Επεξεργασία Δεδομένων του Twitter Σχετικά με Παρενέργειες Φαρμάκων στο Κατανεμημένο Σύστημα Storm

Τζίμας Δημήτριος

Πολυτεχνείο Κρήτης

Σχολή Ηλεκτρονικών Μηχανικών & Μηχανικών Υπολογιστών (ΗΜΜΥ)

Μάρτιος 2016, Χανιά

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Δημήτριος Τζίμας

με θέμα

Επεξεργασία Δεδομένων του Twitter Σχετικά με Παρενέργειες Φαρμάκων στο Κατανεμημένο Σύστημα Storm

---

Processing Twitter Data Regarding Drug Side Effects on the Storm Distributed System

Εξεταστική Επιτροπή:

Αντώνιος Δεληγιαννάκης, Αναπληρωτής Καθηγητής (Επιβλέπων)

Μίνως Γαροφαλάκης, Καθηγητής

Πολυχρόνης Κουτσάκης, Αναπληρωτής Καθηγητής

Η διπλωματική εργασία μου είναι αφιερωμένη...

Στην μητέρα μου και τον αδερφό μου

Στους φίλους μου

Σε όλους όσους με στήριξαν

Περίληψη

Τα τελευταία χρόνια τα Social Media έχουν γίνει αναπόσπαστο κομμάτι της καθημερινότητας των ανθρώπων σε παγκόσμιο επίπεδο. Καθημερινά, όλο και περισσότεροι χρήστες αξιοποιούν τις πλατφόρμες αυτές για να εκφράσουν την άποψη τους για οτιδήποτε επιθυμούν. Αυτό έχει ως αποτέλεσμα να έχει δημιουργηθεί ένας τεράστιος όγκος πληροφορίας στο internet όπου είναι διαθέσιμος για επεξεργασία. Από τις διάφορες Πλατφόρμες Κοινωνικής Δικτύωσης που υπάρχουν, το Twitter αποτελεί μία από τις περισσότερο διαδεδομένες. Λόγω του γρήγορου και άμεσου χαρακτήρα της επικοινωνίας που προσφέρει, οι χρήστες του σχολιάζουν μέσω αυτού όλες τις κοινωνικές δραστηριότητες που συμβαίνουν. Μέσω της επεξεργασίας των δεδομένων που παράγονται στο Twitter, μπορούμε να καταγράψουμε τις αντιδράσεις των χρηστών για οτιδήποτε γεγονός συμβαίνει σε παγκόσμια κλίμακα.

Σε αυτήν την Διπλωματική Εργασία υλοποιείται μία εφαρμογή όπου παρακολουθεί και επεξεργάζεται δεδομένα από το Twitter σε πραγματικό χρόνο. Πιο συγκεκριμένα, συλλέγουμε δεδομένα που αφορούνε αντιδράσεις των χρηστών σε σχέση με Παρενέργειες που δημιουργούνται από την χρήση Φαρμάκων. Στη συνέχεια τα δεδομένα αυτά κατηγοριοποιούνται ανάλογα με τη γνώμη που εκφράζουν(θετική, αρνητική ή ουδέτερη). Τέλος, αποθηκεύουμε όλες τις χρήσιμες πληροφορίες σε μία Βάση Δεδομένων ώστε να μπορούμε να τα έχουμε διαθέσιμα για επιπλέον επεξεργασία. Για να μπορέσουμε να κάνουμε την επεξεργασία σε πραγματικό χρόνο και να μπορούμε στο μέλλον να την επεκτείνουμε, χρησιμοποιήσαμε το κατανεμημένο σύστημα Storm. Στην τοπολογία που τρέχει στο Storm, κάναμε χρήση διαφόρων εργαλείων όπως το Twitter API, το Lingpipe καθώς και ένα Interface για να βλέπουμε τα περιεχόμενα στη Βάση Δεδομένων. Από τα δεδομένα που αποθηκεύσαμε προέκυψαν κάποια πρώτα συμπεράσματα τα οποία και παραθέτουμε.

Abstract

Over the past few years Social Media have become a major part of every person's daily routine all around the world. Every day, more and more users are utilizing these platforms to express their opinions on whatever they wish. As a result, a huge amount of data is available on the Internet waiting for potential uses. Out of the many Social Media Platforms that exist, Twitter is one of the most widespread. As it offers a fast and immediate way to communicate, users are commenting on almost every personal and public event. By processing that data, it is possible to document users' reactions on whatever topic we wish.

This thesis' subject is an application that observes and processes data from Twitter in real time. Specifically, we are gathering data on user reactions relating to side effects of drugs. These reactions are afterwards being tagged as positive, negative or neutral. Finally, all useful information extracted are exported to a Database, readily available for further processing. To achieve real time processing, and for future improvements to be possible, we used the distributed system Apache Storm. In the topology of Storm we used various tools, such as Twitter API, Lingpipe as well as an Interface to the Database. We are also presenting some first conclusions extracted from the datasets we tested on.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον κ. Δεληγιαννάκη για την συνεργασία που είχαμε στα πλαίσια της Διπλωματικής Εργασίας μου. Τόσο για την συνεχή επικοινωνία και βοήθεια που μου παρείχε καθ' όλη τη διάρκεια εκπόνησης της εργασίας καθώς επίσης και για την υπομονή και επιμονή που έδειξε στην προσπάθεια μου. Επίσης θα ήθελα να ευχαριστήσω τους συναδέλφους μου από το εργαστήριο του Softnet, ειδικότερα την Μανικάκη Β., τον Παυλάκη Ν., την Αράπη Ξ. και τον Γιατράκο Ν., για τη βοήθεια τους σε κάποιες στιγμές της εργασίας. Τέλος θα ήθελα να ευχαριστήσω και τα υπόλοιπα μέλη της επιτροπής μου, κ. Γαροφαλάκη και κ. Κουτσακη, για τον χρόνο που αφιέρωσαν για αυτήν την Διπλωματική Εργασία.

Περιεχόμενα

1.Εισαγωγή 13
1.1 Γενικές πληροφορίες 13
1.2 Αντικείμενο διπλωματικής 14
1.3 Οργάνωση κειμένου 16
2. Storm 17
2.1 Γενική Αρχιτεκτονική του Storm 17
2.2 Κύριες Έννοιες του Storm 18
2.2.1 Μοντέλο δεδομένων 18
2.2.2 Spout 18
2.2.3 Bolt 19
2.2.4 Τοπολογία Storm 21
2.3.3 Ροές Δεδομένων 21
2.3 Storm UI 23
2.4 Hadoop Distributed File System (HDFS) 24
3. Twitter API 25
3.1 Τύποι του Twitter API 25
3.2 Streaming API 27
3.2.1 Σύνδεση στο Streaming API 28
3.2.2 Streaming API Request Parameters 28
3.3 Twitter API Libraries 29
3.4 Περιορισμοί του Twitter Streaming API 30
4. Ανάλυση Συναισθήματος (Sentiment Analysis) 33
4.1 Τεχνικές Κατηγοριοποίησης Συναισθήματος 34
4.1.1 Machine Learning based techniques 34
4.1.2 Lexicon based techniques 37
4.2 NLP Tools 39
5. Τοπολογία 41
5.1 Γενική Περιγραφή της Τοπολογίας 41
5.2 Αναλυτική περιγραφή των στοιχείων της τοπολογίας 42
5.2.1 Η μέθοδος Main 42
5.2.2 TwitterSpout 44
5.2.3 FirstBolt 45
5.2.4 ClassyBolt 46

5.2.5 SQLBolt 47
5.2.6 PrepHDFSBolt 49
5.3 CRUD Interface 50
6. Αποτελέσματα - Συμπεράσματα 53
6.1 Twitter API 53
6.1.1 Αποτελέσματα από το Twitter API 53
6.1.2 Συμπεράσματα από το Twitter API 55
6.2 Sentiment Analysis 55
6.2.1 Αποτελέσματα από Sentiment Analysis 55
6.2.2 Συμπεράσματα από Sentiment Analysis 58
7. Μελλοντικές Επεκτάσεις 59
8. Βιβλιογραφία 61

1.Εισαγωγή

1.1 Γενικές πληροφορίες

Τα τελευταία χρόνια είναι γεγονός η τεράστια, και συνεχώς αυξανόμενη, ανάπτυξη των πλατφόρμων κοινωνικής δικτύωσης, τα λεγόμενα Social Networks. Εκατοντάδες εκατομμύρια άνθρωποι απ' όλον τον κόσμο, χρησιμοποιούνε τα Social Networks σε καθημερινή βάση για να δημοσιεύουν τις απόψεις τους ή για να επικοινωνούν μεταξύ τους καθώς επίσης για να έχουνε συνεχή ενημέρωση για το τι συμβαίνει στον κόσμο. Έτσι, δημιουργείται ένας τεράστιος όγκος δεδομένων σε καθημερινή βάση, που μπορεί να αξιοποιηθεί με ποικίλους τρόπους.

Ένα από τα κυρίαρχα Social Networks που υπάρχουν είναι το Twitter. Το Twitter δίνει έμφαση στην δημοσίευση σύντομων μηνυμάτων (140 χαρακτήρες), κάνοντας το ιδανικό για συνεχή σύντομα σχόλια. Με παραπάνω από 300 εκατομμύρια ενεργούς χρήστες κάθε μήνα, δημοσιεύονται περίπου 500 εκατομμύρια tweets την ημέρα σε όλον τον κόσμο. Λόγω αυτών των χαρακτηριστικών και του τεράστιου όγκου πληροφορίας, μέσω του Twitter μπορούμε να βρούμε πληροφορίες για οτιδήποτε συμβαίνει στον κόσμο και αφορά πτυχές της ανθρώπινης δραστηριότητας.

Μία ειδική πτυχή που μπορούμε να εντοπίσουμε στο Twitter είναι η αναφορά σε μηνύματα χρηστών σχετικά με παρενέργειες που μπορεί να αντιμετωπίζουν μετά από την χρήση φαρμάκων [11]. Η συνεχής αύξηση τέτοιων μηνυμάτων από τους χρήστες του Twitter, μπορεί να αξιοποιηθεί για την καλύτερη παρακολούθηση και συσχέτιση φαρμάκων με παρενέργειες. Σύμφωνα με έρευνες, το 90% των παρενεργειών από φάρμακα δεν αναφέρονται στις αρμόδιες αρχές, γεγονός που οδηγεί στην αναζήτηση τέτοιων αναφορών και από άλλες πηγές [10]. Στο Twitter, οι χρήστες μπορούνε πιο άνετα να εκφράσουν την δυσφορία για ένα φάρμακο, για τις παρενέργειες που μπορεί να έχει ή και για το αν ήταν αποτελεσματικό, που μπορεί να μην ανέφεραν ποτέ στον γιατρό τους. Προφανώς και δεν μπορεί να αντικατασταθεί ο ρόλος των γιατρών στον εντοπισμό παρενεργειών. Όμως η άντληση και παρατήρηση δεδομένων από τα Social Media, και στην περίπτωσή μας το Twitter, μπορεί να προσφέρει αρκετά μεγαλύτερες πιθανότητες για τον καλύτερο εντοπισμό μίας παρενέργειας. Ήδη, οι φαρμακοβιομηχανίες έχουν δημιουργήσει, έστω και σε πρωταρχικό στάδιο τέτοιους μηχανισμούς παρακολούθησης των Social Media και αναγνώρισης παρενεργειών από φάρμακα[10]. Επιπλέον, πολύ χρήστες χρησιμοποιούν το Twitter για να δηλώσουν εάν ένα φάρμακο ήταν αποτελεσματικό και αντιμετώπισε μία αρρώστια ή αν δεν είχε τα επιθυμητά αποτελέσματα. Τέτοιες πληροφορίες είναι πολύ χρήσιμες γιατί αποτελούν feedback για την αποτελεσματικότητα φαρμάκων του εμπορίου.

