόγο ενεργοποιούνται από τις συχνές ανανεώσεις των ευρετηρίων. Όταν τα τμήματα εγγραφούν τότε είναι διαθέσιμα για αναζήτηση και ενώ οι εγγραφές δεν είναι τόσο ακριβές σε υπολογιστική ισχύ όσο οι δεσμεύσεις (commits). Τα αποτελέσματα είναι η δημιουργία νέων τμημάτων, κατάργηση τμημάτων της κρυφής μνήμης και πιθανόν να πυροδοτήσουν κάποια συγχώνευση.

Τώρα στο elasticsearch ένα ευρετήριο αποτελείται από ένα ή περισσότερα θραύσματα τα οποία μπορούν να έχουν κανένα ή περισσότερα αντίγραφα. Όλα τα προαναφερθέντα το κάθε ένα είναι ευρετήριο Lucene. Αυτό σημαίνει πως ένα ευρετήριο του elasticsearch αποτελείται από πολλά ευρετήρια Lucene και το καθένα από αυτά με την σειρά τους αποτελούνται από τμήματα. Επομένως όταν κάνουμε αναζήτηση σε ένα ευρετήριο του elasticsearch στην ουσία κάνουμε αναζήτηση σε όλα τα θραύσματα άρα και όλα τα τμήματα και τέλος τα αποτελέσματα συγχωνεύονται. Το ίδιο ισχύει αν κάνουμε αναζήτηση σε πολλά ευρετήρια. Στην ουσία αναζήτηση σε δυο ευρετήρια με ένα θραύσμα το κάθε ένα είναι σχεδόν το ίδιο με το να γίνει αναζήτηση σε ένα ευρετήριο το οποίο αποτελείται από δυο θραύσματα γιατί από κάτω και στις δυο περιπτώσεις γίνεται αναζήτηση σε Lucene ευρετήρια. Το θραύσμα αποτελεί βασική μονάδα κλιμάκωσης για το elasticsearch. Καθώς τα αρχεία εισάγονται στο ευρετήριο δρομολογούνται σε θραύσματα. Ο προκαθορισμένος τρόπος είναι με round-robin και βασίζεται στο hash του αναγνωριστικού του αρχείου. Είναι σημαντικό να τονίσουμε πως το πλήθος των θραυσμάτων ορίζεται κατά την δημιουργία του ευρετηρίου και δεν μπορεί να μεταβληθεί στην συνέχεια.

Για να συνοψίσουμε τα σημαντικά σημεία των όσων προαναφέρθηκαν σχετικά με την δημιουργία, ανανέωση και αναζήτηση των ευρετηρίων είναι:

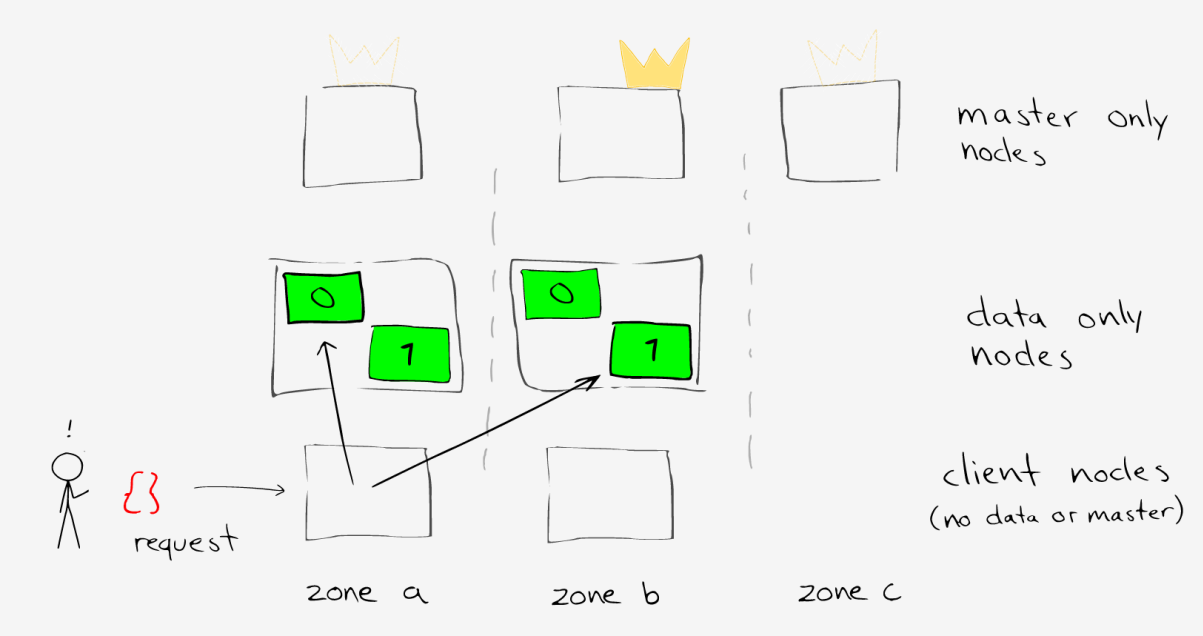
- Η προ επεξεργασία των αρχείων είναι σημαντική καθώς από το τι δεδομένα θα ευρετηριοποιήσουμε θα προέλθουν οι δυνατότητες των αναζητήσεων μας.
- Τα ευρετήρια δημιουργούνται αρχικά στην προσωρινή μνήμη του συστήματος και ανά τακτά χρονικά διαστήματα γίνονται εγγραφές σε τμήματα.
- Τα τμήματα δεν μπορούν να τροποποιηθούν/μεταβληθούν.
- Τα διαγραμμένα αρχεία «σημαδεύονται» και κατά την εγγραφή διαγράφονται ολοκληρωτικά.
- Ένα ευρετήριο αποτελείται από πολλά τμήματα πάνω σε όλα πραγματοποιούνται οι αναζητήσεις και στην συνέχεια τα αποτελέσματα συγχωνεύονται.

- Τα τμήματα συγχωνεύονται κάποιες φορές μετά από εγγραφή.
- Τα πεδία και φίλτρα είναι διαφορετικά για κάθε τμήμα.

Τώρα που έχουμε μια ιδέα τι γίνεται στα χαμηλά επίπεδα αυτής της τεχνολογίας μπορούμε να προχωρήσουμε παρακάτω και να δούμε πως πραγματοποιείται η διεργασία της ευρετηριοποίησης και αναζήτησης σε υψηλότερα επίπεδα και την πορεία των αιτήσεων μέχρι να φτάσουν στο επίπεδο που σταματήσαμε σε αυτό το κομμάτι.

Παρακάτω θα περιγραφούν οι διαδικασίες αποστολής αιτήσεων ευτερηριοποίησης και αναζήτησης. Πρώτα θα δούμε το στάδιο αποδοχής της αίτησης από τον κόμβο και ο ρόλος του ως συντονιστή. Στην συνέχεια ακολουθεί η δρομολόγηση των αιτήσεων στα κύρια θραύσματα για αιτήσεις ευτερηριοποίησης και των διαμοιρασμό φορτίων στα αντίγραφα για αιτήσεις αναζήτησης. Αφού οι αιτήσεις δρομολογηθούν στα σωστά θραύσματα, θα δούμε τι γίνεται σε αυτό το επίπεδο όπως τι αποτελεί μια επιτυχή διεργασία ευρετηριοποίησης και αλλαγές απαραίτητες για την μετατροπή αρχείων και αναζητήσεων στα αντίστοιχα του Lucene. Επίσης για την διαδικασία της αναζήτησης θα αναφέρουμε την διαδικασία διαμοίρασης/συγκέντρωσης.

Για να περιγράψουμε την λειτουργία πρέπει πρώτα να περιγράψουμε την τοπολογία μιας συστοιχίας. Για παράδειγμα έχουμε ένα ευρετήριο Elasticsearch το οποίο είναι χωρισμένο σε «ζώνες» διαθεσιμότητας. Κάθε ζώνη έχει ένα κύριο κόμβο και έχουμε δυο θραύσματα αντίγραφα διαθέσιμα σε δυο από τις τρεις ζώνες. Τέλος έχουμε δυο κόμβους πελάτες μέσω των οποίων περνάνε οι αιτήσεις.

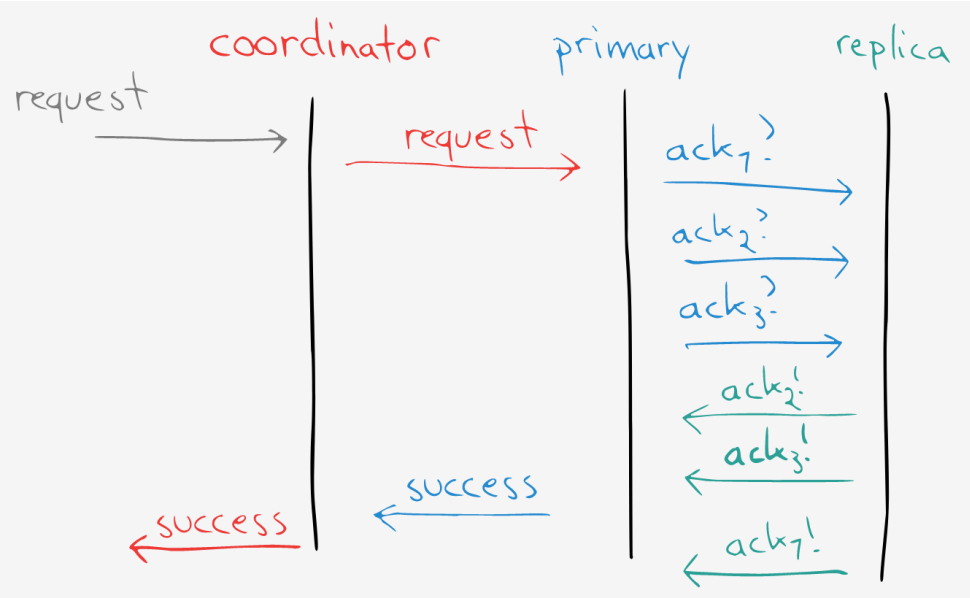


Εικόνα 7. Παράδειγμα τοπολογίας μιας συστοιχίας Elasticsearch με ένα ευρετήριο

Διαφορετικοί κόμβοι μπορούν να έχουν διαφορετικούς ρόλους όπως και ιδιότητες, Στο παράδειγμα μας έχουμε κόμβους αφέντης, δεδομένων και πελάτες σε διάφορες ζώνες. Εδώ θα εστιάσουμε κυρίως στους κόμβους δεδομένων. Όταν γίνεται αποστολή μιας αίτησης σε έναν κόμβο, αυτόματα αυτό γίνεται ο συντονιστής της συγκεκριμένης αίτησης. Στις αρμοδιότητες του είναι να αποφασίσει σε ποιους κόμβους και θραύσματα θα κάνει προώθηση της αίτησης, πως θα γίνει η συγχώνευση των αποτελεσμάτων από διαφορετικούς κόμβους καθώς και πότε να ορίσει μια αίτηση ως ολοκληρωμένη. Στην περίπτωση μας οι συντονιστές είναι οι κόμβοι πελάτες. Ωστόσο για την διεργασία αυτή είναι απαραίτητη η γνώση της κατάστασης μιας συστοιχίας κάτι το οποίο είναι διαθέσιμο σε όλους τους κόμβους μιας συστοιχίας. Μερικές από τις πληροφορίες που περιέχει είναι ο πίνακας δρομολόγησης των θραυσμάτων (ποιοι κόμβοι περιέχουν ποια ευρετήρια και θραύσματα), μεταδεδομένα για κάθε κόμβο και τις

χαρτογραφήσεις των ευρετηρίων. Για κάθε αίτηση, ο συντονισμός στην ουσία είναι η επιλογή των αντιγράφων των θραυσμάτων στα οποία θα προωθηθεί η αίτηση για περαιτέρω επεξεργασία. Η επιλογή μπορεί να είναι τυχαία ή να επηρεαστεί από τυχόν προτιμήσεις της αίτησης.

Μια αίτηση ευρετηριοποίησης (δημιουργία, ανανέωση, διαγραφή) είναι λίγο διαφορετική. Σε αυτή την περίπτωση οι αιτήσεις πρέπει να προωθηθούν στα κύρια θραύσματα καθώς και σε μερικά από τα αντίγραφα. Η δρομολόγηση/προώθηση καθορίζει ποια αρχεία πάνε που και επομένως αποτελεί σημαντικό τμήμα της διαχείρισης και των δυο ειδών αιτήσεων (ευρετηριοποίησης, αναζήτησης) εσωτερικά στο elasticsearch. Τα αρχεία δρομολογούνται με βάση ένα κλειδί δρομολόγησης και τοποθετούνται σε θραύσματα με αριθμό που προκύπτει από τον τύπο hash(key) mod n όπου n είναι το πλήθος των θραυσμάτων στο ευρετήριο. Κατά τον σχεδιασμό των θραυσμάτων πολλές φορές το πιο σημαντικό κομμάτι είναι η επιλογή του κλειδιού. Όταν γίνεται η δρομολόγηση μιας αίτησης ευρετηριοποίησης, γίνεται προώθηση αυτής στο κύριο θραύσμα. Κάθε θραύσμα έχει ακριβώς ένα κύριο θραύσμα και κανένα ή περισσότερα αντίγραφα. Το κύριο θραύσμα θα λειτουργεί σαν συντονιστής για διεργασία για το συγκεκριμένο θραύσμα. Θα αποστείλει τις διεργασίες ευρετηριοποίησης στα αντίστοιχα αντίγραφα και θα περιμένει μέχρι ένας ικανοποιητικός αριθμός αυτών επιστρέψει μήνυμα επιτυχίας. Όταν γίνει αυτό το κύριο θραύσμα θα επιστρέψει μήνυμα επιτυχίας στον αρχικό συντονιστή από τον οποίον ξεκίνησε η όλη διαδικασία. Στην περίπτωση που ο συντονιστής πρέπει να στείλει διεργασίες σε διαφορετικά κύρια θραύσματα αυτό γίνεται παράλληλα και όταν όλες οι διεργασίες γυρίσουν με μήνυμα επιτυχίας η αρχική αίτηση ολοκληρώνεται.



Εικόνα 8. Πορεία των αιτήσεων στο κύριο θραύσμα και στα αντίγραφα

Ωστόσο χαρακτηριστικά τα οποία αναφέρθηκαν στο προηγούμενο τμήμα όπως η ταχύτητα αναζήτησης και ευρετηριοποίησης και άλλα για την δημιουργία ευρετηρίων όπως και εσωτερικά του Lucene πχ αμετάβλητα τμήματα έρχονται σε αντίθεση με τις απαιτήσεις μας για την διεργασία της ευρετηριοποίησης που απαιτεί ασφάλεια, διάρκεια και γρήγορη επιστροφή των αναγνωριστικών. Η λύση του elasticsearch στο πρόβλημα αυτό είναι το αρχείο καταγραφής συναλλαγών όταν οι διεργασίες γραφτούν σε αυτό το αρχείο πλέον χαρακτηρίζονται ως επιτυχείς χωρίς αυτό να σημαίνει πως το αρχείο είναι διαθέσιμο για αναζήτηση από κάποιο τμήμα του ευρετηρίου.

Τώρα το Lucene δεν έχει την έννοια της χαρτογράφησης, τα αρχεία δεν έχουν τύπο και έχουν αυθαίρετο αριθμό πεδίων. Τα πεδία αυτά έχουν συγκεκριμένο τύπο και ιδιότητες. Η

χαρτογράφηση είναι μια αφαίρεση για την περιγραφή της μετατροπής ενός αρχείου σε αρχείο Lucene. Η χαρτογράφηση έχει έννοιες όπως τύπος, ιδιότητες, πολλαπλά πεδία κα. Το Lucene ωστόσο δεν γνωρίζει τίποτα από όλα αυτά. Παρ όλα αυτά δεν υπάρχει κανένα πρόβλημα όσο τα αρχεία που δημιουργούνται είναι κάτι το οποίο μπορεί να διαχειριστεί από το Lucene. Για να συνοψίσουμε την ροή μιας αίτησης ευρετηριοποίησης:

1. Ο κόμβος που δέχεται την αίτηση γίνεται ο συντονιστής. Με την συμβολή των χαρτογραφήσεων αποφασίζει σε ποια θραύσματα να προωθήσει την αίτηση.
2. Η αίτηση προωθείται στο κύριο θραύσμα.
3. Το κύριο θραύσμα γράφει την αίτηση στο αρχείο καταγραφής συναλλαγών και προωθεί την αίτηση στα αντίγραφα.
4. Όταν ικανοποιητικός αριθμός αντιγράφων επιστρέψουν μήνυμα επιτυχίας, το κύριο επιστρέφει μήνυμα επιτυχίας στον αρχικό κόμβο.
5. Ο συντονιστής επιστρέφει μήνυμα επιτυχίας όταν όλες οι αιτήσεις, σε περίπτωση πολλαπλών, επιστρέψουν σε αυτόν με επιτυχία.

Καθ' όλη αυτή την διάρκεια κάθε θραύσμα επεξεργάζεται συνεχώς τα αρχεία που έχει σε αναμονή μετατρέποντας τα σε αντίστοιχα αρχεία Lucene, Στην συνέχεια αυτά προθέτονται στον ρυθμιστή και τελικά καταλήγουν σε τμήματα.

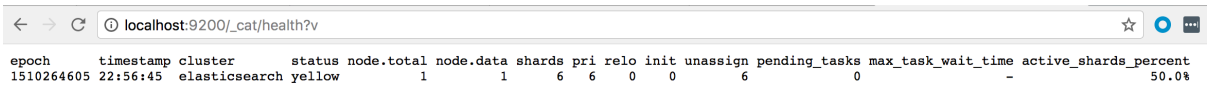
Μια αίτηση αναζήτησης έχει ομοιότητες στον τρόπο διαχείρισης της με μια αίτηση ευρετηριοποίησης ταυτόχρονα ωστόσο υπάρχουν και διαφορές. Για παράδειγμα μια αίτηση αναζήτησης πρέπει και αυτή να δρομολογηθεί. Ωστόσο η δρομολόγηση μπορεί να γίνει σε όλα τα διακριτά τμήματα ή να προσδιοριστεί ένα συγκεκριμένο. Όταν εντοπιστούν τα σχετικά τμήματα, ο συντονιστής θα διαλέξει από τα διαθέσιμα αντίγραφα και θα προσπαθήσει να κατανέμει τον φόρτο εργασίας. Μια αίτηση αναζήτησης για να μπορεί να εκτελεστεί πρέπει πρώτα να μετατραπεί και να προσαρμοστεί σε Lucene ερώτηση (query). Ωστόσο υπάρχουν εξαιρέσεις όπως για παράδειγμα οι χαρτογραφήσεις δεν είναι κάτι κατανοήσιμο από το Lucene ωστόσο κάποιες elasticsearch αναζητήσεις κάνουν χρήση αυτών. Επομένως πρέπει να προσαρμοστούν σε μορφή κατανοητή από το Lucene. Μετά από αυτό το στάδιο έχουμε μια ερώτηση έτοιμη να εκτελεστεί πάνω σε όλα τα θραύσματα άρα είμαστε στην πρώτη φάση, αναζήτηση και επιστροφή των αποτελεσμάτων. Είναι σημαντικό να τονίσουμε πως οι αναζητήσεις πραγματοποιούνται σε πολλαπλά ανεξάρτητα τμήματα και στην συνέχεια συγχωνεύονται. Υπάρχουν πολλές πληροφορίες στην κρυφή μνήμη για κάθε τμήμα που μπορούν να βελτιώσουν την εκτέλεση της αναζήτησης. Ωστόσο οι αναζητήσεις δεν αποθηκεύονται στην κρυφή μνήμη. Οπότε για αναζητήσεις και πληροφορίες που δεν είναι διαθέσιμες στην κρυφή μνήμη η αναζήτηση θα χρειαστεί να φτάσει στα αντεστραμμένα ευρετήρια. Σε αυτές τις περιπτώσεις, τα σχετικά αποτελέσματα θα μπορούσαν να αποθηκευτούν στην κρυφή μνήμη του λειτουργικού συστήματος. Συμπερασματικά, με την χρήση φίλτρων και συσσωματώσεων στην ουσία χειριζόμαστε bitmaps για τιμές που έχουμε ήδη αποθηκεύσει στην κρυφή μνήμη και αυτό έχει ως αποτέλεσμα το elasticsearch να είναι τόσο γρήγορο. Αφού κάθε θραύσμα επιστρέψει τα αποτελέσματα του, ο συντονιστής θα τα συγχωνεύσει. Κατά την συγχώνευση θα βρεθούν τα τελικά επιθυμητά αποτελέσματα και στην συνέχεια ο συντονιστής θα πάρει τα πλήρη αρχεία από τα αντίστοιχα θραύσματα. Αυτή η διαδικασία ωστόσο μπορεί να γίνει και σε ένα βήμα αν τα αρχεία δεν είναι μεγάλα και δεν έχουμε μεγάλο αριθμό από θραύσματα.

Να συνοψίσουμε και να κλείσουμε και αυτό το κομμάτι στο οποίο είδαμε την δρομολόγηση των αιτήσεων σε θραύσματα με την βοήθεια της κατάστασης της συστοιχίας και των χαρτογραφήσεων. Στην συνέχεια πως τα κύρια θραύσματα συντονίζουν την περεταίρω προώθηση στα αντίγραφα και πως τα αρχεία καταγραφής συναλλαγών ισορροπούν την αντοχή και τις έγκαιρες απαντήσεις. Τέλος ακολουθήσαμε την ροή των αιτήσεων αναζήτησης από τα στάδια δρομολόγησης, ισορρόπησης, διαμοίρασης, μετατροπής των ερωτήσεων, συλλογής των αποτελεσμάτων, συγχώνευσης και την τελική απόκτηση των επιθυμητών αρχείων.

3.3.3Σύντομη περιγραφή εκκίνησης

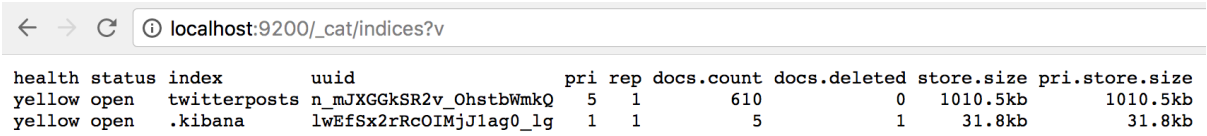
Όπως προηγουμένως έτσι και εδώ για να ξεκινήσουμε χρειαζόμαστε εγκατεστημένη JAVA 8 και την μεταβλητή JAVA_HOME ορισμένη. Όπως επίσης και οι επιλογές εγκατάστασης του elasticsearch ποικίλουν για παράδειγμα zip/tar.gz, deb (Debian, Ubuntu και άλλα Debian based συστήματα), rpm, msi και docker image. Το elasticsearch έρχεται με τις αναγκαίες ρυθμίσεις προκαθορισμένες και η αλλαγή τους είναι επίσης δυνατή σε μια συστοιχία που τρέχει ήδη με την χρήση του Cluster update settings API. Το elasticsearch έχει δυο αρχεία διαμόρφωσης το elasticsearch.yml το οποίο περιέχει τις ρυθμίσεις λειτουργίας του εργαλείου και το log4j2.properties το οποίο περιέχει ρυθμίσεις για τα αρχεία καταγραφής του elasticsearch. Υπάρχουν ωστόσο μερικές ρυθμίσεις που είναι σημαντικό να ρυθμιστούν πριν την εκκίνηση του elasticsearch και αυτό γιατί σχετίζονται με τις συστοιχίες και τους κόμβους όπως για παράδειγμα για να μπορούν οι κόμβοι να εντοπίσουν σε ποια συστοιχία πρέπει να μπουν. Μερικά παραδείγματα είναι τα cluster.name, node.name, network.host, path.data και άλλα. Υπάρχουν πολύ περισσότερες ρυθμίσεις που σχετίζονται με ασφάλεια και ρυθμίσεις συστήματος που ξεφεύγουν από τον σκοπό της πτυχιακής και έχουν σχέση με περιβάλλον παραγωγής (production environments) και για αυτό τον λόγο δεν θα αναλυθούν.