Η συνεχώς αυξανόμενη διαθέσιμη πληροφορία, που σχετίζεται με την ιατρική, απαιτεί και την αντίστοιχη ικανότητα από πλευράς ερευνητών για την συνεχή επεξεργασία της πληροφορίας αυτής. Η χρήση τεχνικών NLP (Natural Language Processing) καθώς και αλγορίθμων μηχανικής μάθησης, ανοίγουν νέους ορίζοντες για

την επεξεργασία του τεράστιου όγκου διαθέσιμων πληροφοριών και την κατεύθυνση για την αυτοματοποίηση της [12]. Μία πτυχή των τεχνικών NLP, που μπορεί να εφαρμοστεί στα δεδομένα από το Twitter αποτελεί η τεχνική του Sentiment Analysis. Δηλαδή η κατηγοριοποίηση ενός κειμένου, στην Εργασία μας ενός tweet, με βάση την γνώμη που εκφράζει αυτό, πχ αρνητική ή θετική. Υπάρχουν πολλές προσεγγίσεις για το Sentiment Analysis, καθώς και τεχνικές NLP, που έχουν αναπτυχθεί μέχρι στιγμής, όμως τα δεδομένα από τα Social Media παρουσιάζουν αρκετές προκλήσεις. Αυτό οφείλεται στη γλώσσα που χρησιμοποιείται στα Social Media που αποτελείται πολύ συχνά από ιδιωματισμούς, ορθογραφικά λάθη, χρήση "slang" γλώσσας, συχνή χρήση ειρωνείας, κτλ. που είναι πολύ διαφορετική από τη γλώσσα που χρησιμοποιείται σε δομημένα κείμενα. Έτσι, αποτελεί ένα ανοιχτό πεδίο ξεχωριστής έρευνας η προσαρμογή των τεχνικών NLP πάνω στην ανάλυση και στην επεξεργασία των δεδομένων από τα Social Media.

Τέλος, οι εφαρμογές για τέτοιου είδους επεξεργασία των δεδομένων των Social Media, στις οποίες έχουμε ανατρέξει, δεν γίνονται σε πραγματικό χρόνο. Εστιάζουν στο κατέβασμα ενός μεγάλου αριθμού δεδομένων από τα Social Media καθώς και άλλες ιντερνετικές πηγές και στην συνέχεια τα επεξεργάζονται με σκοπό την συνεχή βελτίωση αλγορίθμων NLP.

1.2 Αντικείμενο διπλωματικής

Στα πλαίσια της πλαίσια αυτής της Διπλωματικής εργασίας υλοποιήθηκε μία εφαρμογή η οποία σε πραγματικό χρόνο συλλέγει tweets που δημοσιεύονται στο Twitter και περιέχουν μία παρενέργεια από ένα φάρμακο και το όνομα ενός φαρμάκου. Δηλαδή, αλιεύουμε tweet στα οποία υπάρχει συσχέτιση μεταξύ φαρμάκων και παρενεργειών. Στην συνέχεια, πάλι σε πραγματικό χρόνο, τα tweets που λαμβάνουμε τα υποβάλουμε σε Sentiment Analysis για να δούμε εάν έχει θετικό, ουδέτερο ή αρνητικό νόημα. Τέλος, αποθηκεύουμε τα αποτελέσματα μας σε μία βάση δεδομένων έτσι ώστε να μπορούμε να τα έχουμε στη διάθεσή μας συνεχώς.

Για την υλοποίηση της εφαρμογής μας, χρησιμοποιήσαμε το Twitter API έτσι ώστε να έχουμε μία συνεχή ροή από εισερχόμενα tweets, από τα οποία κρατάμε μόνο αυτά που μας ενδιαφέρουν [5]. Για το Sentiment Analysis αυτών των tweet, χρησιμοποιήσαμε ένα open-source εργαλείο που προσφέρει πολλές NLP δυνατότητες, το LingPipe [17]. Επιπλέον, για την αποθήκευση των δεδομένων μας, χρησιμοποιήσαμε τη MySQL. Τέλος, το βασικότερο κομμάτι της εργασίας μας είναι το κατανεμημένο σύστημα Storm, πάνω στο οποίο αναπτύξαμε την εφαρμογή μας χρησιμοποιώντας τα παραπάνω εργαλεία.

Το Apache Storm είναι ένα ελεύθερο και ανοιχτού κώδικα κατανεμημένο σύστημα κατάλληλο για υπολογισμούς σε πραγματικό χρόνο. Το Storm καθιστά εύκολη και αξιόπιστη την επεξεργασία απεριόριστων ροών από δεδομένα, πραγματοποιώντας με επεξεργασία σε πραγματικό χρόνο ότι έκανε το Hadoop σε επεξεργασία

παρτίδας(batch processing). Αρχικά είχε αναπτυχθεί από την εταιρία Backtype(2011) όπου στη συνέχεια εξαγοράστηκε από την Twitter. Πλέον, λόγω και της άδειας χρήσης που προσφέρει το Apache Foundation, είναι πλήρως διαθέσιμο για τους χρήστες του.

Το Storm έχει πλέον πολλές περιπτώσεις χρήσης. Αναλύσεις σε πραγματικό χρόνο, online Μηχανική μάθηση, Συνεχείς Υπολογισμοί, Κατανεμημένο RPC είναι από τις περισσότερο διαδεδομένες περιπτώσεις χρήσης του Storm. Χρησιμοποιείται για όλες αυτές τις περιπτώσεις λόγω των χαρακτηριστικών του. Είναι πολύ γρήγορο, επεκτάσιμο, ανεκτικό σε σφάλματα, εγγυάται ότι θα γίνει η επεξεργασία όλων των δεδομένων και επίσης είναι εύκολο να το στήσεις και τα λειτουργεί[1]. Πιο συγκεκριμένα:

Γρήγορο: Ένα benchmark το χρονομέτρησε σε πάνω από ένα εκατομμύριο επεξεργασμένες πλειάδες ανά κόμβο ανά δευτερόλεπτο.

Επεκτάσιμο: Οι τοπολογίες του Storm εκτελούνται εξ αρχής παράλληλα και τρέχουν πάνω σε ένα σύμπλεγμα από μηχανήματα(cluster). Τα διάφορα μέρη της τοπολογίας μπορούν να κλιμακωθούν μεμονωμένα με μικροαλλαγές στον παραλληλισμό τους με πολύ εύκολο τρόπο.

Ανεκτικό σε σφάλματα: Όταν ένας worker στον cluster "πεθάνει", το Storm θα τον επανεκκινήσει αυτόματα. Στην περίπτωση που σταματήσει ένας ολόκληρος κόμβος, τότε το Storm θα αναθέσει τον worker σε έναν άλλον κόμβο.

Εγγυάται επεξεργασία δεδομένων: Ένας από τους βασικούς μηχανισμούς του Storm είναι η δυνατότητα να παρακολουθείται όλη η διαδρομή μίας πλειάδας μέσα στην τοπολογία. Έτσι μπορεί το Storm να εντοπίσει που έχει γίνει λάθος για μία πλειάδα και να φροντίσει για την επεξεργασία της.

Ευκολία χρήσης: Ένας Storm cluster είναι εύκολο να αναπτυχθεί, απαιτεί ελάχιστες ρυθμίσεις και διαμόρφωση για να είναι έτοιμο να λειτουργήσει. Ο διαμορφώσεις που έχει έτοιμες το Storm το καθιστούν κατάλληλο για την παραγωγή.

1.3 Οργάνωση κειμένου

Η Διπλωματική εργασία χωρίζεται σε 8 κεφάλαια. Το Κεφάλαιο 1 είναι μία εισαγωγή σε αυτήν την Διπλωματική, δίνοντας κάποιες γενικές πληροφορίες για τα θέματα που καταπιανόμαστε στην εργασία και ιδιαίτερα γιατί την υλοποιήσαμε πάνω στο Storm. Στο Κεφάλαιο 2 αναλύουμε το θεωρητικό υπόβαθρο και τις βασικές στις οποίες βασίζεται το Storm. Στο 3ο Κεφάλαιο γίνεται ανάλυση του Twitter API και όλων των χαρακτηριστικών που χρησιμοποιήσαμε στην υλοποίησή μας. Στο Κεφάλαιο 4 παρουσιάζουμε το θεωρητικό υπόβαθρο γύρω από το Sentiment Analysis, περιγράφοντας τις κυριότερες πρακτικές. Επίσης αναφέρουμε πληροφορίες για τα αντίστοιχα εργαλεία που χρησιμοποιήσαμε. Στο Κεφάλαιο 5 περιγράφουμε την υλοποίηση της Διπλωματικής μας. Εξηγούμε την λειτουργία όλων των στοιχείων της τοπολογίας μας, τα Bolt και τα Spout καθώς και κάποιων εργαλείων τα οποία τα χρησιμοποιήσαμε στη συνέχεια. Στο επόμενο Κεφάλαιο, το 6ο, παραθέτουμε τα αποτελέσματα που προέκυψαν από την εκτέλεση της τοπολογίας μας και κάποια πρώτα συμπεράσματα που βγάλαμε από αυτά. Στο 7ο Κεφάλαιο κάνουμε αναφορά σε μελλοντικές επεκτάσεις για την Διπλωματικής μας και στη συνέχεια παραθέτουμε τις Βιβλιογραφικές μας Αναφορές.

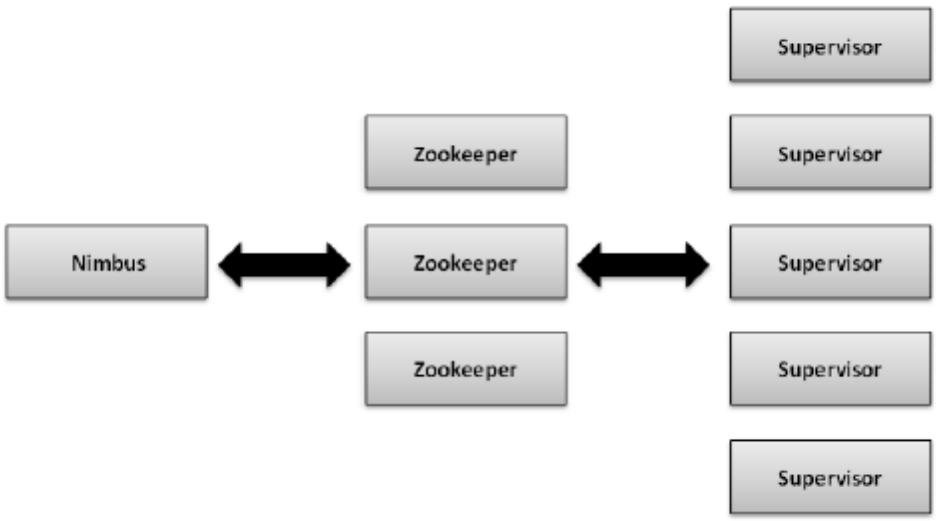
2. Storm

2.1 Γενική Αρχιτεκτονική του Storm

Ένα σύμπλεγμα Storm (Storm cluster) αποτελείται από 2 είδη κόμβων: τον κόμβο Master και τους κόμβους Workers. Ο κόμβος Master τρέχει ένα Nimbus, δηλαδή είναι υπεύθυνο για την παρακολούθηση του cluster για τυχόν λάθη και την εκτέλεση των λειτουργιών που απαιτούνται σε αυτή την περίπτωση, αναθέτει καθήκοντα στα μηχανήματα του cluster και κατανέμει κατάλληλα τον κώδικα σε όλο το cluster. Σε ένα Storm cluster υπάρχει μόνο ένας κόμβος Master. Κάθε κόμβος Worker αποτελείται από δύο στιγμιότυπα: μία ή παραπάνω διεργασίες worker (worker process) και ένα στιγμιότυπο από μία Επιβλέπουσα διεργασία (Supervisor daemon). Ο Supervisor επιβλέπει το πότε σταματάνε ή αρχίζουν τα worker processes στον κόμβο, ελέγχοντας τον όγκο εργασιών που έχει ανατεθεί στον κόμβο από το Nimbus. Κάθε worker process ενός κόμβου εκτελεί ένα κομμάτι από μία τοπολογία. Έτσι, μία τοπολογία που τρέχει στον cluster αποτελείται από πολλά worker processes που είναι διασκορπισμένα σε πολλά μηχανήματα.