Και τώρα πρέπει να έχουμε ένα elasticsearch να τρέχει πάνω στο οποίο μπορούμε να δοκιμάσουμε τις διάφορες δυνατότητες και λειτουργίες που παρέχει. Το elasticsearch παρέχει ένα περιεκτικό και ισχυρό REST API με την χρήση του οποίου μπορούμε να αλληλοεπιδράσουμε με αυτό. Μερικές από τις δυνατές λειτουργίες είναι ο έλεγχος της κατάστασης και των στατιστικών των συστοιχιών, κόμβων και ευρετηρίων. Η διαχείριση των προαναφερθέντων τμημάτων καθώς και των δεδομένων και μεταδεδομένων που αυτά περιέχουν. Εκτέλεση αναζητήσεων και διεργασιών όπως δημιουργία, ενημέρωση και διαγραφή καθώς και πιο εξειδικευμένων όπως σελιδοποίηση, ταξινόμηση, φιλτράρισμα και άλλες. Παρακάτω θα αναφέρω μερικά βασικά παραδείγματα για βασικές λειτουργίες. Αφού έχουμε το elasticsearch να τρέχει αυτό σημαίνει πως έχει δημιουργηθεί μιας συστοιχίας και 5 κόμβους με την προϋπόθεση πως αρχίσαμε με τις προκαθορισμένες ρυθμίσεις φυσικά. Το *cat API* παρέχει πληροφορίες σχετικά με τις συστοιχίες, τους κόμβους, τα θραύσματα και γενικά διάφορες πληροφορίες σχετικές με το σύστημα μας και την κατάσταση του. Ο ευκολότερος τρόπος πλοήγησης στο elasticsearch προσωπικά πιστεύω είναι από το URL ωστόσο μπορεί να γίνει μέσο οποιουδήποτε εργαλείου που διαχειρίζεται HTTP εντολές όπως για παράδειγμα Postman, μπορεί ωστόσο να γίνει και από το Kibana. Το elasticsearch είναι προκαθορισμένο να τρέχει στο localhost:9200 και από κει και πέρα μπορούμε να αποκτήσουμε πρόσβαση σε όλα τα REST endpoints που παρέχονται. Πίσω στο παράδειγμα μας, αν πάμε στο *localhost:9200/_cat/health?v* θα πάρουμε ως αποτέλεσμα πληροφορίες για την κατάσταση της συστοιχίας μας και άλλες σχετικές πληροφορίες με αυτό.



Εικόνα 9. Πληροφορίες για την συστοιχία από το cat API του elasticsearch

Ένα άλλο παράδειγμα που επιστρέφει πληροφορίες σχετικά με τα ευρετήρια μας όπως πόσα είναι, τα αναγνωριστικά τους, πλήθος αρχείων, μέγεθος κλπ.
localhost:9200/_cat/indices?v



Εικόνα 10. Πληροφορίες για τα ευρετήρια από το cat API του elasticsearch

Στο *localhost:9200/_cat* μπορούμε να δούμε όλες τις δυνατές λειτουργίες αυτού του API.
Σε αρκετά γενικευμένη μορφή οι εντολές στο elasticsearch έχουν την μορφή:
<REST verb> /{index}/{type}/{id} όπου το ρήμα μπορεί να είναι ένα από τα GET, PUT, POST, DELETE κα στην συνέχεια το όνομα του ευρετηρίου πάνω στο οποίο θέλουμε να πραγματοποιήσουμε την ενέργεια, ο τύπος και τέλος το αναγνωριστικό. Για παράδειγμα η δημιουργία ενός ευρετηρίου γίνεται με την εντολή PUT localhost:9200/customer. Η τοποθέτηση ενός εγγράφου στο ευρετήριο εκτός από την εντολή PUT localhost:9200/customer/external/1 προϋποθέτει και ένα αντικείμενο JSON με το περιεχόμενο του εγγράφου.

```
{
    "account_number": 0,
    "balance": 16623,
    "firstname": "Bradshaw",
    "lastname": "Mckenzie",
    "age": 29,
    "gender": "F",
    "address": "244 Columbus Place",
    "employer": "Euron",
    "email": "bradshawmckenzie@euron.com",
    "city": "Hobucken",
    "state": "CO"
}
```

Εικόνα 11. Παράδειγμα αρχείου σε μορφή JSON.

Για την ανάκτηση δεδομένων από τα ευρετήρια μας υπάρχει διαθέσιμο το Search API. Η αναζήτηση βασίζεται σε παραμέτρους που περνάμε και η εκτέλεση μπορεί να γίνει είτε στο REST Request URI είτε στο REST Request body. Στην δεύτερη περίπτωση έχουμε την δυνατότητα να είμαστε πιο εκφραστικοί όσον αφορά την αναζήτηση μας και να έχουμε καλύτερη αναπαράσταση των JSON αντικειμένων. Ένα παράδειγμα εκτέλεσης αναζήτησης είναι GET localhost:9200/customer/ _search?q=*&sort= customer_number:asc όπου μπορούμε να δούμε το _search είναι το REST endpoint για την αναζήτηση και στην συνέχεια έχουμε το q=* που δηλώνει ότι θέλουμε όλα τα έγγραφα, sort= customer_number:asc και τέλος ταξινόμηση με βάση τον αριθμό του πελάτη σε αύξουσα σειρά. Τέλος η διαγραφή ενός ευρετηρίου μπορεί να γίνει με την εντολή DELETE localhost:9200/customer.
Και πάλι για τον τερματισμό μπορούμε να χρησιμοποιήσουμε control + C ή αν ανακτήσουμε το PID της διεργασίας και να εκτελέσουμε στο τερματικό kill –SIGTERM {PID}.

3.4Kibana

3.4.1Εισαγωγή και γρήγορη εκκίνηση

Το kibana είναι μια πλατφόρμα ανοιχτού κώδικα που στόχο έχει την ανάλυση και οπτικοποίηση δεδομένων, τα οποία αποκτά από το elasticsearch. Το kibana λειτουργεί αποκλειστικά με το elasticsearch για την αναζήτηση και την αλληλεπίδραση με τα δεδομένα

που αυτό αποθηκεύει. Παρέχει την δυνατότητα ανάλυσης και οπτικοποίηση σε πληθώρα επιλογών όπως διαγράμματα, πίνακες, χάρτες κα. Το kibana καθιστά εύκολη την κατανόηση δεδομένων μεγάλου όγκου καθώς μπορούμε να τρέξουμε ερωτήσεις στο elasticsearch από την διεπαφή του kibana και να δούμε απευθείας τα αποτελέσματα στους πίνακες που έχουμε φτιάξει. Για την εγκατάσταση και στην περίπτωση του kibana έχουμε διαφορετικές διανομές ανάλογα με το σύστημα μας. Στο ιδανικό σύστημα αν θέλουμε να αποφύγουμε οποιουδήποτε είδους ασυμβατότητες και σφάλματα οι εκδόσεις του elasticsearch και του kibana πρέπει να είναι ίδιες. Σε καμία περίπτωση το kibana δεν πρέπει να έχει έκδοση μεγαλύτερη του elasticsearch. Το elasticsearch μπορεί να έχει μεγαλύτερη minor έκδοση (πχ elasticsearch-5.1.x και kibana-5.0.x) από το kibana για την περίπτωση που κάνουμε αναβάθμιση το σύστημα μας και πρώτα αναβαθμίζεται το elasticsearch και στην συνέχεια το kibana. Ενώ η χρήση διαφορετικών patch εκδόσεων (δηλαδή 5.1.1 και 5.1.2) υποστηρίζεται ωστόσο καλό είναι οι δυο εκδόσεις να μην διαφέρουν.

Για την εκκίνηση του kibana πρέπει στο τερματικό να προηγηθούμε στο φάκελο kibana-x.y.z/bin και εκτελέσουμε ./kibana. Το kibana κατά την εκκίνηση διαβάζει τις ρυθμίσεις λειτουργίας του από το αρχείο kibana.yml και η προκαθορισμένη διεύθυνση στην οποία είναι διαθέσιμη η εφαρμογή είναι το localhost:5601. Για οποιαδήποτε αλλαγή στις προκαθορισμένες ρυθμίσεις όπως το host ή θύρα (port) στο οποίο τρέχει η εφαρμογή, ο τομέας (domain) του elasticsearch στο οποίο θέλουμε να συνδέσουμε το kibana και άλλες σχετικές ρυθμίσεις πρέπει να ορισθούν στο αρχείο kibana.yml. Πριν αρχίσουμε να χρησιμοποιούμε το kibana πρέπει πρώτα να προσδιορίσουμε ποια ευρετήρια του elasticsearch θέλουμε να επεξεργαστούμε. Κατά την πρώτη εκκίνηση μας ζητείται να προσδιορίσουμε το όνομα του ευρετηρίου. Στην συνέχεια περνάμε στο discovery με προκαθορισμένο το πρότυπο ευρετήριο που ορίσαμε, το χρονικό διάστημα είναι ορισμένο στα 15 τελευταία λεπτά και η αναζήτηση επιστρέφει όλα τα αρχεία του ευρετηρίου μας για την συγκεκριμένη χρονική περίοδο. Από εδώ και πέρα μπορούμε να δημιουργήσουμε οπτικοποίησεις στο visualize και στην συνέχεια να τις συνδυάσουμε στους πίνακες μας στην καρτέλα dashboard. Τέλος με control + C στο τερματικό όπου τρέχει η διεργασία θα τερματίσει το kibana.

4.Διαδικασία και τρόπος εκτέλεσης της πτυχιακής

Η αρχική ιδέα της πτυχιακής ήταν η επεξεργασία, ανάλυση και εξόρυξη πληροφορίας από twitter posts με την χρήση του Hadoop. Ενώ αρχικά ολοκλήρωσα την εγκατάσταση του σε τοπικό μηχάνημα στην συνέχεια θεώρησα πως δεν είναι κάτι με το οποίο θέλω να συνεχίσω. Λόγοι που με οδήγησαν σε αυτή την απόφαση το όχι και τόσο ελκυστική τεκμηρίωση και ίσως οι δυνατότητες του όσον αφορά την επεκτασιμότητα και συνδυασμό με άλλες τεχνολογίες. Πριν βρω την «κύρια» τεχνολογία που θα είναι υπεύθυνη για την αποθήκευση των δεδομένων και την τελική παρουσίαση των αποτελεσμάτων θεώρησα πως θα μπορούσα να ξεκινήσω με την συλλογή των twitter posts. Οπότε ακολούθησαν τα απαραίτητα βήματα για την έναρξη αυτής της διαδικασίας.

4.1Twitter API και tweppy βιβλιοθήκη

Πρώτο βήμα ήταν η δημιουργία λογαριασμού στο twitter για να μπορέσω να δημιουργήσω ένα project και να αποκτήσω τα απαραίτητα κλειδιά για την επικυρωμένη σύνδεση στην υπηρεσία που παρέχει το twitter για αναζήτηση και ανάκτηση των post το οποίο είναι ένα rest endpoint το λεγόμενο twitter api. Στην συνέχεια κατέβασα και εγκατέστησα το tweepy μια από τις πιο γνωστές και πολυχρησιμοποιημένες βιβλιοθήκες της python για ανάκτηση και επεξεργασία των tweets. Το script ως στην απλή του μορφή αναζητούσε για posts τα οποία είχαν συγκεκριμένες λέξεις κλειδιά στο κείμενο τους και ήταν γραμμένα στην αγγλική γλώσσα. Στην συνέχεια αφού γινόταν η λήψη ως JSON αντικείμενο κρατούσε τα πεδία τα οποία θεώρησα απαραίτητα για περαιτέρω επεξεργασία και ίσως φαινόντουσαν χρήσιμα στην όλη ανάλυση και τέλος αποθηκεύονταν σε αρχείο. κάτι το οποίο δεν είναι αποδοτικό από θέμα οργάνωσης, χώρου και ύστερης ανάκτησης αλλά αργότερα θα χρησιμοποιούνταν τεχνολογία για τον συγκεκριμένο σκοπό. Στην συνέχεια μου έγινε γνωστό η στοίβα ELK το οποίο κάλυπτε με τα διάφορα εργαλεία του ακριβώς τις διεργασίες που ήθελα να πραγματοποιήσω όπως συλλογή και οπτικοποίηση των δεδομένων. Αφού έμαθα τις δυνατότητες και γενικές λειτουργίες όλων των εργαλείων που προσφέρει η συγκεκριμένη εταιρεία αποφάσισα πως αυτά που θα χρειαστώ είναι τα Logstash, Elasticsearch και Kibana. Το πρώτο με το οποίο ασχολήθηκα ξεκινώντας να διαβάζω τον τρόπο χρήσης του, να ανακαλύπτω τις δυνατότητες του και πως μπορώ να το προσαρμόσω στις δικές μου ανάγκες ήταν το Elasticsearch. Όταν το κατέβασα και το έκανα εγκατάσταση ήταν στην έκδοση 5.6.2 και το αναφέρω αυτό γιατί αργότερα προέκυψαν προβλήματα συμβατότητας με τα άλλα εργαλεία τα οποία θα αναφέρω αργότερα όταν φτάσω σε εκείνο το σημείο υλοποίησης.

4.2 Διαμόρφωση του Logstash

Αφού δημιούργησα μια συστοιχία με έναν κόμβο και έμαθα τις βασικές διεργασίες και τρόπους λειτουργίας του εργαλείου καθώς είναι αυτό που παρέχει δυνατότητες ευρετηριοποίησης, αναζήτησης, επεξεργασίας και ανάκτησης δεδομένων και πληροφοριών θεώρησα πως είναι η στιγμή να αρχίσω να συλλέγω πραγματικά δεδομένα με το python script να τραβάει τα tweets και στην συνέχεια να τα περνάει στο Elasticsearch. Ωστόσο ανακάλυψα πως το Logstash έχει την ίδια δυνατότητα με την χρήση του twitter προσθέτου εισόδου οπότε αποφάσισα και αυτή η διεργασία να γίνεται με το συγκεκριμένο εργαλείο. Οπότε στη συνέχεια έπρεπε να μάθω περισσότερα για αυτό και να αφήσω προς το παρόν τα άλλα εργαλεία. Η χρήση του twitter plugin, που είναι και η μοναδική πηγή δεδομένων, ήταν πολύ ξεκάθαρη στο αρχείο pipeline.conf που είναι το υπεύθυνο αρχείο για την φόρτωση των ρυθμίσεων λειτουργίας του εργαλείου στην περιοχή εισόδου χρειάστηκαν τα πεδία με τα κλειδιά αυθεντικοποίησης για την σύνδεση με το twitter api και πεδία ανάλογα με τις προτιμήσεις στο

περιεχόμενο των tweets, γλώσσας στην οποία είναι γραμμένο το κείμενο κλπ. Ένα πρόβλημα με το συγκεκριμένο πρόσθετο που αντιμετώπισα ήταν πως ενώ αναφέρονταν η δυνατότητα φιλτραρίσματος ανάλογα με την γεωγραφική τοποθεσία δεν κατάφερα να έχω τέτοια αποτελέσματα δεν γνωρίζω αν δεν υποστηρίζεται πλέον κάτι τέτοιο από το πρόσθετο ή ήταν δικό μου λάθος. Με αποτέλεσμα στην συνέχεια αν θελήσω κάτι τέτοιο να πρέπει να φιλτράρω τα tweets με δικό μου τρόπο ή με συγκεκριμένη αναζήτηση στο Elasticsearch.

```
1   #Configuration for Logstash pipeline
2   input {
3       twitter{
4           consumer_key => "[redacted]"
5           consumer_secret => "[redacted]"
6           oauth_token => "[redacted]"
7           oauth_token_secret => "[redacted]"
8           keywords => ["karlef"]
9           languages => ["en"]
10          full_tweet => true
11      }
12  }
```

Εικόνα 12. Διαμόρφωση του σταδίου εισόδου του logstash

Το επόμενο βήμα ήταν να κάνω διαμόρφωση το τμήμα εξόδου, προσωρινά χωρίς κάποια επεξεργασία στο ενδιάμεσο τμήμα φιλτραρίσματος, για τον έλεγχο της σωστής λήψης των tweets. Στην έξοδο κάνω χρήση δυο προσθέτων του stdout που τυπώνει τα αποτελέσματα της σωλήνωσης στο τερματικό ως ένα JSON αντικείμενο και του elasticsearch το οποίο όπως γίνεται κατανοητό και από το όνομα του περνάει τα δεδομένα στο Elasticsearch. Η διαμόρφωση του προσθέτου για το elasticsearch χρειάζεται τουλάχιστον το πεδίο hosts στο οποίο δηλώνουμε την διεύθυνση στην οποία τρέχει το elasticsearch και επιπλέον δήλωσα και το όνομα του ευρετηρίου για να μην δημιουργήσει κάποιο προκαθορισμένο όνομα κάτω από το οποίο θα αποθηκεύει τα tweets. Ένα ακόμα πρόβλημα που αντιμετώπισα κατά τη διαμόρφωση του logstash παρόμοιο με το πρόβλημα που προανέφερα ήταν ρυθμίσεις του elasticsearch προσθέτου οι οποίες πάλι αναφέρονταν στην τεκμηρίωση αλλά δεν κατάφερα να τις χρησιμοποιήσω. Οι ρυθμίσεις αυτές έδιναν την δυνατότητα εκτέλεσης κάποιου script πάνω στα δεδομένα πριν αυτά ολοκληρώσουν το τελικό στάδιο εξόδου και περάσουν στο elasticsearch για αποθήκευση. Συγκεκριμένα στο script ορίζεται το όνομα του αρχείου ή το ίδιο το script ανάλογα με το script_type το οποίο δέχεται 3 δυνατές τιμές inline αν στο πεδίο script έχουμε κατευθείαν τον κώδικα index αν το script είναι ευρετηριοποιημένο μέσα στο elasticsearch και τέλος file αν είναι σε ξεχωριστό αρχείο μέσα στον φάκελο του elasticsearch μέσα στο config/scripts και στο script_lang δηλώνεται η γλώσσα στην οποία είναι γραμμένο το script. Στην περίπτωση μου έβαλα το όνομα του script μου latlongcalc.py του οποίου την λειτουργία θα περιγράψω αναλυτικά παρακάτω, σαν γλώσσα python και τύπο αρχείου. Για να δω αν γίνεται όντως κλήση του script το έβαλα επιπλέον να γραφεί ένα αρχείο με τα δεδομένα εισόδου. Το αποτέλεσμα ήταν να βλέπω τα tweets να γράφονται στην έξοδο αλλά δεν γινόταν κλήση του script οπότε έπρεπε να βρω άλλο τρόπο επεξεργασίας των δεδομένων πριν ολοκληρωθούν τα στάδια του logstash για να αποφύγω κάποιου είδους ύστερη επεξεργασία των δεδομένων αφού έχουν περαστεί και αποθηκευτεί στο elasticsearch κάτι που θα έκανε την όλη διαδικασία πιο περίπλοκη.

```
119  output {
120      stdout{codec => rubydebug { metadata => true } }
121      elasticsearch {
122          hosts => ["localhost:9200"]
123          index => "twitterposts"
124      }
125  }
```

Εικόνα 13. Διαμόρφωση του σταδίου εξόδου του logstash

Σε αυτό το στάδιο άρχισα η διαμόρφωση του ενδιάμεσου σταδίου φιλτραρίσματος. Το πρώτο πρόσθετο που χρησιμοποίησα ήταν το drop με το οποίο μπορείς να απορρίψεις τα δεδομένα αν δεν πληρούν συγκεκριμένες προϋποθέσεις. Στην περίπτωση μου το χρησιμοποίησα γιατί παρατήρησα πως κάποια από τα tweets δεν περιείχαν τις λέξεις αναζήτησης στο κυρίως κείμενο αλλά μπορούσαν να βρίσκονται σε κάποιον σύνδεσμο από τις εικόνες του χρήστη και γενικά σε άσχετα πεδία. Τα επόμενα 2 χρησιμοποιούμενα πρόσθετα καλύπτουν την ανάγκη να κρατάω συγκεκριμένα πεδία από τα tweets και όχι ολόκληρα. Ωστόσο σε κάποιες περιπτώσεις είναι εμφωλευμένα κάτι το οποίο δεν είναι δυνατό με το prune στο οποίο δηλώνεις whitelist_names με πεδία που θες να κρατήσεις ή blacklist_names με πεδία τα οποία θες να παραλείψεις (δουλεύει μόνο με top-level πεδία). Με αποτέλεσμα τα εμφωλευμένα πεδία τα οποία ήθελα να κρατήσω έπρεπε να τα αντιγράψω πρώτα σε καινούργια πεδία top-level πριν απορρίψω τα πεδία κάτω από τα οποία βρίσκονταν. Στην συνέχεια κάνω την πρώτη κλήση script το οποίο υπολογίζει τις γεωγραφικές συντεταγμένες στην περίπτωση που ο χρήστης παρέχει κάποιες πληροφορίες με τις οποίες είναι δυνατό. Επομένως έπρεπε να ελέγξω αν έστω και ένα από τα πεδία τα οποία μπορούν να περιέχουν την τοποθεσία του χρήστη δεν είναι κενό. Για αυτή τη λειτουργία βασίστηκα στο αν ο χρήστης έχει ενεργοποιήσει την τοποθεσία του, έχει προσθέσει τοποθεσία στο tweet του ή έχει ενεργοποιημένη την πληροφορία χρονικής ζώνης στην οποία ανήκει. Όποιο από αυτά τα 3 πεδία δεν είναι κενό αποθηκεύεται σε προσωρινό πεδίο το οποίο θα περαστεί στο script και πριν την ολοκλήρωση του θα διαγραφεί για να μην έχουμε επανάληψη πληροφορίας. Η λύση που βρήκα για να κάνω κλήση του python script μου κατά την διαδικασία του φιλτραρίσματος ήταν η χρήση του ruby προσθέτου με το οποίο μπορείς να εκτελέσεις ένα ruby script και μεσώ αυτού του script να καλέσεις το script που επιθυμείς. Ίσως δεν είναι και η πιο έξυπνη και πιο εύχρηστη λύση αλλά ήταν από τις περισσότερο προτεινόμενες λύσεις στο ίντερνετ και λειτουργεί. Όπου ουσιαστικά καλείς από το τερματικό την εντολή και περνάς σαν παράμετρο το προσωρινό πεδίο που δημιουργήσαμε στο προηγούμενο στάδιο.

```
75      if [country]{
76          ruby{
77              code => 'event.set("loc", event.get("city")+" "+event.get("country"))'
78          }
79      }
80      else if [location]{
81          ruby{
82              code => 'event.set("loc", event.get("location").gsub(","," "))'
83          }
84      }
85      else if [timezone]{
86          ruby{
87              code => 'event.set("loc", event.get("timezone"))'
88          }
89      }
90      ruby {
91          code => 'require "open3"
92                   location = event.get("loc")
93                   cmd =  "python /Users/kar_lef/Desktop/Thesis/scripts/latlongcalc.py #{location} "
94                   stdin, stdout, stderr = Open3.popen3(cmd)
95                   event.set("[geolocation][lat]", stdout.readline.delete!("\n"))
96                   event.set("[geolocation][lon]", stdout.readline.delete!("\n"))
97                   err = stderr.read
98                   if err.to_s.empty?
99                     filter_matched(event)
100                  else
101                    event.set("ext_script_err_msg", err)
102                  end'
103         remove_field => [ "loc" ]
104     }
```