Το Storm απαιτεί έναν Zookeeper ο οποίος συντονίζει το Nimbus με τους Supervisors που υπάρχουν σε κάθε κόμβο του cluster. Ο Zookeeper είναι μία κεντρικοποιημένη υπηρεσία η οποία διατηρεί τις πληροφορίες για τις ρυθμίσεις του cluster, δίνει ονόματα στους κόμβους, παρέχει κεντρικοποιημένο συγχρονισμό και παρέχει ομαδικές υπηρεσίες στον cluster. Επίσης αποθηκεύει και τις καταστάσεις του Nimbus και των Supervisor.[3]

Μια σχηματική αναπαράσταση της αρχιτεκτονικής του Storm είναι η εξής:



Εικόνα 2.1 Αρχιτεκτονική Συμπλέγματος Storm[4]

2.2 Κύριες Έννοιες του Storm

Το Storm αποτελείται από διαφορετικά στοιχεία, καθένα από τα οποία είναι υπεύθυνο για την διαχείριση συγκεκριμένων κομματιών της επεξεργασίας. Στη συνέχεια του κεφαλαίου θα αναλύσουμε τις σημαντικότερες έννοιες του Storm.

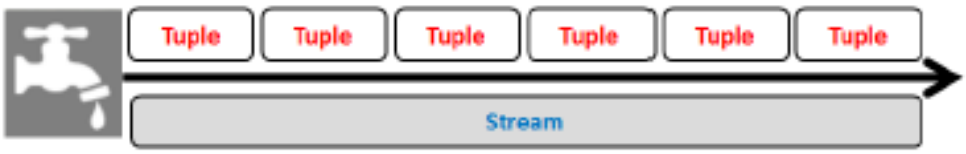
2.2.1 Μοντέλο δεδομένων

Τα διαφορετικά στοιχεία που αποτελούν μία εφαρμογή, χρησιμοποιούν για την μεταφορά των δεδομένων μεταξύ τους την πλειάδα (tuple).

Ένα tuple αποτελείται από μία λίστα με προκαθορισμένο αριθμό πεδίων, όπου κάθε πεδίο μπορεί να έχει διαφορετικό τύπο δεδομένων. Τα δεδομένα μπορεί να αποτελούνται από γνωστούς τύπους δεδομένων (int, char, float, String arrays, …) ή μπορούμε μέσω του Storm να φτιάξουμε κάποιον δικό μας τύπο δεδομένων που θα υπάρχει σε ένα tuple. Το tuple έχει δυναμικούς τύπους δεδομένων, έτσι ώστε να μην χρειάζεται να καθορίσεις τον τύπο δεδομένων ενός tuple, αλλά μόνο τα ονόματα τους. Μπορούμε να επεξεργαστούμε κάθε στοιχείο ενός tuple χρησιμοποιώντας είτε το όνομά του, είτε τη θέση του στη λίστα του tuple [3].

2.2.2 Spout

Η πηγή των δεδομένων για κάθε τοπολογία είναι τα Spout. Τα Spout διαβάζουν δεδομένα από εξωτερικές πηγές (π.χ. APIs, αρχεία, Βάσεις Δεδομένων, ...) και στη συνέχεια τα μεταδίδουν στα άλλα στοιχεία της τοπολογίας για επεξεργασία (Εικόνα 2.2). Ένα Spout μπορεί να προγραμματιστεί ώστε να στέλνει δεδομένα σε πολλές ροές .



Εικόνα 2.2 Γραφικό παράδειγμα ενός τυπικού Spout

Οι κύριες μέθοδοι ενός Spout είναι οι εξής (*Εικόνα 2.3*):

open() : Η μέθοδος αυτή καλείται μία φορά κατά την αρχικοποίηση του Spout. Σε αυτήν την μέθοδο υλοποιούμε τη σύνδεση του Spout με την εξωτερική πηγή δεδομένων και στην συνέχεια τα προωθούμε στην μέθοδο nextTuple().

nextTuple(): Αυτή η μέθοδος αποτελεί τον πυρήνα ενός Spout. Καλείται συνεχώς για να παίρνει το επόμενο διαθέσιμο tuple, αν υπάρχει, και να το μεταδώσει στην τοπολογία.

ack(): Η μέθοδος καλείται όταν το προηγούμενο tuple που έστειλε το Spout το έλαβε επιτυχώς ο παραλήπτης του.

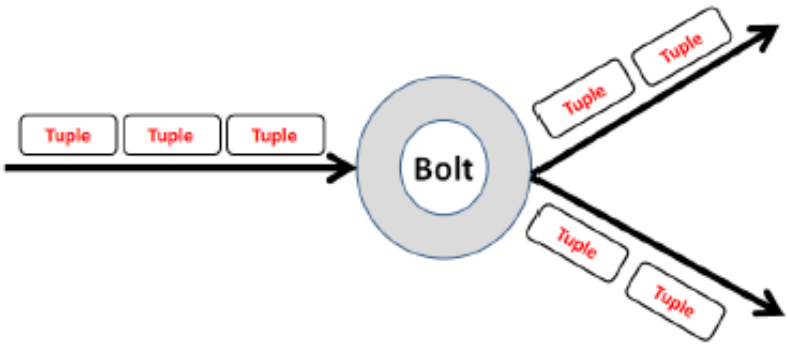
fail(): Η μέθοδος καλείται όταν ένα tuple δεν έφτασε στον παραλήπτη του. Μπορεί να ξαναβάλει πάλι το tuple στην σειρά για αποστολή είτε και να τερματίσει το process εάν έχουν προκύψει πάρα πολλές αποτυχίες.

```
public class RandomSentenceSpout extends BaseRichSpout {
  SpoutOutputCollector _collector;
  Random _rand;


  @Override
  public void open(Map conf, TopologyContext context, SpoutOutputCollector collector) {
    _collector = collector;
    _rand = new Random();
  }

  @Override
  public void nextTuple() {
    Utils.sleep(100);
    String[] sentences = new String[]{ "the cow jumped over the moon", "an apple a day keeps the doctor away",
        "four score and seven years ago", "snow white and the seven dwarfs", "i am at two with nature" };
    String sentence = sentences[_rand.nextInt(sentences.length)];
    _collector.emit(new Values(sentence));
  }

  @Override
  public void ack(Object id) {
  }

  @Override
  public void fail(Object id) {
  }

  @Override
  public void declareOutputFields(OutputFieldsDeclarer declarer) {
    declarer.declare(new Fields("word"));
  }

}
```

Εικόνα 2.3 Παράδειγμα Sample Κώδικα ενός Spout[2]

2.2.3 Bolt

Τα Bolt αποτελούν το κύριο στοιχείο μίας τοπολογίας Storm. Ουσιαστικά είναι τα στοιχεία της τοπολογίας στα οποία γίνεται όλη η επεξεργασία των δεδομένων που λαμβάνουμε. Τα Bolt δέχονται ως είσοδο τα tuples, τα επεξεργάζεται ανάλογα με τις συναρτήσεις που έχει ορίσει ο χρήστης και στη συνέχεια στέλνει τα εξερχόμενα tuples σε άλλα Bolts. Όπως και τα Spout, τα Bolt μπορούνε να στέλνουν tuples σε πολλαπλές ροές καθώς επίσης και να δέχεται tuples από διαφορετικές ροές(Εικόνα 2.4).



Εικόνα 2.4 Γραφικό παράδειγμα ενός τυπικού Bolt

Οι κύριες μέθοδοι σε ένα Bolt είναι (*Εικόνα 2.5*):

prepare(): Η μέθοδος αυτή καλείται πριν αρχίσει το Bolt να δέχεται tuples. Με αυτή την συνάρτηση αρχικοποιείται το Bolt ανάλογα και με τη λειτουργικότητα του κάθε Bolt.

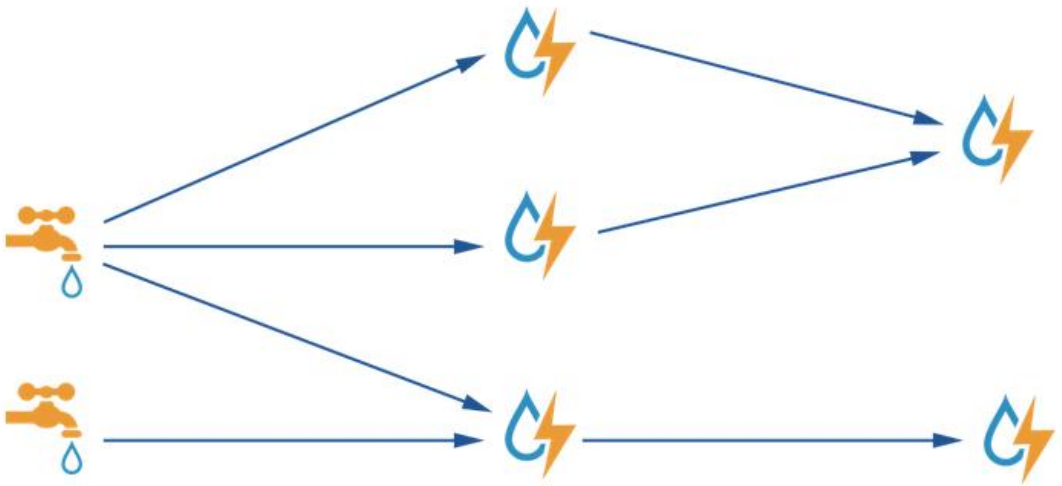
execute(): Η μέθοδος αυτή αποτελεί την κύρια μέθοδο του Bolt. Η execute εκτελείται κάθε φορά που έρχεται ένα tuple στο Bolt. Σε αυτή τη μέθοδο γίνεται η επεξεργασία των εισερχόμενων tuples και το αποτέλεσμα της επεξεργασίας είτε εκπέμπεται ως tuple στην ροή εξόδου του Bolt, είτε εκτελείται κάποια άλλη ενέργεια (π.χ. εγγραφή σε αρχείο, αποθήκευση σε Βάση Δεδομένων).

```
public class RollingCountAggBolt extends BaseRichBolt {
  private static final long serialVersionUID = 5537727428628598519L;
  private static final Logger LOG = Logger.getLogger(RollingCountAggBolt.class);
  //Mapping of key->upstreamBolt->count
  private Map<Object, Map<Integer, Long>> counts = new HashMap<Object, Map<Integer, Long>>();
  private OutputCollector collector;


  @SuppressWarnings("rawtypes")
  @Override
  public void prepare(Map stormConf, TopologyContext context, OutputCollector collector) {
    this.collector = collector;
  }

  @Override
  public void execute(Tuple tuple) {
    Object obj = tuple.getValue(0);
    long count = tuple.getLong(1);
    int source = tuple.getSourceTask();
    Map<Integer, Long> subCounts = counts.get(obj);
    if (subCounts == null) {
      subCounts = new HashMap<Integer, Long>();
      counts.put(obj, subCounts);
    }
    //Update the current count for this object
    subCounts.put(source, count);
    //Output the sum of all the known counts so for this key
    long sum = 0;
    for (Long val: subCounts.values()) {
      sum += val;
    }
    collector.emit(new Values(obj, sum));
  }

  @Override
  public void declareOutputFields(OutputFieldsDeclarer declarer) {
    declarer.declare(new Fields("obj", "count"));
  }
}
```

Εικόνα 2.5 Παράδειγμα Sample Κώδικα ενός Bolt[2]