Εικόνα 14. Τρόπος κλήσης του latlongcalc.py

Το latlongcalc.py δέχεται σαν παράμετρο το αλφαριθμητικό που περιέχει την τοποθεσία και αν περιέχει κενά τα παίρνει σαν πολλές παραμέτρους. Στην συνέχεια παίρνει μια μια τις παραμέτρους και τις βάζει σε μια μεταβλητή αν δεν είναι η 1η ή η τελευταία παράμετρος όταν περνάει την παράμετρο στην μεταβλητή προσθέτει επιπλέον « ,+ » κάτι το οποίο είναι απαραίτητο για την σωστή κλήση του Google maps api το οποίο με βάση την τοποθεσία που του περνάμε επιστρέφει ένα αντικείμενο JSON από του οποίου τα πεδία κρατάμε το ['geometry']['location']['lat'] και ['geometry']['location']['lng'] τα οποία τυπώνουμε στο τερματικό και πίσω στο ruby script μας τα διαβάζουμε και τα περνάμε σε νέα πεδία στο tweet μας [geolocation][lat] και [geolocation][lon]. Η ονομασία των εμφωλευμένων πεδίων είναι σημαντική για την ύστερη αναγνώριση των πεδίων ως μεταβλητών γεωγραφικής πληροφορίας για την οπτικοποίηση τους σε χάρτη. Σε περίπτωση σφάλματος σε ένα από τα δυο scripts θα δημιουργηθεί καινούργιο πεδίο «ext_script_err_msg» το οποίο θα περιέχει το μήνυμα σφάλματος.

```
1   import requests
2   import sys
3   import os
4
5   #CURRENT_DIR = os.path.dirname(__file__)
6   #file_path = os.path.join(CURRENT_DIR, 'test.txt')
7   #f = open(file_path,"w")
8   location = ""
9   for i in range(1,len(sys.argv)):
10      # f.write(sys.argv[i])
11      if(i!=1 and i!= len(sys.argv)):
12          location = location +","+ sys.argv[i]
13      else:
14          location = location + sys.argv[i]
15  #f.write(location)
16  #f.close()
17  address = 'https://maps.googleapis.com/maps/api/geocode/json?address='+location+'&key=
18  response = requests.get(address)
19  resp_json_payload = response.json()
20
21  print(resp_json_payload['results'][0]['geometry']['location']['lat'])
22  print(resp_json_payload['results'][0]['geometry']['location']['lng'])
```

Εικόνα 15. latlongcalc.py – Python script για τον υπολογισμό γεωγραφικών συντεταγμένων.

4.3 Διαμόρφωση του Elasticsearch

Μέχρι αυτό το στάδιο πέρα από την εγκατάσταση του elasticsearch και την δημιουργία των index όπου θα αποθηκεύονται τα δεδομένα το μόνο που χρειάστηκε για το elasticsearch ήταν το mapping των πεδίων που θα αποθηκεύονταν σε αυτό. Η διαδικασία αυτή κάνει γνωστή στο πρόγραμμα το είδος των πεδίων. Ειδικά άμα θέλουμε να τα χρησιμοποιήσουμε αργότερα σε κάποια από τις οπτικοποιήσεις στο kibana όπως για παράδειγμα ο ορισμός των πεδίων ως keywords ή geo_point. Για το mapping απαιτείται η αποστολή ενός json αντικειμένου με όλες τις απαραίτητες πληροφορίες. Γενικά για όλες τις ενέργειες GET, POST, PUT, DELETE που είναι απαραίτητες στο elasticsearch τις πραγματοποιώ με το Postman.

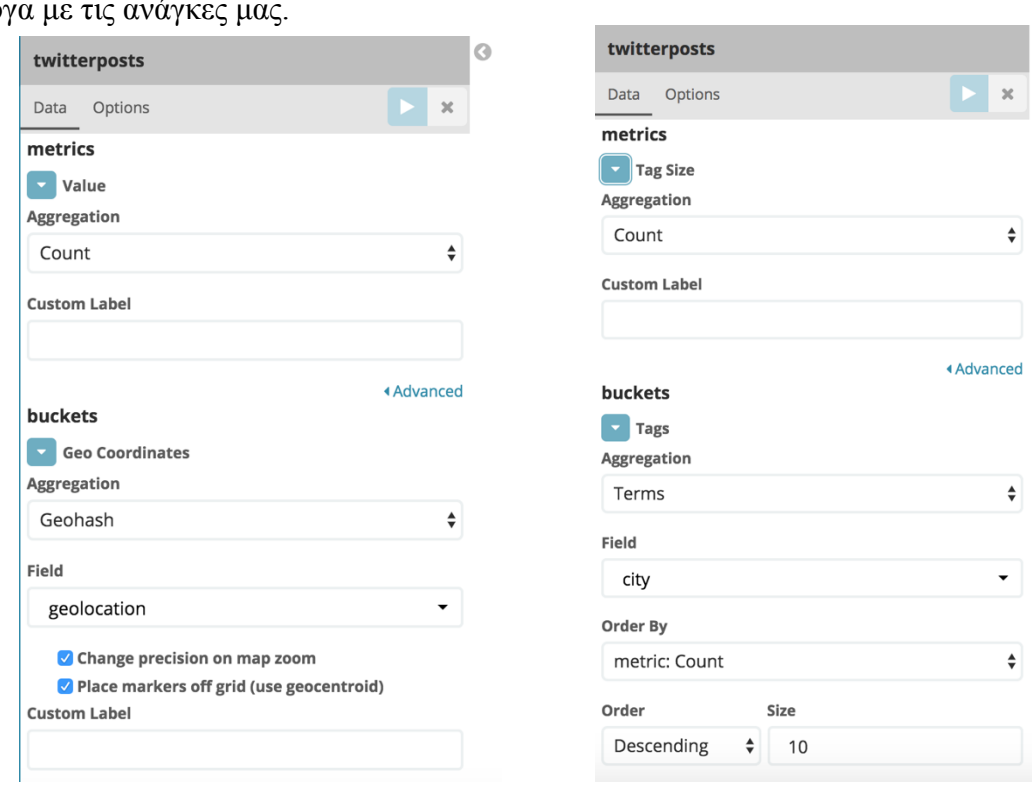
```
{
  "mappings": {
    "logs": {
      "properties": {
        "country": {
          "type": "keyword"
        },
        "city": {
          "type": "keyword"
        },
        "friends_count": {
          "type": "long"
        },
        "timezone": {
          "type": "keyword"
        },
        "retweet_count": {
          "type": "long"
        },
        "screen_name": {
          "type": "keyword"
        },
        "text": {
          "type": "text"
        },
        "lang": {
          "type": "keyword"
        },
        "total_posts": {
          "type": "long"
        },
        "followers": {
          "type": "long"
        },
        "location": {
          "type": "keyword"
        },
        "user_description": {
          "type": "text"
        },
        "sentiment": {
          "type": "keyword"
        },
        "geolocation": {
          "type": "geo_point"
        }
      }
    }
  }
}
```

Εικόνα 16. JSON αντικείμενο για το mapping των πεδίων των δεδομένων στο elasticsearch

4.4 Διαμόρφωση του Kibana

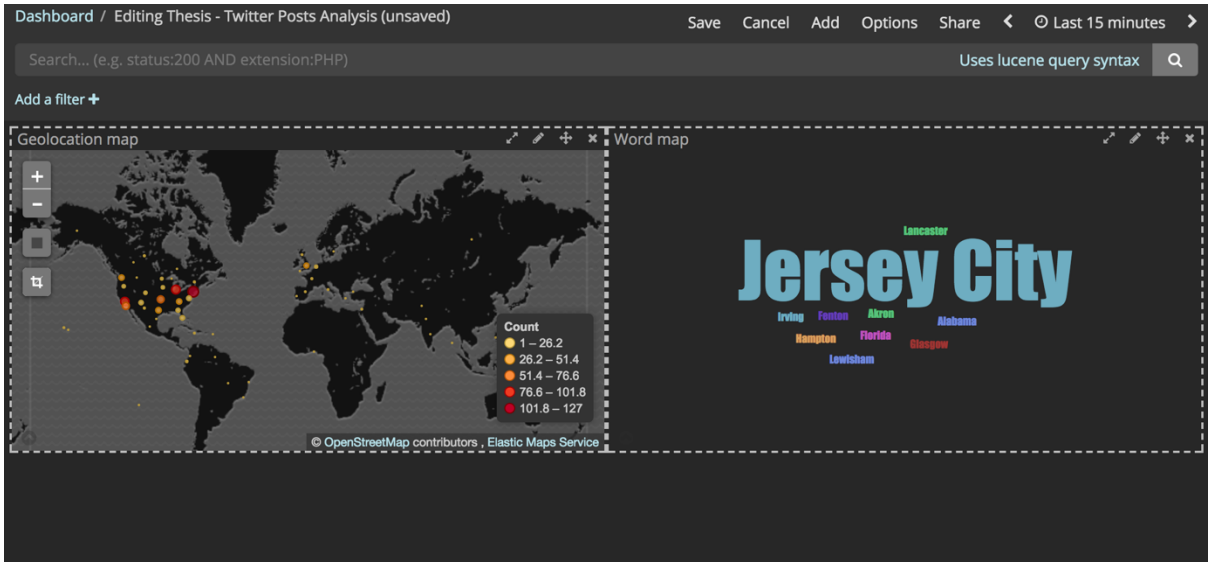
Στην συνέχεια εγκατέστησα το kibana (έκδοση 5.6.3) και θεώρησα πως έστω και με λίγα δεδομένα μπορώ να δημιουργήσω ήδη μερικές απλές οπτικοποιήσεις για να δω την γενική λειτουργία του εργαλείου και να έχω κάποια εμφανή αποτελέσματα. Για την λειτουργία του

το kibana προϋποθέτει να έχουμε το elasticsearch να τρέχει. Όταν ξεκινάει το kibana για πρώτη φορά πρέπει να προσδιορίσουμε το όνομα του ευρετηρίου στο elasticsearch στο οποίο είναι αποθηκευμένα τα δεδομένα. Όταν ξεκίνησα το kibana στο τερματικό υπήρχαν προειδοποιήσεις πως οι εκδόσεις του elasticsearch και του kibana είναι διαφορετικές. Ωστόσο το kibana ξεκίνησε και αφού προσδιόρισα το όνομα του ευρετηρίου, στην κύρια καρτέλα (Discover) ήταν διαθέσιμα τα δεδομένα από το elasticsearch οπότε θεώρησα πως δεν υπάρχει κανένα πρόβλημα. Στην συνέχεια στην καρτέλα Visualize όπου γίνεται η δημιουργία των αντικειμένων οπτικοποίηση έφτιαξα ένα tag cloud και ένα coordinate map αλλά στις ρυθμίσεις του coordinate map έδειχνε σφάλμα πως δεν υπάρχουν δεδομένα τύπου geo_point για την δημιουργία του χάρτη ενώ ξεκάθαρα στο προηγούμενο βήμα της χαρτογράφησης στο elasticsearch προσδιόρισα πεδίο αυτού του τύπου. Το πρόβλημα ήταν στην διαφορά των εκδόσεων των δυο εργαλείων. Αναγκάστηκα να κάνω αναβάθμιση του elasticsearch σε 5.6.3 και τότε μπορούσα να δω τον χάρτη να λειτουργεί με κανονικά αποτελέσματα. Για την δημιουργία των οπτικοποιήσεων χρειάζεται να διαμορφώσουμε συγκεκριμένες ρυθμίσεις ανάλογα με τις ανάγκες μας.



Εικόνα 17. Ρυθμίσεις των οπτικοποιήσεων tag cloud (πρώτη) και coordinate map (δεύτερη)

Στην συνέχεια στην καρτέλα Dashboard δημιούργησα ένα νέο ταμπλό που αποτελείται από οπτικοποιήσεις τις επιλογής μας τα οποία έχουν δημιουργηθεί στο Visualize. Του έδωσα την ονομασία Thesis-Twitter Posts Analysis και στην συνέχεια πρόσθεσα τις δυο οπτικοποιήσεις που δημιούργησα προηγουμένως.



Εικόνα 18. Παράδειγμα του ταμπλό με τις δυο οπτικοποιήσεις.