2.2.4 Τοπολογία Storm

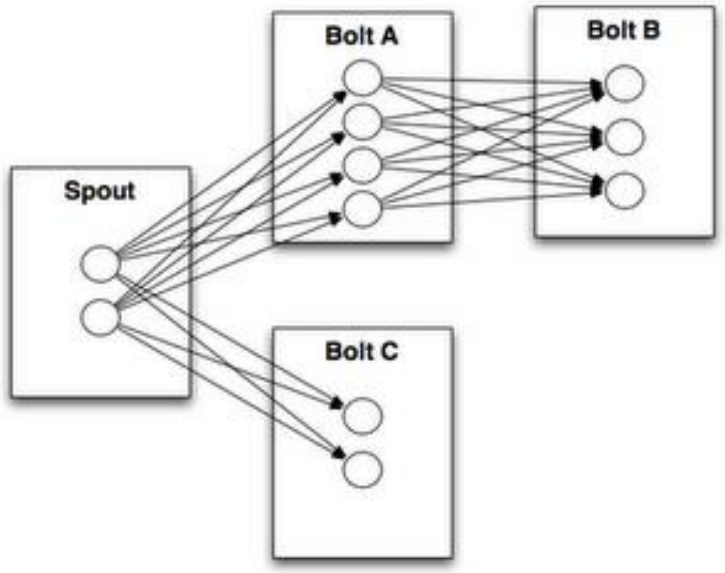
Μία συγκεκριμένη διάταξη από Spouts και Bolts και η σύνδεση μεταξύ αυτών των στοιχείων αποτελεί μία τοπολογία. Η τοπολογία ουσιαστικά περιέχει όλη την λογική της εφαρμογής και μέσω αυτής την εκτελούμε. Μία τοπολογία μπορεί να αναπαρασταθεί με έναν άκυκλο κατευθυνόμενο γράφο, όπου κάθε κόμβος προωθεί στον επόμενο κόμβο κάποια δεδομένα τα οποία τα έχει επεξεργαστεί. Στην *Εικόνα 2.6* παρατηρούμε πως συνδέονται μεταξύ τους οι κόμβοι μίας τοπολογίας. Οι κόμβοι μίας τοπολογίας τρέχουνε σε παραλληλία. Το επίπεδο της παραλληλίας τους ορίζεται κατά τον ορισμό της τοπολογίας και αντιπροσωπεύει τον αριθμό των νημάτων (threads) που δημιουργούνται για την εκτέλεση των αντίστοιχων εργασιών. Μία τοπολογία συνεχίζει να εκτελείται στον cluster έως ότου ρητώς τερματιστεί.



Εικόνα 2.6 Αφαιρετική μορφή μίας Τοπολογίας

2.3.3 Ροές Δεδομένων

Σε μία τοπολογία στο Storm, είναι απαραίτητο να ορίσουμε ποιες ροές θα λαμβάνει κάθε Bolt και που θα στέλνει της ροές εξόδου. Αυτό επιτυγχάνεται μέσω των ομαδοποιήσεων ροών - stream grouping (Εικόνα 2.7). Επειδή η ομαδοποίηση ροών αποτελεί σημαντική πτυχή μιας τοπολογίας, το Storm προσφέρει κάποιες έτοιμες ομαδοποιήσεις ροών ενώ επιτρέπει και την δημιουργία δικών μας ομαδοποιήσεων.



Εικόνα 2.7 Παράδειγμα από πιθανές ροές δεδομένων σε μία τοπολογία

Οι ομαδοποιήσεις ροών που προσφέρει το Storm είναι:

Τυχαία ομαδοποίηση (shuffle): Τα tuples κατανέμονται με τυχαίο τρόπο μεταξύ των task σε ένα Bolt το οποίο έχει ως αποτέλεσμα κάθε task να δέχεται ίσο αριθμό από tuples.

Ομαδοποίηση πεδίων (fields): Η ροή χωρίζεται σύμφωνα με τα πεδία που έχουν οριστεί σε αυτή. Έτσι, ένα δεδομένο σύνολο τιμών των πεδίων θα στέλνεται στο ίδιο task του Bolt.

Ομαδοποίηση Όλων (all): Τα tuples μία ροής στέλνονται σε όλα τα task του Bolt.

Γενική ομαδοποίηση (global): Όλα τα tuples της ροής στέλνονται σε ένα task του Bolt, σε αυτό με το χαμηλότερο ID.

Καμία ομαδοποίηση (none): Προς το παρών, η καμία ομαδοποίηση λειτουργεί ως τυχαία ομαδοποίηση. Στο μέλλον, τα tuples θα πηγαίνουν στο task ενός Bolt που είναι αντίστοιχο με αυτό από το οποίο στάλθηκαν.

Στοχευμένη ομαδοποίηση (direct): Το task που παράγει το tuple θα αποφασίζει σε ποιο task, του Bolt που θα το λάβει, θα στείλει το tuple.

Τοπική/Τυχαία ομαδοποίηση (local/shuffle): Τα tuples από ένα task θα πηγαίνουν σε task που είναι στον ίδιο worker στον cluster. Αν δεν είναι δυνατόν, λειτουργεί ως τυχαία ομαδοποίηση.

Ομαδοποίηση μερικού κλειδιού (partial key): Η ροή διαχωρίζεται βάση των πεδίων που έχουν προσδιοριστεί στην ομαδοποίηση, όπως και στην ομαδοποίηση πεδίων, αλλά ο φόρτος μοιράζεται στα bolt που έχουν είσοδο από αυτή τη ροή. Αυτός ο τρόπος παρέχει καλύτερη χρήση των πόρων.

2.3 Storm UI

Το Storm παρέχει ένα γραφικό περιβάλλον (UI) όπου μπορούμε να δούμε όλες τις πληροφορίες που αφορούν τον Storm cluster. Μπορούμε να δούμε ποιες τοπολογίες τρέχουν αυτή τη στιγμή στον cluster, τυχόν λάθη που συμβαίνουν κατά την εκτέλεση μιας τοπολογίας καθώς και διάφορα στατιστικά για τον cluster (Εικόνα 2.8).

Storm UI

Cluster Summary

Version	Nimbus uptime	Supervisors	Used slots	Free slots	Total slots	Executors	Tasks
0.9.5	3h 25m 43s	18	0	36	36	0	0

Topology summary

Name	Id	Status	Uptime	Num workers	Num executors	Num tasks

Supervisor summary

Id	Host	Uptime	Slots	Used slots
4f6786ce-fe84-4f67-8eac-8d0862378264	clu06.softnet.tuc.gr	3h 23m 56s	3	0
6c8df7bd-fe81-459f-8843-15bb1522ec16	clu03.softnet.tuc.gr	3h 25m 28s	5	0
20bab5f3-74ab-47db-bace-ecd23e6ebbd8	clu16.softnet.tuc.gr	3h 25m 40s	1	0
59fa9ba7-cf53-499a-abb8-4e730c2e98df	clu10.softnet.tuc.gr	3h 24m 7s	1	0
88f8c145-b9e4-4a44-a159-492237d1b159	clu20.softnet.tuc.gr	3h 25m 39s	1	0
162d22ea-708b-4227-8a99-8f02c13d39b7	clu25.softnet.tuc.gr	3h 25m 36s	1	0
804fa430-5789-46f4-b6b2-71885a6eead6	clu26.softnet.tuc.gr	3h 25m 25s	1	0
968c1898-2932-49f0-abe8-d963e6157ec9	clu02.softnet.tuc.gr	3h 24m 3s	5	0
6643a889-1c21-44b8-aeae-ca2fbdeded8f	clu09.softnet.tuc.gr	3h 24m 11s	1	0
7614c24c-26b9-4022-8ad8-cb8ee45e0c68	clu24.softnet.tuc.gr	3h 25m 24s	1	0
1210779b-ad04-42f2-95e8-33f24e752fd4	clu05.softnet.tuc.gr	3h 24m 4s	3	0
a541ea98-a1fa-427b-8954-d4ef0aa0236b	clu19.softnet.tuc.gr	3h 25m 30s	1	0
aec0f530-a71a-4202-bd14-8e89821de63b	clu07.softnet.tuc.gr	3h 23m 57s	3	0
b660c768-312c-454c-97c7-2f90d4bf2028	clu14.softnet.tuc.gr	3h 25m 41s	1	0
e967898d-7ce1-4423-8f1b-5507bbc36e44	clu04.softnet.tuc.gr	3h 25m 38s	5	0

Εικόνα 2.8 Το UI του Storm του cluster που είναι στημένος στο SoftNet

2.4 Hadoop Distributed File System (HDFS)

Σε αυτήν την εργασία παράλληλα με το Storm, χρησιμοποιούμε και ένα module από το Hadoop, το Hadoop Distributed File System (HDFS).

Το Apache Hadoop είναι ένα open-source project το οποίο χρησιμοποιείται για κατανεμημένο υπολογισμό μεγάλου όγκου δεδομένων. Τα κύρια χαρακτηριστικά του είναι η αξιοπιστία και η επεκτασιμότητα. Η βιβλιοθήκη λογισμικού που διαθέτει περιλαμβάνει ένα πλαίσιο με προγραμματιστικά μοντέλα που του επιτρέπει κατανεμημένη επεξεργασία δεδομένων σε cluster [20]. Έχει σχεδιαστεί για να τρέχει είτε σε ένα μόνο μηχάνημα είτε και σε πάρα πολλά, επιτυγχάνοντας έτσι την επεκτασιμότητά του. Το Hadoop περιέχει διάφορα εργαλεία για να επιτυγχάνει τον σκοπό του, εμείς όμως χρησιμοποιήσαμε μόνο το HDFS.

Ενώ στο σύστημα μας χρησιμοποιούμε το Storm για την επεξεργασία των δεδομένων στον cluster, χρησιμοποιούμε το HDFS για να αποθηκεύουμε δεδομένα που χρησιμοποιούμε στο σύστημα μας στο κατανεμημένο σύστημα αρχείων που προσφέρει. Το HDFS είναι γραμμένο σε Java και ακολουθεί την αρχιτεκτονική master/slave. Ο HDFS cluster περιλαμβάνει έναν κόμβο NameNode, έναν master server όπου είναι υπεύθυνος για την διαχείριση του συστήματος αρχείων και να κανονίζει την πρόσβαση στο σύστημα από τους πελάτες (clients). Ο κόμβος NameNode συνοδεύεται από ένα αριθμό από DataNodes, οι οποίοι διαχειρίζονται τον αποθηκευτικό χώρο του κόμβου που βρίσκονται(συνήθως κάθε κόμβος στον cluster έχει έναν DataNode). Το HDFS παρέχει ένα Java API για την πρόσβαση στα αρχεία του, αλλά μπορεί να παράξει clients και για άλλες γλώσσες προγραμματισμού.

3. Twitter API

Οι χρήστες του Twitter παράγουν περισσότερα από 500 εκατομμύρια tweets σε καθημερινή βάση. Ένα ποσοστό των μηνυμάτων αυτών είναι διαθέσιμο σε ερευνητές αλλά και άλλους ανθρώπους μέσω του Public API που παρέχει το Twitter χωρίς κανένα κόστος [5]. Σε αυτό το κεφάλαιο θα αναλύσουμε το τρόπο με τον οποίο αξιοποιήσαμε το API για να έχουμε πρόσβαση και να εξάγουμε τα δεδομένα του Twitter για να τα αξιοποιήσουμε στην διπλωματική αυτή εργασία.