4.5 Εκπαίδευση και έλεγχος των κατηγοριοποιητών

Το επόμενο βήμα ήταν να αρχίσω το τμήμα της συναισθηματικής ανάλυσης. Για την επεξεργασία και την συναισθηματική ανάλυση χρησιμοποίησα την βιβλιοθήκη nltk (natural language toolkit) όπως και την scikit-learn και για την χρήση των classifiers που παρέχουν αυτές οι δυο βιβλιοθήκες χρειάστηκε να κάνω εγκατάσταση και άλλες προαπαιτούμενες όπως numpy, scypi κα και από το nltk κατέβασα μερικά από τα σετ δεδομένων που παρέχει όπως movie_review, twitter_posts καθώς και stopwords, wordnet τα οποία είναι απαραίτητα για την αφαίρεση των stopwords και την χρήση συνωνύμων αντίστοιχα. Η γενική ιδέα είναι να έχουμε ένα python script όπου κάνουμε εκπαίδευση τους κατηγοριοποιητή μας με μεγάλα σετ δεδομένων. Αυτό ωστόσο απαιτεί χρόνο κάτι το οποίο θέλουμε να αποφύγουμε να γίνεται κάθε φορά κατά την κλήση του script (analysis.py) το οποίο θα κάνει κατηγοριοποίηση τα κείμενα των posts κατά την λήψη τους. Επομένως η διαδικασία της εκπαίδευσης διαχωρίστηκε σε ένα ξεχωριστό script (classifier.py). Η κλήση του analysis.py γίνεται κατά το στάδιο φιλτραρίσματος του logstash ακριβώς με τον ίδιο τρόπο με το latlongcalc.py μετά από αυτό. Καλούμε το script με παράμετρο το κείμενο του post και διαβάζουμε ότι επιστρέφει ανάλογα pos ή neg.

```
107     ruby {
108         code => 'require "open3"
109                  text = event.get("text")
110                  cmd =  "python /Users/kar_lef/Desktop/Thesis/scripts/analysis.py #{text} "
111                  stdin, stdout, stderr = Open3.popen3(cmd)
112                  event.set("sentiment", stdout.read.delete!("\n"))
113                  err = stderr.read
114                  if err.to_s.empty?
115                    filter_matched(event)
116                  else
117                    event.set("ext_script_err_msg", err)
118                  end'
119     }
120 }
```

Εικόνα 19. Κλήση του analysis.py στο στάδιο φιλτραρίσματος του logstash πριν αποθηκευτεί το post στο elasticsearch.

Για το στάδιο της εκπαίδευσης δημιούργησα διαφορετικά scripts ανάλογα με το είδος των σετ δεδομένων (nltk data, scv ,plain txt) που έπρεπε να φορτώσω αλλά όλα ακολουθούν το ίδιο σκεπτικό. Τα δεδομένα μας είναι χωρισμένα σε θετικά και αρνητικά κείμενα και στην

περίπτωση που δεν αποτελούν ζευγάρι αυτά τα δυο δεδομένα δηλαδή κείμενο και χαρακτηρισμός πρέπει κατά την φόρτωση τους να προσθέσουμε τον χαρακτηρισμό για να μπορεί να γίνει η διαδικασία της εκπαίδευσης. Αφού φορτώσουμε τα δεδομένα στο πρόγραμμα στην συνέχεια τα περνάμε από μια προ επεξεργασία κειμένου. Όπως για παράδειγμα tokenization, αφαιρέσει stopwords, stemming και lemmatization. Ως αποτέλεσμα πρέπει να έχουμε κάθε κείμενο ως μια πλειάδα που αποτελείται από το κείμενο ως λίστα λέξεων και το χαρακτηριστικό (pos ή neg).

```
41  def docs_preprocessing(docname,sentiment,text):
42      docs = []
43      for line in open(docname, 'r'):
44          json_data = json.loads(line)['text'] #reading text from tweet
45          tokens = tokenizer.tokenize(json_data) #split text in tokens
46          filtered_tokens = [word for word in tokens if word not in stopwords.words('english')]#removing stopwords
47          stemed_tokens = [ps.stem(word) for word in filtered_tokens]#stemming
48          lemmatized_tokens = [lemmatizer.lemmatize(word) for word in filtered_tokens]#lemmatization
49          text.extend(lemmatized_tokens)#storing all the words for later feature extraction
50          docs.append((list(filtered_tokens),sentiment))
51      return docs
```

Εικόνα 20. Συνάρτηση επεξεργασίας των δεδομένων.

Καθ' όλη την επεξεργασία των δεδομένων πρέπει να αποθηκεύσουμε κάθε λέξη ώστε στο επόμενο βήμα να μπορέσουμε να βγάλουμε μια λίστα συχνοτήτων από την οποία παρακάτω θα πάρουμε τους όρους που θα λάβουν μέρος στην εκπαίδευση. Αυτό πάλι μπορεί να αλλάξει ανάλογα με τα δεδομένα. Για παράδειγμα σε σετ δεδομένων με κανονικό πλήρες κείμενο μπορούμε να πάρουμε τμήμα των πιο συχνά εμφανιζόμενων λέξεων αλλά και τμήμα των σπάνιων καθώς μπορούν να έχουν περισσότερη αξία. Στην περίπτωση των twitter posts ως σετ δεδομένων οι πιο σπάνιες λέξεις συνήθως είναι περίεργα ονόματα χρηστών και γενικά «λέξεις» χωρίς νόημα οπότε σε αυτή την περίπτωση πρέπει να αποφύγουμε τις πιο σπάνιες. Αφού έχουμε την λίστα συχνοτήτων μπορούμε να βγάλουμε τα features για κάθε κείμενο που θα χρησιμοποιηθεί, η λογική σε αυτή τη μέθοδο είναι απλή. Ελέγχουμε από την λίστα συχνοτήτων ποιες λέξεις υπάρχουν στο κείμενο μας και φτιάχνουμε dictionary με τους όρους και τις τιμές true/false ανάλογα με την ύπαρξη ή όχι του κάθε όρου στο κείμενο.

```
34  def document_features(document):
35      document_words = set(document)
36      features = {}
37      for word in word_features:
38          features[word] = (word in document_words)
39      return features
40
```

Εικόνα 21. Επιλογή των features για την εκπαίδευση.

Τώρα το σύνολο των πλειάδων των κείμενων πρέπει να ανακατευθεί ώστε να μην έχουμε όλα τα θετικά και στην συνέχεια όλα τα αρνητικά κείμενα και να μοιράσουμε το σύνολο σε σετ εκπαίδευσης και ελέγχου. Με το πρώτο θα κάνουμε την εκπαίδευση και με το δεύτερο θα ελέγξουμε την ακρίβεια.

```
85  all_words = nltk.FreqDist(w.lower() for w in text) #find the frequencies
86  word_features = all_words.keys()[:4000] + all_words.keys()[-4000:] #from
87
88  posfeaturesets = [(document_features(d), c) for (d, c) in pos_docs] #ext
89  negfeaturesets = [(document_features(d), c) for (d, c) in neg_docs] #ext
90
91  train_set = posfeaturesets[:4800] + negfeaturesets[:4800] #create traini
92  test_set = posfeaturesets[4800:] + negfeaturesets[4800:] #create testing
93
94  shuffle(train_set)
95  shuffle(test_set)
```

Εικόνα 22. Δημιουργία των λέξεων χαρακτηριστικών και των σετ εκπαιδεύσεων και ελέγχου.

Τέλος μπορούμε να δείξουμε τους ν όρους που έπαιξαν τον πιο σημαντικό ρόλο κατά την εκπαίδευση και αν είμαστε ικανοποιημένοι με την ακρίβεια του κατηγοριοποιητή μας μπορούμε να προχωρήσουμε στην αποθήκευση του. Η αποθήκευση των κατηγοριοποιητών γίνεται με χρήση του pickle με το οποίο γράφουμε αντικείμενα ως bytes και στην συνέχεια μπορούμε να τα ανακτήσουμε από άλλο script.

```
101    classifier = nltk.NaiveBayesClassifier.train(train_set) #tra
102    accuracy = nltk.classify.accuracy(classifier, test_set) #pri
103    print "The accuracy for this round is: " , accuracy
104    classifier.show_most_informative_features(10) #show top 10 m
105    if(accuracy > 0.7):
106        print "Saving classifier..."
107        save_classifier = open("naivebayes.pickle","wb")
108        pickle.dump(classifier,save_classifier)
```

Εικόνα 23. Εκπαίδευση του κατηγοριοποιητή και αποθήκευση του.

4.6 Analysis.py script κατά την συλλογή των δεδομένων

Το analysis.py script το οποίο χρησιμοποιείται για την κατάταξη των κειμένων ως θετικά ή αρνητικά είναι πιο απλό. Καθώς η διεργασία της εκπαίδευσης έχει γίνει προηγουμένως στην συγκεκριμένη φάση χρειάζεται μόνο να επιλέξουμε τα features για το συγκεκριμένο κείμενο που αναλύουμε, να φορτώσουμε στο script τους ήδη αποθηκευμένους classifiers από τα pickle αρχεία και να κάνουμε την ταξινόμηση με την βοήθεια αυτών.

```
1   import sys
2   import pickle
3   import os
4   from nltk.corpus import wordnet
5   from nltk.classify import ClassifierI
6   from statistics import mode
7
8
9   def extract_features(document): #creates features based on the extracted features from the training requires a file with the words
10      document_words = set(document)
11      features = {}
12      for word in word_features:
13          features[word] = (word in document_words)
14      # print features
15      return features
16
17  def bag_of_words(words): #creates features from all the words of our sentence
18      features = {}
19      for word in words:
20          features[word] = True
21          syns = wordnet.synsets(word)
22          for w in syns:
23              features[w.lemmas()[0].name()] = True
24      return features
25      # return dict([word, True] for word in words)
26
27  CURRENT_DIR = os.path.dirname(__file__)
28
29  sentiment = ""
30  text = []
31  word_features = []
32  for i in range(1,len(sys.argv)):
33      text.append(sys.argv[i])
34  bag_of_words(text)
35  file_path = os.path.join(CURRENT_DIR, 'features_set.txt')
36  file = open(file_path,"r")
37  for line in file:
38      word =  line.rstrip('\n')
39      word_features.append(word)
40  # print "Features were read from file."
41
42  file_path = os.path.join(CURRENT_DIR, 'naivebayes.pickle')
43  classifier_f = open(file_path,"rb")
44  naivebayes_classifier = pickle.load(classifier_f)
45
46  sentiment = naivebayes_classifier.classify(bag_of_words(text))
47  print sentiment
```

Εικόνα 24. analysis.py (v.0.1)

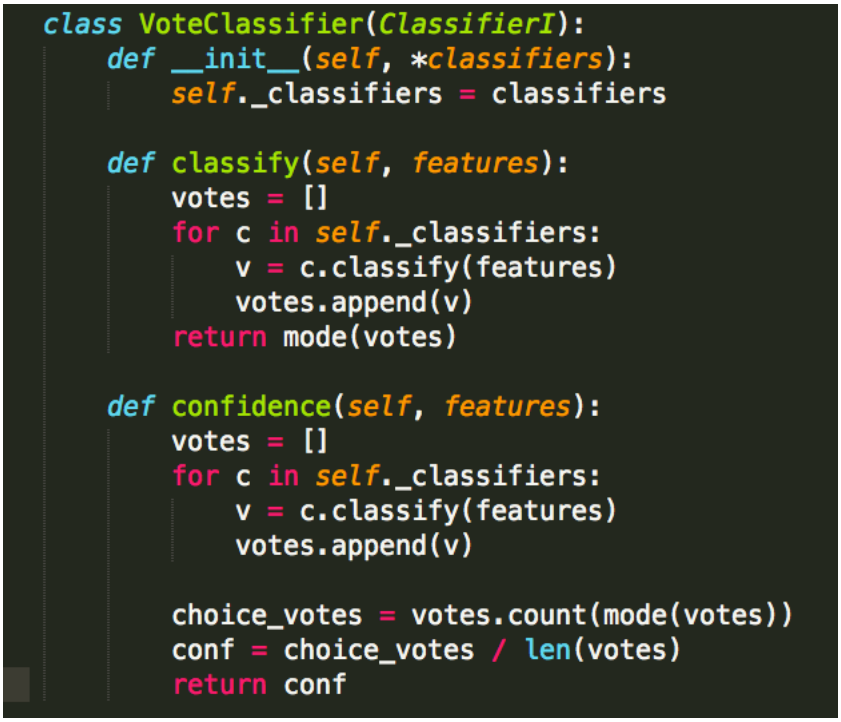
Εδώ χρησιμοποίησα δυο διαφορετικές συναρτήσεις για την επιλογή των features γιατί με την πρώτη η οποία κάνει χρήση της λίστας των λέξεων από την εκπαίδευση δεν έδινε καλά αποτελέσματα. Έτσι δοκίμασα τον δεύτερο τρόπο που δημιουργεί μια λίστα με όλες τις λέξεις και ακόμα με τη συνάρτηση synsets και το data set wordnet παρέχουν συνώνυμα τα οποία προσθέτω και αυτά στην λίστα για αντιστοίχιση με ίσως περισσότερες παρόμοιες λέξεις κατά την ταξινόμηση του κειμένου.