3.1 Τύποι του Twitter API

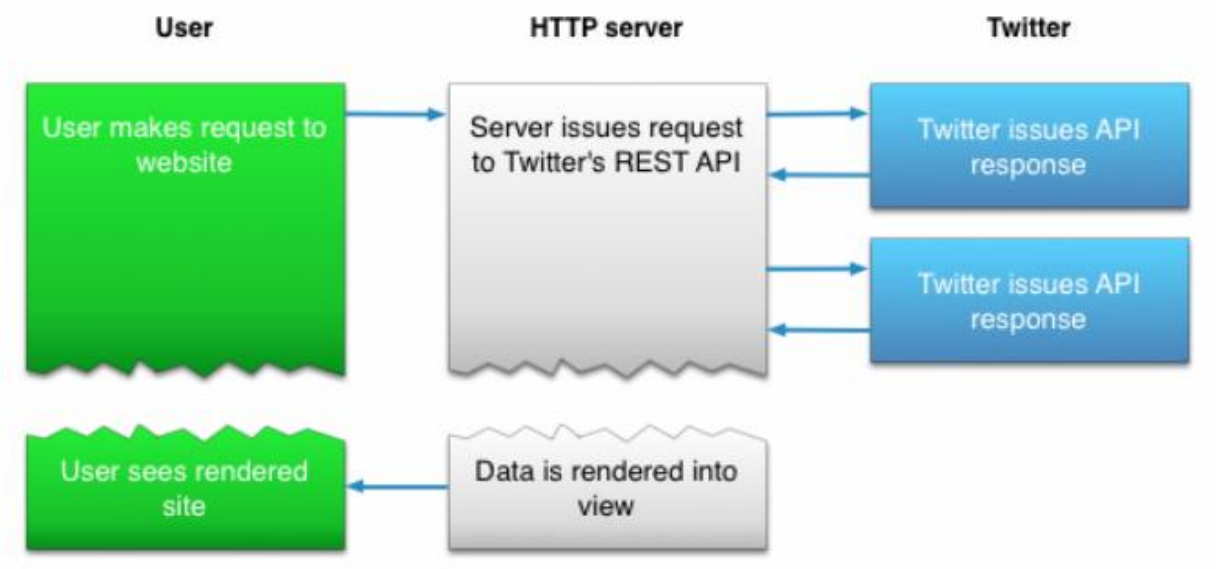
Το public API του Twitter μπορούμε να το κατηγοριοποιήσουμε σε δύο τύπους, βασιζόμενοι στην διαφορετική σχεδίαση τους καθώς και στο πως αποκτούν πρόσβαση στα δεδομένα του Twitter. Οι δύο τύποι είναι:

REST API: Βασίζεται στην αρχιτεκτονική REST, που χρησιμοποιείται ευρέως στον σχεδιασμό εφαρμογών Web. Για την ανάκτηση δεδομένων χρησιμοποιείται η μέθοδος PULL. Για να συλλέξει τα δεδομένα, ο χρήστης πρέπει να το απαιτήσει ρητά.

STREAMING API: Παρέχει μία συνεχής ροή από πληροφορίες που είναι διαθέσιμες δημόσια στο Twitter. Το API αυτό, χρησιμοποιεί τη μέθοδο PUSH για την συλλογή των δεδομένων. Από τη στιγμή που θα γίνει η αίτηση για τα δεδομένα, το API παρέχει μία συνεχής ροή δεδομένων χωρίς ο χρήστης να χρειάζεται να δώσει κάποια άλλη είσοδο.

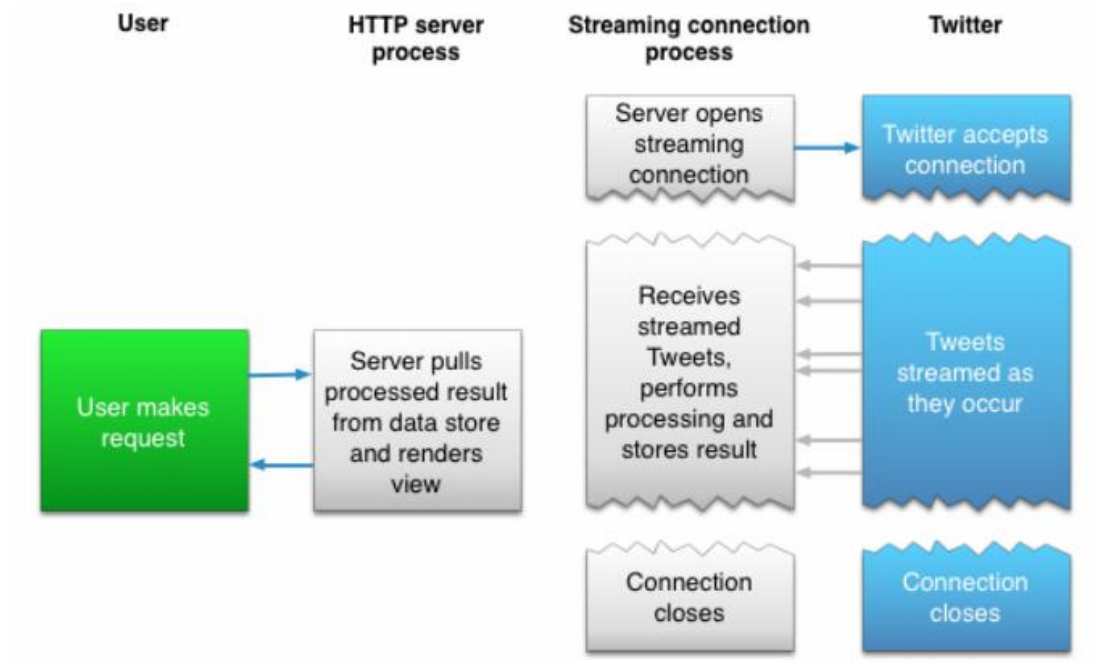
Όπως είναι εύκολα κατανοητό, το είδος της εφαρμογής που θέλουμε να σχεδιάσουμε καθορίζει και τον τύπο του API που θα χρησιμοποιήσουμε.

Αν λάβουμε ως παράδειγμα, μία Web εφαρμογή που δέχεται διαφορετικά αιτήματα από χρήστες, κάνει αντίστοιχα ένα ή περισσότερα αιτήματα στο Twitter API και τυπώνει το αποτέλεσμα, ως απάντηση στην αρχική αίτηση που είχε κάνει ο χρήστης. Η εφαρμογή αυτή θα χρησιμοποιεί το REST API αφού τα αιτήματα προς το Twitter API έχουν την παρακάτω δομή (Εικόνα 3.1):



Εικόνα 3.1 Παράδειγμα σύνδεσης στο Twitter API μέσω REST API [5]

Στην άλλη περίπτωσή, όπου μία εφαρμογή συνδεθεί με το Streaming API, δεν θα μπορεί να συνδεθεί με το Twitter API σε κάθε αίτημα ενός χρήστη, όπως στο παραπάνω παράδειγμα. Αντ' αυτού, ο κώδικας ο οποίος διατηρεί την σύνδεση με το Streaming API, τρέχει σε μία διαφορετική διεργασία από την διεργασία όπου χειρίζεται τα αιτήματα HTTP (Εικόνα 3.2):



Εικόνα 3.2 Παράδειγμα σύνδεσης στο Twitter API μέσω Streaming API [5]

Η διαδικασία που ακολουθείται κατά το streaming των δεδομένων παίρνει τα tweets εισόδου, πραγματοποιεί όποιο parsing ή φιλτράρισμα χρειάζεται και μετά αποθηκεύει τα αποτελέσματα σε μία αποθηκευτική δομή. Η διεργασία που χειρίζεται τα αιτήματα HTTP, συνδέεται με αυτή την αποθηκευτική δομή για να έχει τα αποτελέσματα από τα αιτήματα. Το μοντέλο του Streaming API μπορεί να είναι λίγο πιο περίπλοκο, όμως καταφέρνει να έχει μία συνεχής ροή από tweets, γεγονός που μπορεί να αξιοποιηθεί από πολλές εφαρμογές.

Για την υλοποίηση της εφαρμογής στα πλαίσια αυτής της Διπλωματικής, χρειαζόμαστε τη συνεχή ροή από tweets, σε πραγματικό χρόνο, για να μπορούμε στην συνέχεια να τα επεξεργαζόμαστε με τον τρόπο που θέλουμε. Για αυτό το λόγο χρησιμοποιούμε το Streaming API του Twitter αντί του REST.

3.2 Streaming API

Το Streaming API, όπως αναφέραμε και στην εισαγωγή, προσφέρει στους προγραμματιστές πρόσβαση στο παγκόσμιο stream από δεδομένα του Twitter [6].

Το Twitter προσφέρει διαφορετικά endpoints στα οποία συνδέεται το Streaming API, καθένα από τα οποία είναι σχεδιασμένο για διαφορετικές περιπτώσεις χρήσεις. Αυτά είναι:

Public Streams

Περιέχουν ροές δεδομένων που κυκλοφορούνε δημόσια στο Twitter. Είναι κατάλληλα για να ακολουθείς συγκεκριμένα θέματα στο Twitter ή συγκεκριμένους χρήστες καθώς και για data mining.

User Streams

Περιέχουν ροές που αφορούνε σχεδόν όλα τα δεδομένα που σχετίζονται με ένα συγκεκριμένο χρήστη του Twitter.

Site Streams

Ουσιαστικά αποτελεί την έκδοση των User Streams για πολλούς χρήστες. Τα Site Streams απευθύνονται σε εφαρμογές που συνδέονται στο Twitter από πολλαπλούς χρήστες.

Firehose

Για τη σύνδεση σε αυτό το endpoint απαιτείται ειδική άδεια, η οποία παρέχεται σε ειδικές περιπτώσεις. Με αυτό, ένας χρήστης έχει πρόσβαση σε όλα τα δημόσια tweets που δημιουργούνται.

Με βάση τον σκοπό της εργασίας μας και έχοντας υπ' όψιν τα παραπάνω χαρακτηριστικά των διαφορετικών endpoint, στην υλοποίησή μας επιλέγουμε να συνδεθούμε σε ένα Public Stream.

3.2.1 Σύνδεση στο Streaming API

Η σύνδεση με οποιοδήποτε API του Twitter μπορεί να πραγματοποιηθεί μόνο με πιστοποιημένα αιτήματα. Για την πιστοποίηση των αιτημάτων, το Twitter χρησιμοποιεί το Open Authentication (OAuth).

Επειδή οι κωδικοί είναι εύκολο να παραβιαστούν, το OAuth παρέχει έναν ασφαλέστερο τρόπο για να πραγματοποιείται η σύνδεση με τα API του Twitter. Με αυτόν τον τρόπο, ο κωδικός ενός χρήστη δεν εμφανίζεται πουθενά και ούτε κοινοποιείται με άλλες εφαρμογές. Για να χρησιμοποιήσει ένας χρήστης το Twitter API, θα πρέπει να δημιουργήσει μία εφαρμογή στη σχετική σελίδα του Twitter, μέσω της οποίας θα επικοινωνεί τελικά με το API.

Η εφαρμογή ενός χρήστη, για να μπορεί να χρησιμοποιήσει το OAuth, θα πρέπει:

- **Να αποκτήσει τα Access Tokens**

  Τα Access Tokens είναι 4 μοναδικοί κωδικοί. Αποτελούνται από το Consumer Key και Secret, τα οποία παράγονται και αφορούν την σύνδεση μεταξύ της εφαρμογής και του χρήστη του Twitter που την δημιούργησε. Στη συνέχεια, αφού εισάγεις τους δύο αυτούς κωδικούς, δημιουργούνται άλλοι δύο κωδικοί, τα Access Token και Secret. Τα δύο αυτά tokens ουσιαστικά πιστοποιούν την εφαρμογή με το Twitter API για να μπορεί να συνδεθεί με αυτό και να κάνει τα αιτήματα που θέλει ο χρήστης.

- **Να πιστοποιεί όλα τα αιτήματα HTTP που κάνει στο Twitter API με βάση τα Access Tokens της εφαρμογής**

  Για κάθε αίτημα HTTP που κάνει ο χρήστης θα πρέπει να παρέχει και τα tokens που έχουν δημιουργηθεί για την εφαρμογή του για να μπορεί το κάθε αίτημα να εκτελεσθεί από το Twitter API.