4.7 Χρήση και συμπερίληψη διαφορετικών κατηγοριοποιητών σε έναν ομαδοποιημένο κατηγοριοποιητή

Τώρα που έχουμε περιγράψει την γενική ροή επεξεργασίας των δεδομένων μπορούμε να δοκιμάσουμε διαφορετικούς κατηγοριοποιητές και στην συνέχεια να συγκρίνουμε τα αποτελέσματα που δίνει ο καθένας. Όπως γνωρίζουμε τα αποτελέσματα κάθε κατηγοριοποιητή εξαρτώνται από τα δεδομένα (data set) που του παρέχουμε όπως για παράδειγμα μορφή των δεδομένων, μέγεθος του σετ εκπαίδευσης και ελέγχου όπως επίσης και τι προ επεξεργασία θα υποστούν τα δεδομένα όπως για παράδειγμα stemming, lemmatization αφαίρεση stop words κτλπ. Ακόμα τα χαρακτηριστικά που εξάγονται και θα χρησιμοποιηθούν παίζουν πολύ σημαντικό ρόλο στην εκπαίδευση καθώς και συγκεκριμένες ρυθμίσεις που υποστηρίζει ο κάθε κατηγοριοποιητή. Συγκεκριμένα κατά την εκπαίδευση χρησιμοποιήθηκαν οι παρακάτω κατηγοριοποιητές :

- NaiveBayes
- MultinomialNB
- BernoulliNB
- LogisticRegression
- SGDClassifier
- LinearSVC

Μια γνωστή τεχνική που ακολουθούν πολλοί σε αυτή την περίπτωση είναι η ομαδοποίηση των κατηγοριοποιητών. Με την ομαδοποίηση εννοούμε τον συνυπολογισμό των αποτελεσμάτων που αυτή παρέχουν. Φτιάχνουμε έναν «δικό μας» κατηγοριοποιητή ο οποίος περιλαμβάνει όλους τους κατηγοριοποιητές που χρησιμοποιούμε και έχουμε εκπαιδεύσει. Αφού ο καθένας δώσει την απάντηση του ως προς το sentiment του παρεχόμενου κειμένου ο κατηγοριοποιητής μας επιστρέφει την τελική απάντηση που είναι η απάντηση με τις περισσότερες εμφανίσεις.
Στην κλάση αυτή έχουμε προσθέσει δυο μεθόδους. Η μέθοδος classify, για κάθε κατηγοριοποιητή που περιέχεται στο αντικείμενο μας, μέσα σε έναν βρόχο κάνει για τον καθένα κατηγοριοποίηση του κειμένου και τέλος με την συνάρτηση mode ανακτά την απάντηση με τις περισσότερες εμφανίσεις. Η μέθοδος confidence δίνει ένα ποσοστό που προσδιορίζει το πλήθος των κατηγοριοποιητών που συνέβαλαν στο τελικό αποτέλεσμα. Όσο πιο κοντά ήμαστε σε μια ομόφωνη απάντηση τόσο μεγαλύτερο το ποσοστό βεβαιότητας.



```
class VoteClassifier(ClassifierI):
    def __init__(self, *classifiers):
        self._classifiers = classifiers

    def classify(self, features):
        votes = []
        for c in self._classifiers:
            v = c.classify(features)
            votes.append(v)
        return mode(votes)

    def confidence(self, features):
        votes = []
        for c in self._classifiers:
            v = c.classify(features)
            votes.append(v)

        choice_votes = votes.count(mode(votes))
        conf = choice_votes / len(votes)
        return conf
```

Εικόνα 25. Ορισμός του ομαδοποιημένου κατηγοριοποιητή

5.Αποτελέσματα και συμπεράσματα

Σε αυτή την ενότητα της πτυχιακής θα αναφερθώ αρχικά στα πλεονεκτήματα του elasticsearch ως τεχνολογία καθώς και προβλήματα του σε πραγματικά περιβάλλοντα και στην συνέχεια θα συνοψίσω τα αποτελέσματα της συναισθηματικής ανάλυσης με βάση τις διάφορες προ επεξεργασίες κειμένου και αλγορίθμους που χρησιμοποιήθηκαν.

5.1 Σύνοψη του elasticsearch ως τεχνολογία για επεξεργασία big data

Το elasticsearch είναι ένα εξαιρετικό σύστημα για ευρετηριοποίηση αρχείων και παρέχει μια ισχυρή αναζήτηση σε πλήρες κείμενο. Καθώς επίσης η απλή και ταυτόχρονα ισχυρή DSL (Domain Specific Language) γλώσσα ερωτήσεων που βασίζεται σε μορφή JSON το καθιστούν μια από τις καλύτερες επιλογές για μηχανή αναζήτησης σε εφαρμογές ιστού. Ωστόσο παρόλα τα θετικά του δεν είναι σε θέση να αντικαταστήσει τελείως τις ήδη υπάρχουσες τεχνολογίες για αποθήκευση και ανάλυση όπως σχεσιακές βάσεις δεδομένων και συστήματα όπως το Hadoop που είναι σχεδιασμένα για τις συγκεκριμένες διεργασίες.

Το elasticsearch είναι ισχυρό και αποτελεσματικό για διεργασίες αναζήτησης για τους παρακάτω λόγους:

- Είναι αναπτυγμένο πάνω στην τεχνολογία του Lucene
- Παρέχει αναζήτηση σε πλήρες κείμενο καθώς και αυτόματη συμπλήρωση
- Τα αποτελέσματα είναι ανεκτικά σε γραπτά λάθη
- Είναι ελεύθερο από κάποιο συγκεκριμένο «σχήμα» (Schema free)
- Παρέχει ένα RESTful API για την εκτέλεση διεργασιών
- Ταχύτητα και επεκτασιμότητα

Ωστόσο υπάρχουν προβλήματα τα οποία το elasticsearch δεν μπορεί να αντιμετωπίσει αποτελεσματικά και ο συνδυασμός τεχνολογιών είναι απαραίτητος. Πάντα στις επιλογές μας παίζει ρόλο τι πραγματικά θέλουμε να πετύχουμε. Για παράδειγμα μερικά από τα προβλήματα είναι:

- Σφάλματα κατά την εισαγωγή δεδομένων από ροή
- Προβλήματα ρυθμίσεων κατά την κλιμάκωση του συστήματος σε πραγματικό περιβάλλον παραγωγής
- Η έννοια ελεύθερου σχήματος (schema free) δεν είναι ακριβώς ίδια με του Hadoop ή NoSQL
- Μεταφορτώσεις μεγάλου όγκου
- Έλλειψη ισχυρών διεργασιών ανάλυσης

Και εδώ έρχονται τεχνολογίες όπως το Hadoop με το HDFS που καθιστά το σύστημα εξαιρετικά ανεκτικό σε σφάλματα και αποτρέπει την απώλεια δεδομένων. Καθώς επίσης μαζική μεταφόρτωση και εισαγωγή δεδομένων από ροή υποστηρίζονται με την βοήθεια συμπληρωματικών εργαλείων. Και εργαλεία SQL που υποστηρίζουν την πλήρη ισχύ αναζήτησης που μια σχεσιακή βάση δεδομένων παρέχει. Ωστόσο για να στηθεί ένα τέτοιο σύστημα καθώς και η συντήρηση απαιτεί μεγάλο κόστος. Επίσης απαιτεί άτομα με εξειδικευμένες γνώσεις στον συγκεκριμένο τομέα για να έχουμε αποδοτικά αποτελέσματα και την σωστή τους χρήση.

5.2 Σύγκριση των αποτελεσμάτων ακρίβειας του sentiment analysis

Στους πίνακες που ακολουθούν παρακάτω έχουν καταγραφεί ποσοστά ακρίβειας κατά την διαδικασία εκπαίδευσης και ελέγχου. Τα αποτελέσματα διαχωρίστηκαν σε πρώτο επίπεδο με βάση το σετ δεδομένων πάνω στα οποία έγινε η εκπαίδευση. Τα επόμενα επίπεδα

διαχωρισμού βασίζονται στο αν έχουν γίνει τροποποιήσεις των δεδομένων και τι είδους. Τα 3 βασικά που λήφθηκαν υπόψιν είναι αν υπήρξαν τεχνικές NLP, αν κατά την εκπαίδευση εκτός από τους πιο συχνά εμφανιζόμενους όρους χρησιμοποιούμε και τους πιο σπάνιους και τέλος αν χρησιμοποιούμε συγκεκριμένα μέρη του λόγου όπως ρήματα, επιρρήματα και επίθετα για παράδειγμα. Καθώς επίσης καταγράφηκαν και ποσοστά για τον συνδυασμό των προαναφερθέντων.

Από τα παρακάτω ποσοστά είναι εμφανές για το $1^{ο}$ σετ δεδομένων πως στην περίπτωση όπου έχουμε επεξεργασία των δεδομένων με μεθοδολογίες NLP όλες οι επιπλέον μεθοδολογίες οδηγούν σε αύξηση των ποσοστών ακριβείας καθώς και ο συνδυασμός τους. Ωστόσο στην περίπτωση που δεν εφαρμόζουμε τεχνικές NLP παρατηρούμε πως οι κατηγοριοποιητές δίνουν καλύτερα αποτελέσματα απ' ότι με την εφαρμογή τους. Επίσης η περίπτωση συνυπολογισμού των πιο σπάνιων όρων δίνει τα μεγαλύτερα ποσοστά από όλα τα αποτελέσματα γενικά. Κάτι περίεργο που παρατηρείται σε αυτό το σετ δεδομένων και στην συγκεκριμένη περίπτωση (χωρίς μεθοδολογίες NLP) είναι πως ο συνδυασμός των δυο μεθοδολογιών (πιο σπάνιοι όροι και μέρη του λόγου) δίνει χαμηλότερα αποτελέσματα ενώ το αναμενόμενο αποτέλεσμα θα ήταν να πάρουμε ακόμα υψηλότερα ποσοστά ακριβείας. Αυτό θα μπορούσε ωστόσο να αιτιολογηθεί από την μείωση των ολικών λέξεων που λαμβάνουν μέρος στην εκπαίδευση. Τώρα στο $2^{ο}$ σετ δεδομένων στην $1^{η}$ περίπτωση (χρήση NLP μεθοδολογιών) έχουμε ακριβώς την ίδια συμπεριφορά με το $1^{ο}$ σετ. Σταδιακή αύξηση των ποσοστών με κάθε μεθοδολογία και των τελικό συνδυασμό. Στην περίπτωση όμως που δεν κάνουμε χρήση NLP μεθοδολογιών έχουμε την ακριβώς αντίθετη συμπεριφορά σε σχέση με το $1^{ο}$. Η μέθοδος που χρησιμοποιεί τα μέρη του λόγου δίνει καλύτερα αποτελέσματα από αυτήν των σπάνιων όρων και ο συνδυασμός των δυο μεθοδολογιών δίνει τα καλύτερα αποτελέσματα στην κατηγορία αυτή. Παρ όλα αυτά και στα δυο σετ δεδομένων βλέπουμε καλύτερα αποτελέσματα στις περιπτώσεις όπου δεν κάνουμε χρήση μεθοδολογιών NLP. Αυτά τα αποτελέσματα πιθανών να οφείλονται στο ότι απλές τεχνικές NLP όπως stemming και lemmatization δεν είναι αρκετές και ίσα ίσα κάνουν λιγότερο πλούσια σε πληροφορία τα δεδομένα μας. Ωστόσο ακόμα και μετά την αφαίρεση των προαναφερθέντων μεθοδολογιών τα αποτελέσματα δεν αυξάνονται δραστικά ακόμα και με τον συνδυασμό των άλλων μεθόδων (69,75%). Επομένως, πιθανόν να απαιτείται πιο εξειδικευμένη επεξεργασία όπως για παράδειγμα χρήση regular expressions για εύρεση οντοτήτων όπως όνομα, γνωστά πρόσωπα κλπ. (named entity) και να έχουν δίπλα τους κάποιου είδος επίθετο ή επίρρημα.