3.2.2 Streaming API Request Parameters

Κατά την εκτέλεση ενός HTTP αιτήματος , υπάρχει δυνατότητα ο χρήστης να ορίσει κάποιες παραμέτρους με σκοπό να ορίσει τι είδους δεδομένα μπορούν να επιστρέφονται από ένα Streaming API endpoint. Η δυνατότητα αυτή είναι εξαιρετικά χρήσιμη, καθώς ο χρήστης μπορεί να ορίσει της παραμέτρους αυτές έτσι ώστε να εμποδίσει να του επιστρέφονται δεδομένα που είναι άχρηστα για την εκάστοτε

εφαρμογή. Στα πλαίσια της εργασίας μας, αυτές οι παράμετροι μας ήταν εξαιρετικά χρήσιμοι στο να φιλτράρουμε τα δεδομένα που λαμβάνουμε.

Οι πιο σημαντικές παράμετροι είναι:

language: σε αυτήν την παράμετρο μπορούμε να ορίσουμε σε μία λίστα αναγνωριστικά γλωσσών που υπάρχουν στην λίστα BCP 47. Έτσι μπορούμε να περιορίσουμε τα εισερχόμενα tweets ώστε να είναι μόνο στις γλώσσες που έχουμε επιλέξει.

follow: με αυτήν την παράμετρο μπορούμε να λαμβάνουμε tweets που παράγονται μόνο από συγκεκριμένους χρήστες και όχι απλά τυχαία tweets. Αυτό γίνεται εισάγοντας σε μία λίστα τα ID των χρηστών του Twitter που θέλουμε να ακολουθήσουμε.

locations: με αυτήν την παράμετρο μπορούμε να συλλέγουμε tweets από μία συγκεκριμένη περιοχή που θα ορίσουμε. Αυτό γίνεται δίνοντας μια λίστα με ζευγάρια συντεταγμένων που ορίζουν ένα «κουτί» γύρω από την περιοχή που θέλουμε.

track: με την παράμετρο αυτή μπορούμε να λαμβάνουμε tweets που περιέχουν συγκεκριμένες λέξεις ή φράσεις. Αυτό γίνεται εισάγοντας μία λίστα με φράσεις, χωρισμένες με κόμματα. Μία φράση μπορεί να έχει έναν ή παραπάνω όρους, χωρισμένους με κενά. Λαμβάνουμε ένα tweet μόνο εάν υπάρχουν όλοι οι όροι μίας φράσης μέσα στο tweet, αγνοώντας πεζά-κεφαλαία, σημεία στίξης και την σειρά εμφανίσεις των όρων. Κάθε φράση πρέπει να είναι το πολύ 60 bytes. Δυστυχώς, το ακριβές ταίριασμα των φράσεων (quote) που εισάγουμε δεν υποστηρίζεται από το API.

3.3 Twitter API Libraries

Η πρόσβαση στο Twitter API πραγματοποιείται μέσω βιβλιοθηκών που έχουν δημιουργηθεί για αυτό το σκοπό. Καλύπτεται ένα ευρύ φάσμα γλωσσών προγραμματισμού και έτσι ο χρήστης δεν θα έχει πρόβλημα να δημιουργήσει μια εφαρμογή σε όποια γλώσσα προγραμματισμού το επιθυμεί. Αξίζει να σημειωθεί ότι οι περισσότερες από αυτές τις βιβλιοθήκες έχουν δημιουργηθεί από τρίτους.

Για την υλοποίηση της εργασίας μας, χρησιμοποιήσαμε την βιβλιοθήκη Twitter4j. Μία βιβλιοθήκη για την γλώσσα προγραμματισμού Java. Έχει δημιουργηθεί από έναν προγραμματιστή, τον Yusuke Yamamoto και η άδεια χρήσης του είναι Apache License[7]. Τα κύρια χαρακτηριστικά της είναι:

- Πλήρης υποστήριξη σε Java.
- Δεν χρειάζεται άλλες βιβλιοθήκες για να τη χρησιμοποιήσουμε.
- Έχει ενσωματωμένη υποστήριξη για το OAuth, που χρησιμοποιείται για την πιστοποίηση των συνδέσεων με το Twitter API.
- Είναι συμβατή με την καινούργια έκδοση 1.1 του Twitter API

3.4 Περιορισμοί του Twitter Streaming API

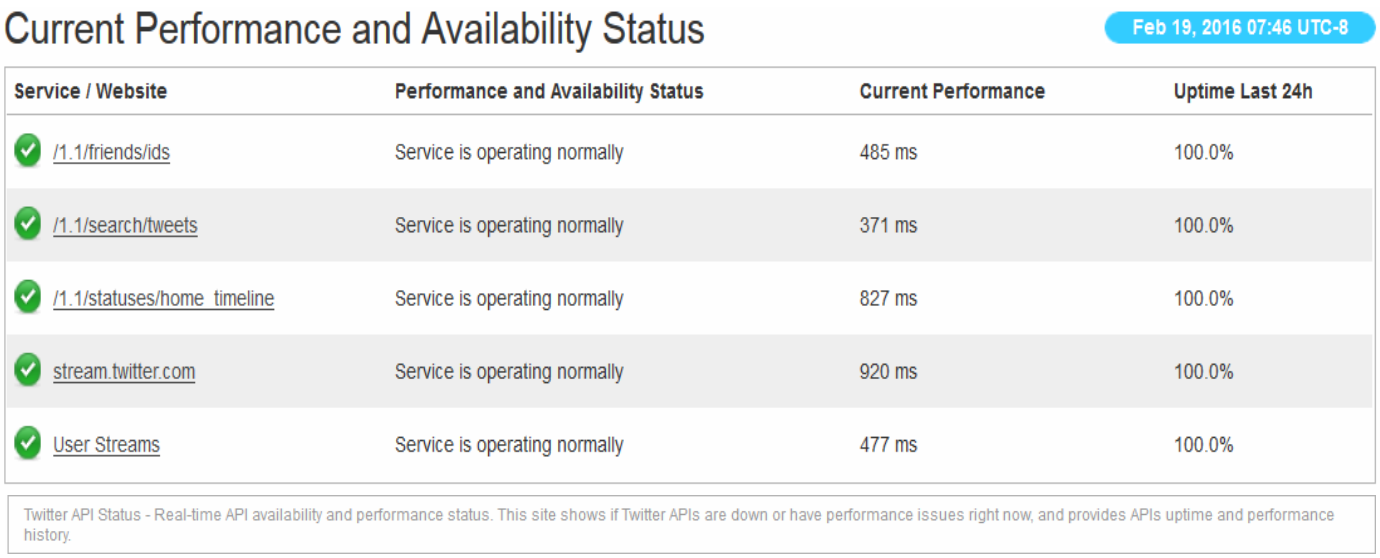
Το Twitter API μπορεί να προσφέρει πάρα πολλές δυνατότητες στον χρήστη για να αξιοποιήσει ώστε να φτιάξει την εφαρμογή του. Όμως στα εργαλεία που προσφέρει, υπάρχουν και αρκετοί περιορισμοί όπου δεν αφήνουν το χρήστη να το αξιοποιήσει στο έπακρο. Με τους περιορισμούς αυτούς ήρθαμε αντιμέτωποι και στην δικιά μας εργασία και έθεσαν σημαντικά όρια στην λειτουργικότητα της εφαρμογής μας.

Οι περιορισμοί αυτοί είναι:

- Υπάρχει ένα όριο στον αριθμό των tweets που λαμβάνουμε μέσω του API σε σχέση με τον συνολικό όγκο tweets που δημοσιεύονται στο Twitter. Αυτή την στιγμή το όριο αυτό είναι στο 1% των συνολικών tweets. Αυτό μας εμποδίζει να έχουμε μία πλήρη εικόνα των πληροφοριών που υπάρχουν στο Twitter καθώς επίσης περιορίζει σημαντικά και τον αριθμό των tweets που λαμβάνουμε.
- Υπάρχει επίσης όριο στα ορίσματα που μπορούμε να βάλουμε στα Request Parameters, με σκοπό να φιλτράρουμε τα tweets που θέλουμε να παίρνουμε. Συγκεκριμένα, μπορούμε να έχουμε μέχρι 400 keywords στην παράμετρο track, 25 γεωγραφικά κουτιά για την παράμετρο location και μέχρι 5,000 UserID για την παράμετρο follow.
- Μπορούμε να έχουμε μόνο μία σύνδεση στο Twitter API ανοιχτή κάθε φορά. Κάθε καινούργια σύνδεση που κάνουμε μέσω της δηλωμένης εφαρμογής μας, θα τερματίζει την παλιά και μετά θα συνδέεται. Το ίδιο αποτέλεσμα θα έχουμε και αν προσπαθήσουμε να συνδεθούμε από δύο διαφορετικές εφαρμογές μέσω της ίδιας IP, όπου έτσι κινδυνεύουμε να μας αποκλείσουν την IP.

Για εφαρμογές που θέλουν να ξεπεράσουν τους περιορισμούς αυτούς, το Twitter προσφέρει το Firehose. Το Twitter Firehose παρέχει πρόσβαση στο 100% των δημόσιων tweets για κάποια τιμή. Τα δεδομένα από το Firehose μπορούν να αγοραστούν από τρίτες εταιρίες που συνεργάζονται με το Twitter. Αυτή τη στιγμή, η μεγαλύτερη εταιρία που πουλάει δεδομένα του Twitter είναι η GNIP.

Η GNIP, που έχει πρόσφατα αγοραστεί από το Twitter, προσφέρει πλήρη πρόσβαση τόσο σε παλαιότερα tweets όσο και στον συνολικό αριθμό των streaming δεδομένων. Επίσης, προσφέρει σημαντικά μεγαλύτερο αριθμό ορισμάτων που εισάγουμε στα Request parameters, γεγονός που αναβαθμίζει σημαντικά την λειτουργικότητα μίας εφαρμογής.



Current Performance and Availability Status

Feb 19, 2016 07:46 UTC-8

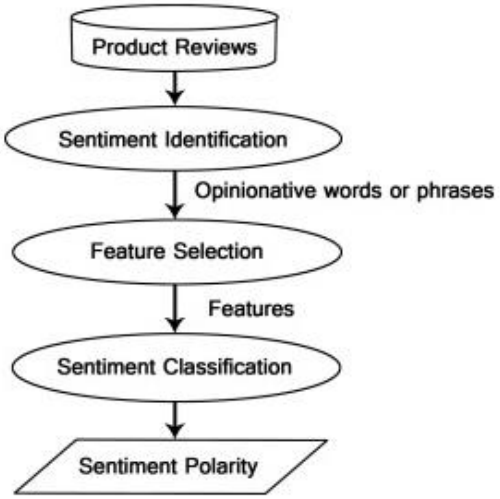
Service / Website	Performance and Availability Status	Current Performance	Uptime Last 24h
/1.1/friends/ids	Service is operating normally	485 ms	100.0%
/1.1/search/tweets	Service is operating normally	371 ms	100.0%
/1.1/statuses/home_timeline	Service is operating normally	827 ms	100.0%
stream.twitter.com	Service is operating normally	920 ms	100.0%
User Streams	Service is operating normally	477 ms	100.0%

Twitter API Status - Real-time API availability and performance status. This site shows if Twitter APIs are down or have performance issues right now, and provides APIs uptime and performance history.

Εικόνα 3.3 Πίνακας με την διαθεσιμότητα των διαφορετικών Υπηρεσιών του Twitter API [5]

4. Ανάλυση Συναισθήματος (Sentiment Analysis)

Η ανάλυση συναισθήματος (Sentiment Analysis) ή αλλιώς εξόρυξη γνώμης (Opinion Mining) είναι μία υπολογιστική μελέτη για την γνώμη, συμπεριφορά ή συναίσθημα ενός ατόμου απέναντι σε μία οντότητα. Μία οντότητα μπορεί να είναι κάποιος άνθρωπος, ένα γεγονός, μία ταινία, κτλ. Αυτά εκφράζονται μέσω κάποιας κριτικής, ενός σχολίου και γενικότερα μέσω κάποιου είδους κειμένου. Γι' αυτό το λόγω χρησιμοποιείται το Natural Language Processing (NLP). Ενώ μπορεί η έννοια του Sentiment Analysis να ταυτίζεται με το Opinion Mining, έχουν στην πραγματικότητα διαφορές. Το Opinion Mining αναλύει τι γνώμη των ανθρώπων για π.χ. ένα προϊόν ενώ το Sentiment Analysis αναγνωρίζει το συναίσθημα που εκφράζεται σε μία γνώμη και το αναλύει. Έτσι, ο στόχος του SA είναι να αναγνωρίζει το συναίσθημα σε ένα κείμενο που εκφράζει γνώμη και να το κατηγοριοποιεί ανάλογα με την πολικότητα του (π.χ. θετικό ή αρνητικό) (Εικόνα 4.1).



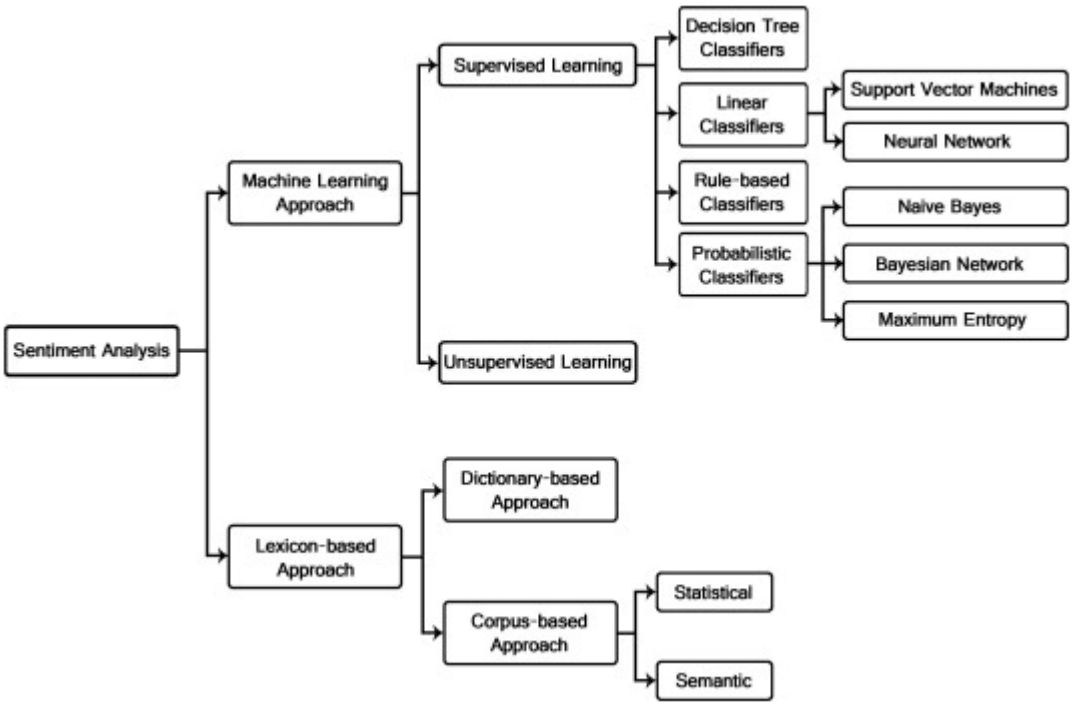
Εικόνα 4.1 Γενικό παράδειγμα SA σε product reviews [13]

Το Sentiment Analysis κάνει κατηγοριοποίηση σε 3 επίπεδα: σε επίπεδο κειμένου (document-level), σε επίπεδο πρότασης (sentence-level) και σε επίπεδο πτυχής (aspect level). Στο document-level, ένα ολόκληρο κείμενο θεωρείται ως η βασική πληροφοριακή μονάδα και επιχειρείται να κατηγοριοποιηθεί ως κείμενο με θετική ή αρνητική γνώμη ή συναίσθημα. Στο sentence-level, αναλύεται η κάθε πρόταση με σκοπό να αναγνωρίσουμε τι συναίσθημα εκφράζει. Αυτό γίνεται σε δύο επίπεδα. Πρώτα, γίνεται αναγνώριση για το αν η πρόταση είναι αντικειμενική ή υποκειμενική. Στην περίπτωση που εκφέρει άποψη, αντικειμενική, τότε θα αποφασιστεί αν έχει θετικό ή αρνητικό συναίσθημα. Στο τρίτο, στο aspect-level, ο στόχος είναι να κατηγοριοποιηθεί το συναίσθημα σε σχέση με τις συγκεκριμένες πτυχές των διαφορετικών οντοτήτων που υπάρχουν μέσα στην πρόταση. Το πρώτο βήμα σε αυτό το επίπεδο είναι να εντοπιστούν οι διάφορες οντότητες μέσα στην πρόταση καθώς και οι πτυχές αυτές.

4.1 Τεχνικές Κατηγοριοποίησης Συναισθήματος

Στο πεδίο του Sentiment Analysis χρησιμοποιούνται διάφορες τεχνικές όπου κάθε μία χρησιμοποιεί και διαφορετικό είδος αλγορίθμων ή συνδυασμό αυτών [16]. Οι δύο μεγάλες κατηγορίες στις οποίες μπορούμε να τις χωρίσουμε είναι οι εξής (Εικόνα 4.2):

- **Τεχνικές βασισμένες σε μηχανική μάθηση (Machine Learning based techniques)**
- **Τεχνικές βασισμένες σε λεξικό (Lexicon based techniques)**



Εικόνα 4.2 Σχήμα με τις κατηγορίες Sentiment Analysis[13]

4.1.1 Machine Learning based techniques

Οι προσεγγίσεις μηχανικής μάθησης όπου είναι εφαρμόσιμες στο Sentiment Analysis ανήκουν στην κατηγορία της εποπτευόμενης (supervised) εκπαίδευσης. Για να εφαρμοστούν οι τεχνικές μηχανικής μάθησης χρειάζονται δύο διαφορετικές κατηγορίες κειμένων: τα κείμενα για εκπαίδευση και τα κείμενα όπου θα γίνουν τα test. Ένας αυτόματος ταξινομητής (classifier) χρησιμοποιεί τα κείμενα εκπαίδευσης για να μάθει τα διαφοροποιητικά χαρακτηριστικά των κειμένων και στη συνέχεια να δοκιμάσει την ακρίβεια και την αποδοτικότητά του στα κείμενα για test. Ένας μεγάλος αριθμός από τεχνικές machine learning έχει χρησιμοποιηθεί για να την κατηγοριοποίηση γνωμών.

Οι αλγόριθμοι που έχουν συναντήσει περισσότερο στην βιβλιογραφία και έχουν χρησιμοποιηθεί περισσότερο για κατηγοριοποίηση γνώμης είναι:

Naïve Bayes (NB): Το μοντέλο κατηγοριοποίησης Naïve Bayes υπολογίζει την a-posteriori πιθανότητα μίας κλάσης, βασισμένο στην κατανομή των λέξεων της κλάσης στο κείμενο. Αυτό το μοντέλο χρησιμοποιεί για το feature extraction την τεχνική Bag of Words (BOW), μη λαμβάνοντας υπ΄ όψιν τη θέση της λέξης στο κείμενο. Χρησιμοποιεί το Θεώρημα Bayes για να υπολογίσει την πιθανότητα ένα δεδομένο σύνολο από feature (feature set) να ανήκει σε ένα συγκεκριμένο label. Θεωρώντας ότι όλα τα features είναι ανεξάρτητα μεταξύ τους, προκύπτει ότι:

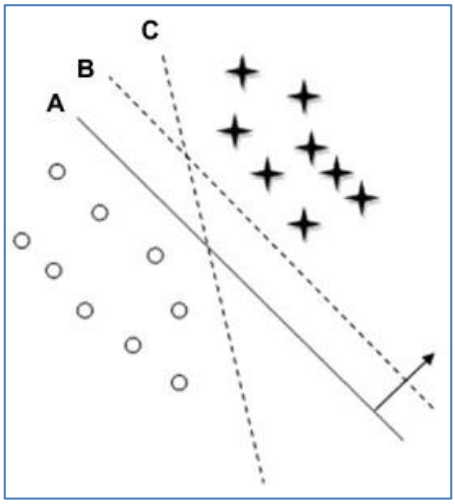
$$P(label \mid features) = \frac{P(label)*P(f1 \mid label)*\ldots..*P(fn \mid label)}{P(features)}$$

όπου P(label) είναι η a priori πιθανότητα για ένα τυχαίο feature να ορίσει ένα label, P(f|label) είναι η a priori πιθανότητα για κάθε σύνολο από feature να κατηγοριοποιηθεί ως label, P(features) είναι η a priori πιθανότητα ότι ένα δεδομένο σύνολο από features εμφανίζεται.

Maximum Entropy (ME): Η μέθοδος αυτή κωδικοποιεί feature sets, που έχουν ήδη κάποιο label, σε διανύσματα. Αυτό το κωδικοποιημένο διάνυσμα χρησιμοποιείται για να υπολογιστεί ένα βάρος για κάθε feature, τα οποία στην συνέχεια συνδυάζονται για να καθοριστεί το πιο πιθανό label για το feature set. Ο classifier παραμετροποιείται από ένα σύνολο από X{weights}, το οποίο χρησιμοποιείται για να συνδυάσει τα κοινά features που παράγονται από ένα feature set με ένα X{encoding}. Η πιθανότητα για κάθε label υπολογίζεται έτσι:

$$P(fs \mid label) = \frac{dotprod(weights\ ,\ encode(fs\ ,\ label))}{sum(dotprod(weights\ ,\ encode(fs\ ,l))\ forlinlabels)}$$

Support Vector Machines (SVM): Η κύρια αρχή της μεθόδου αυτής είναι να εντοπίσεις τους γραμμικούς διαχωριστές (linear separators) στο χώρο αναζήτησης οι οποίοι μπορούνε καλύτερα να χωρίσουν τις διαφορετικές κλάσεις, δηλαδή όλα τα στοιχεία της κάθε κλάσης να έχουν το μεγαλύτερο διάστημα από τα στοιχεία της άλλης κλάσης. Στο παρακάτω γραφικό παράδειγμα έχουμε δύο διαφορετικές κλάσεις και 3 διαφορετικά υπερεπίπεδα (hyperplanes), τα Α, Β, C. Παρατηρούμε ότι το hyperplane Α παρέχει τον καλύτερο διαχωρισμό μεταξύ των κλάσεων καθώς η κανονική απόσταση από κάθε ένα από τα σημεία είναι η μεγαλύτερη και έτσι αντιπροσωπεύει το μεγαλύτερο περιθώριο(Εικόνα 4.3):



Εικόνα 4.3 SVM Linear Separators [13]

Σημαντικό κομμάτι για την supervised τεχνική κατηγοριοποίησης που διαλέγουμε αποτελεί η επιλογή των features. Τα features ουσιαστικά λένε στον classifier πως αναπαρίστανται τα κείμενα. Τα features που χρησιμοποιούνται περισσότερο στην κατηγοριοποίηση συναισθήματος είναι:

Terms presence and frequency: Τα features αυτά περιλαμβάνουν κυρίως uni-grams ή n-grams και τη συχνότητα τους στο κείμενο, όπου n-gram αποτελεί μία ακολουθία από n tokens, που εδώ είναι οι λέξεις. Είτε δίνεται σε κάθε λέξη μία δυαδική τιμή, ανάλογα με το αν εμφανίζεται ή όχι, είτε χρησιμοποιείται η συχνότητα εμφάνισης όρων για την σχετική σημασία τους στο κείμενο. Έχουν χρησιμοποιηθεί ευρέως και με επιτυχία στην κατηγοριοποίηση συναισθήματος.

Part of Speech (POS): Το POS χρησιμοποιείται για την αποσαφήνιση εννοιών που στη συνέχεια χρησιμοποιούνται για την επιλογή των features. Με το POS, κάθε όρος σε κάθε όρο μίας πρότασης θα μπαίνει μία ταμπέλα που δείχνει τη θέση και το ρόλο της στην γραμματική της πρότασης. Για παράδειγμα, μπορούμε να εντοπίζουμε επίθετα και επιρρήματα που συνήθως δείχνουν συναίσθημα.