Positive/Negative.txt						
Classifiers	NaiveBayes	MultinomialNB	BernoulliNB	Logistic Regression	Stochastic Gradient Descent	Linear Support Vector
With NLP	63,65%	64,21%	64,03%	64,05%	63,27%	63,46
Frequent + rare features	65,53%	65,53%	65,63%	65,16%	64,78%	64,5%
Parts of speech	68,64%	68,54%	68,73%	67,89%	67,23%	68,83%
Frequent + rare features & Parts of speech	69,58%	69,77%	69,39%	68,64%	69,02%	67,89%

Without NLP	64,9%	66,11%	63,85%	66,26%	65,90%	65,96%
Frequent + rare features	79,06%	79,96%	79,81%	78,46%	73,34	77,4%
Parts of speech	72,43%	72,13%	73,64%	71,98%	67,16%	69,42%
Frequent + rare features & Parts of speech	73,79%	74,69%	74,24%	75,9%	71,98%	71,83

Positive_tweets/Negative_tweets.json						
Classifiers	NaiveBayes	MultinomialNB	BernoulliNB	Logistic Regression	Stochastic Gradient Descent	Linear Support Vector
With NLP	62,5%	62,74%	62,25%	61,25%	61,5%	61%
Frequent + rare features	65,25%	63,75%	65%	65,75%	63,74%	64%
Parts of speech	70,5%	70%	71%	68,75%	68,5%	69%
Frequent + rare features & Parts of speech	70,75%	70,75%	70,5%	69,25%	70%	69%
Without NLP	64%	64,75%	64,5%	61,25%	61,5%	60,75%
Frequent + rare features	66,25%	66,25%	66,5%	65,25%	65,75%	64%
Parts of speech	71,5%	71,5%	70,75%	71%	69,75%	71,25%
Frequent + rare features & Parts of speech	72,75%	72,5%	71,75%	72,25%	72,3%	72,15%

6.Βιβλιογραφικές Αναφορές

[1] Wikipedia (2008), Data, Διαθέσιμο στο: https://en.wikipedia.org/wiki/Data [Τελευταία επίσκεψη 19/12/2017]
[2] Twitter (2017), Docs, Διαθέσιμο στο: https://developer.twitter.com/en/docs [Τελευταία επίσκεψη 19/12/2017]
[3] Elastic (2017), Elasticsearch References, Διαθέσιμο στο: https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html [Τελευταία επίσκεψη 19/12/2017]
[4] Elastic (2017), Kibana User Gide, Διαθέσιμο στο: https://www.elastic.co/guide/en/kibana/current/index.html [Τελευταία επίσκεψη 19/12/2017]
[5] Elastic (2017), Logstash User Gide, Διαθέσιμο στο: https://www.elastic.co/guide/en/logstash/current/index.html [Τελευταία επίσκεψη 19/12/2017
[6] Wikipedia (2017), Elasticsearch, Διαθέσιμο στο: https://en.wikipedia.org/wiki/Elasticsearch [Τελευταία επίσκεψη 19/12/2017]
[7] Opensource (2016), Gathering insights from data: an overview of the Elastick stack, Διαθέσιμο στο: https://opensource.com/life/16/6/overview-elastic-stack [Τελευταία επίσκεψη 19/12/2017]
[8] Vrepin (2016), How to incorporate external utility scripts into Logstash pipeline, Διαθέσιμο στο: http://vrepin.org/vr/Logstash-filter-external-script/ [Τελευταία επίσκεψη 19/12/2017]
[9] Textminingonline (2014), Dive into NLTK, Part II: Sentence Tokenize and Word Tokenize, Διαθέσιμο στο: http://textminingonline.com/dive-into-nltk-part-ii-sentence-tokenize-and-word-tokenize [Τελευταία επίσκεψη 22/12/2017]
[10] Youtube (2015), NLTK with Python 3 for Natural Language Processing, Διαθέσιμο στο: https://www.youtube.com/playlist?list=PLQVvvaa0QuDf2JswnfiGkliBInZnIC4HL [Τελευταία επίσκεψη 22/12/2017]
[11] Laurentluce (2012), Twitter sentiment analysis using Python and NLTK, Διαθέσιμο στο: http://www.laurentluce.com/posts/twitter-sentiment-analysis-using-python-and-nltk/ [Τελευταία επίσκεψη 22/12/2017]
[12] Import.io, What is data and why is it important, Διαθέσιμο στο: https://www.import.io/post/what-is-data-and-why-is-it-important/ [Τελευταία επίσκεψη 22/12/2017]
[13] Telegraph (2010), 10 ways data is changing how we live, Διαθέσιμο στο: http://www.telegraph.co.uk/technology/7963311/10-ways-data-is-changing-how-we-live.html [Τελευταία επίσκεψη 22/12/2017]
[14] TechTarget (2017), Data, Διαθέσιμο στο: http://searchdatamanagement.techtarget.com/definition/data [Τελευταία επίσκεψη 22/12/2017]
[15] Github (2015), Hello world example of Python script, Διαθέσιμο στο: https://github.com/logstash-plugins/logstash-filter-zeromq/issues/2 [Τελευταία επίσκεψη 22/12/2017]
[16] Elastic (discuss) (2016), How to call python script from Logstash to process (partialy) logs, Διαθέσιμο στο: https://discuss.elastic.co/t/how-to-call-python-script-from-logstash-to-process-partialy-logs/40009 [Τελευταία επίσκεψη 22/12/2017]
[17] Elastic (2014), Elasticsearch from the Top Down, Διαθέσιμο στο: https://www.elastic.co/blog/found-elasticsearch-top-down [Τελευταία επίσκεψη 22/12/2017]
[18] Adil Moujahid (2014), An Introduction to Text Mining using Twitter Streaming API and Python, Διαθέσιμο στο: http://adilmoujahid.com/posts/2014/07/twitter-analytics/ [Τελευταία επίσκεψη 22/12/2017]

[19] Dataquest (2016), Working with streaming data: Using the Twitter API to capture tweets, Διαθέσιμο στο: https://www.dataquest.io/blog/streaming-data-python/ [Τελευταία επίσκεψη 22/12/2017]
[20] IBM, The four V's of Big Data, Διαθέσιμο στο: http://www.ibmbigdatahub.com/infographic/four-vs-big-data [Τελευταία επίσκεψη 22/12/2017]
[21] Oracle, The Foundation for Data Innovation, Διαθέσιμο στο: https://www.oracle.com/big-data/index.html [Τελευταία επίσκεψη 22/12/2017]
[22] Wikipedia (2017), Big Data, Διαθέσιμο στο: https://en.wikipedia.org/wiki/Big_data [Τελευταία επίσκεψη 22/12/2017]
[23] SAS, What it is and why it matters, Διαθέσιμο στο: https://www.sas.com/en_us/insights/big-data/what-is-big-data.html [Τελευταία επίσκεψη 22/12/2017]
[24] Forbes (2015), Big Data: 20 mind-boggling facts everyone must read, Διαθέσιμο στο: https://www.forbes.com/sites/bernardmarr/2015/09/30/big-data-20-mind-boggling-facts-everyone-must-read [Τελευταία επίσκεψη 22/12/2017]
[25] Datamation (2017), Big Data Trends, Διαθέσιμο στο: https://www.datamation.com/big-data/big-data-trends.html [Τελευταία επίσκεψη 22/12/2017]
[26] Treasure Data Blog (2015), Elasticsearch vs Hadoop for Advanced Analytics, Διαθέσιμο στο: https://blog.treasuredata.com/blog/2015/08/31/hadoop-vs-elasticsearch-for-advanced-analytics/ [Τελευταία επίσκεψη 22/12/2017]
[27] 3 Pillar Global, Advantages of Elasticsearch, Διαθέσιμο στο: https://www.3pillarglobal.com/insights/advantages-of-elastic-search [Τελευταία επίσκεψη 22/12/2017]
[28] Apiumhub (2017), Elastic Search: Advantages, Case studies & Books, Διαθέσιμο στο: https://apiumhub.com/tech-blog-barcelona/elastic-search-advantages-books/ [Τελευταία επίσκεψη 22/12/2017]
[29] Scikit learn (2017), Feature extraction, Διαθέσιμο στο: http://scikit-learn.org/stable/modules/feature_extraction.html [Τελευταία επίσκεψη 22/12/2017]
[30] Marco Bonzanini (2015), Sentiment analysis with Python and scikit-learn, Διαθέσιμο στο: https://marcobonzanini.com/2015/01/19/sentiment-analysis-with-python-and-scikit-learn/ [Τελευταία επίσκεψη 22/12/2017]
[31] Quora (2010), What is the best approach for text categorization, Διαθέσιμο στο: https://www.quora.com/What-is-the-best-approach-for-text-categorization [Τελευταία επίσκεψη 22/12/2017]
[32] Quora (2013), What are the best supervised learning algorithms for sentiment analysis in text? I want to do text analysis to determine whether a news item is positive or negative for a given subject, Διαθέσιμο στο: https://www.quora.com/What-are-the-best-supervised-learning-algorithms-for-sentiment-analysis-in-text-I-want-to-do-text-analysis-to-determine-whether-a-news-item-is-positive-or-negative-for-a-given-subject [Τελευταία επίσκεψη 22/12/2017]
[33] Real Python (2014), Twitter sentiment – Python, Docker, Elasticsearch, Kibana, Διαθέσιμο στο: https://realpython.com/blog/python/twitter-sentiment-python-docker-elasticsearch-kibana/ [Τελευταία επίσκεψη 22/12/2017]
[34] NLTK, Natural Language Processing with Python, Διαθέσιμο στο: http://www.nltk.org/book/ [Τελευταία επίσκεψη 22/12/2017]
[35] Python Programming, Data Analysis Tutorials, Διαθέσιμο στο: https://pythonprogramming.net/data-analysis-tutorials/ [Τελευταία επίσκεψη 22/12/2017]

[36] SteamHacker (2010), Text classification for sentiment analysis – Naive Bayes classifier, Διαθέσιμο στο: https://streamhacker.com/2010/05/10/text-classification-sentiment-analysis-naive-bayes-classifier/ [Τελευταία επίσκεψη 22/12/2017]
[37] NLTK (2017), NLTK 3.2.5 Documentation, Διαθέσιμο στο: http://www.nltk.org/index.html [Τελευταία επίσκεψη 22/12/2017]
[38] Neal Caren, Big Data, Διαθέσιμο στο: http://nealcaren.web.unc.edu/big-data/ [Τελευταία επίσκεψη 22/12/2017]
[39] Business Intelligence (2014), 7 ways Big data affects everyday life, Διαθέσιμο στο: https://businessintelligence.com/bi-insights/7-ways-big-data-affects-everyday-life/ [Τελευταία επίσκεψη 22/12/2017]
[40] Elastic (2013), Elasticsearch from the Bottom Up, Part 1, Διαθέσιμο στο: https://www.elastic.co/blog/found-elasticsearch-from-the-bottom-up [Τελευταία επίσκεψη 22/12/2017]
[41] Emma,H. Xiaohui,L. Yong,S. (2013). The role of text pre-processing in sentiment analysis. London. Information Technology and Quantitative Management, Elsevier B.V.
[42] Walaa,M. Ahmed,H. Hoda,K. (2014). Sentiment Analysis algorithms and applications: A survey. Egypt. Ain Shams Engineering Journal