Negations: Οι αρνήσεις είναι σημαντικό feature που πρέπει να λαμβάνεται υπ' όψιν καθώς μπορεί να αλλάξει το νόημα μιας πρότασης.

Opinion words and phrases: Σε αυτήν την κατηγορία λέξεων ή φράσεων ανήκουν λέξεις ή φράσεις που χρησιμοποιούνται συχνά για να εκφράσουν θετική ή αρνητική γνώμη.

4.1.2 Lexicon based techniques

Στις τεχνικές που χρησιμοποιούνε λεξικό, το classification γίνεται συγκρίνοντας τα features από ένα κείμενο με λέξεις που υπάρχουν στο λεξικό και στις οποίες έχει αντιστοιχηθεί κάποιο συναίσθημα από πριν. Στο λεξικό περιλαμβάνονται λίστες με λέξεις ή φράσεις που χρησιμοποιούνται από τους ανθρώπους για να εκφράσουν συναίσθημα ή γνώμη (opinion words). Η προσέγγιση που ακολουθείται είναι: ορίζουμε λεξικά για θετικές και αρνητικές λέξεις, αναλύουμε το κείμενο και στο τέλος εάν έχει περισσότερες λέξεις από το θετικό λεξικό το ορίζουμε ως θετικό αλλιώς ως αρνητικό. Οι τεχνικές με λεξικό είναι μη-εποπτευόμενη εκπαίδευση (unsupervised learning) καθώς δεν χρειάζεται προηγούμενη εκπαίδευση για να κατηγοριοποιήσεις τα δεδομένα.

Οι κυριότερες μέθοδοι για την σύνταξη των λεξικών είναι οι εξής:

Dictionary based techniques: Σε αυτές τις τεχνικές ξεκινάμε με την συλλογή, με το χέρι, ενός συνόλου από γνωστές λέξεις που εκφράζουν συναίσθημα. Στη συνέχεια ψάχνουμε στο λεξικό WordNet για συνώνυμα και αντώνυμα και τα προσθέτουμε στο λεξικό μας. Αυτή η διαδικασία επαναλαμβάνεται μέχρι να μην υπάρχουν άλλες λέξεις να προστεθούν. Η προσέγγιση αυτή έχει μειονέκτημα στο ότι δεν μπορεί να βρει opinion words για ειδικότερα θέματα.

Corpus based techniques: Αυτές οι τεχνικές βασίζονται σε συντακτικά μοτίβα σε μεγάλα corpora. Μπορούν να παράγουν opinion words με σχετικά μεγάλη ακρίβεια, χρειάζονται όμως ένα πολύ μεγάλο δεδομένων για training. Έχουν πλεονέκτημα ότι μπορούν να παράξουν opinion words για ειδικότερα θέματα, ανάλογα και με τα δεδομένα που παρέχουν για το training.

Γίνεται προφανές ότι οι τεχνικές που χρησιμοποιούνε machine learning μπορούν να παρέχουν πολύ καλύτερα αποτελέσματα από τις τεχνικές με λεξικό. Οι τεχνικές με λεξικό έχουν σαφείς περιορισμούς καθώς λαμβάνουν κυρίως υπ' όψιν κάθε λέξη ξεχωριστά, πράγμα που οδηγεί πολύ εύκολα σε αστοχίες. Οι τεχνικές machine learning, μπορεί να απαιτούν μεγάλο όγκο δεδομένων για να εκπαιδευτούν, όμως εστιάζουν πέρα από κάθε λέξη και στη σύνδεση με τις υπόλοιπες λέξεις για να κάνουν την κατηγοριοποίηση, γεγονός που προσεγγίζει περισσότερο την ανθρώπινη κρίση. Επιπλέον στην εργασία μας, όπου το κείμενο όπου αναλύεται είναι από το Twitter, που η γλώσσα και ο τρόπος που την χειριζόμαστε είναι πολύ διαφορετικός, μία τεχνική με λεξικό δεν θα είχε τα επιθυμητά αποτελέσματα. Έτσι, εστιάσαμε σε τεχνικές machine learning για να κάνουμε το Sentiment Analysis πάνω στα δεδομένα που λαμβάνουμε από το Twitter API. Στην *Εικόνα 4.4* παραθέτουμε έναν πίνακα με τα καλύτερα ποσοστά απόδοσης που έχουμε εντοπίσει στην βιβλιογραφία για Sentiment classification.

Authors	Classifier	Features	Domain	Accuracy
Turney	Pointwise Mutual Information	bigrams	movies, cars, banks	66-84%
Pang & Lee	NB, ME, SVMs	unigrams, bigrams, feature presence or frequence, negation	IMDb[1]	82.9%
Pang & Lee	NB, SVMs	sentence level subjectivity based on minimum cuts	IMDb	87.2%
Mullen & Collier	Hybrid SVM	Turney values, Osgood values, lemma models	IMDb[1], record reviews	87%
Bai et al.	two-stage Markov Blanket Classifier	dependence among words, minimal vocabulary	IMDb[1]	87.5%
Whitelaw et al.	SMO with linear kernel	appraisal groups	movie reviews	90.2%
Kennedy & Inkpen	SVMs, term counting, combination	term frequencies	IMDb	86.2%
Annett & Kondrak	WordNet and SVM	num of pos/neg adj/adv, min distance from pivot words in WordNet	movie reviews, blog posts	65.4-77.5%
Zhou & Chaovalit	ontology-supported polarity mining	n-grams, words, word senses	movie reviews	72.2%
Abbasi et al.	Genetic Algorithm(GA), Information Gain(IG), GA+IG	stylistic and syntactic features	movie reviews, web forum postings	91.7%

Εικόνα 4.4 Μερικά από τα καλύτερα ποσοστά απόδοσης που έχουμε εντοπίσει στην βιβλιογραφία για sentiment classification[15]

4.2 NLP Tools

Για να μπορούμε να εφαρμόσουμε το Sentiment Analysis στα tweets που συλλέγουμε, μπορούμε να χρησιμοποιήσουμε ένα NLP toolkit.

Τα Natural Language Processing toolkits προσφέρουνε επιλογές για όλο το φάσμα των πεδίων που περιλαμβάνει το NLP και επομένως περιλαμβάνουν και τα εργαλεία εκείνα που μας επιτρέπου να κάνουμε Sentiment Analysis. Επίσης πολλά συνδυάζουν το NLP μαζί με machine learning, εστιάζοντας ιδιαίτερα στο κομμάτι αυτό. Για την εργασία μας, θα πρέπει να βρούμε κάποια εργαλεία τα οποία να μπορούμε να τα ενοποιήσουμε με το project μας και όχι κάποια τα οποία δουλεύουν μόνα τους χωρίς να μπορούμε να τα χρησιμοποιήσουμε μέσα στο project μας. Κάνοντας μια έρευνα στο internet, τα εργαλεία όπου χρησιμοποιούνται περισσότερο για το Sentiment Analysis και μπορούν να τρέξουν σε Java είναι:

StanfordNLP: Έχει αναπτυχθεί από το NLP Group στο πανεπιστήμιο του Stanford και περιλαμβάνει ένα μεγάλο εύρος από εργαλεία ανάλυσης γλώσσας, όπως Part-Of-Speech tagging (POS), Named Entity Recognition (NER) καθώς και Sentiment Analysis (SA). Για τα περισσότερα από αυτά υποστηρίζει πολλές γλώσσες όμως για το SA υποστηρίζει μόνο Αγγλικά. Για το Sentiment Analysis χχρησιμοποιείται ένα καινούργιο είδος από Recursive Neural Network, όπου έχουν αναπτύξει και έχει εκπαιδευτεί σε ένα συγκεκριμένο dataset. Η άδεια χρήσης του είναι GNU GPL v3 [18].

Lingpipe: Έχει αναπτυχθεί από την εταιρία Alias-i και αποτελεί μία πλατφόρμα για διάφορες εργασίες πάνω στην ανάλυση γλώσσας. Και αυτό παρέχει εργαλεία για όλα τα είδη NLP και υποστηρίζει αρκετές γλώσσες ενώ μπορούμε να του παρέχουμε συνεχώς νέα δεδομένα για εκμάθηση. Για το Sentiment Analysis, χρησιμοποιεί language model classifiers για την κατηγοριοποίηση των κειμένων. Για το feature selection χρησιμοποιούνται κυρίως τα n-gram. Η άδεια χρήσης του είναι GNU AGPL v3 [17].

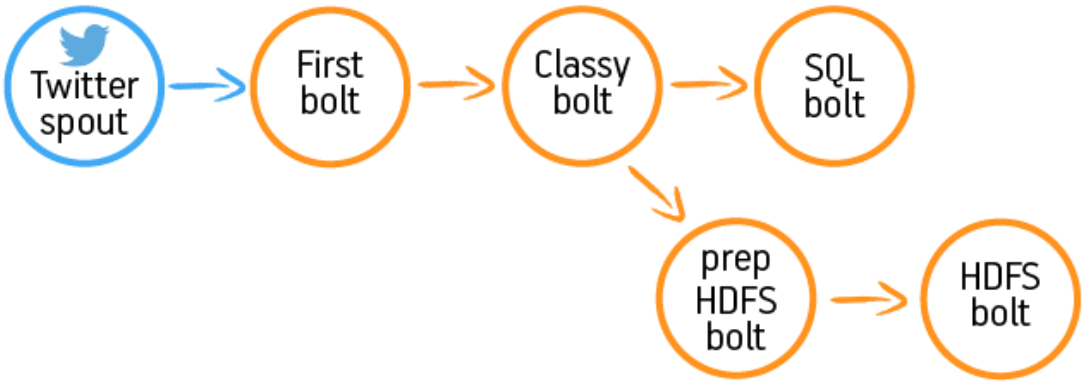
Ανάμεσα στα δύο αυτά εργαλεία, για να χρησιμοποιήσουμε στην εργασία αυτή διαλέξαμε το Lingpipe. Οι κύριοι λόγοι για αυτή την επιλογή είναι:

- Παρέχονται αρκετά tutorials από τους δημιουργούς του, οπότε είναι πιο εύκολο στην εκμάθηση του.
- Η κοινότητα που το χρησιμοποιεί είναι μεγάλη, οπότε μπορούμε να βοηθηθούμε εύκολα σε περίπτωση κάποιου προβλήματος.
- Μπορούμε να το εκπαιδεύσουμε με νέα δεδομένα συνέχεια. Ειδικά στην περίπτωση της εργασίας μας, όπου τα δεδομένα αφορούν ένα ειδικό θέμα, είναι πολύ χρήσιμο για να αυξάνουμε συνέχεια την επιτυχία στο Sentiment Analysis.

5. Τοπολογία

5.1 Γενική Περιγραφή της Τοπολογίας

Η τοπολογία που δημιουργήσαμε για το Sentiment Analysis των εισερχόμενων tweets, αποτελείται, σε μία αφαιρετική περιγραφή, από ένα Spout και 5 Bolt.



Εικόνα 5.1 Γενική Μορφή της Τοπολογίας

Το Spout, με την ονομασία TwitterSpout, μαζεύει συνεχώς τα tweets από το Twitter API και τα στέλνει για επεξεργασία στο επόμενο Bolt. Σημειώνεται ότι τα tweets που παίρνουμε, τα φιλτράρουμε με μία λίστα από παρενέργειες οπότε κρατάμε μόνο αυτά που θέλουμε. Μαζί με το κείμενο του κάθε tweet, στέλνουμε ως δεδομένα την ώρα που το συλλέξαμε, το μοναδικό URL του κάθε tweet και ένα μοναδικό ID που το προσθέτουμε εμείς.

Το Bolt όπου πηγαίνουν τα tuples από το Spout, είναι το Bolt "FirstBolt". Στο FirstBolt γίνεται η πρώτη επεξεργασία των tweets που έχουμε συλλέξει και ελέγχουμε αν περιέχουν κάποιο όνομα από φάρμακο στο κείμενο τους. Τα tweets στα οποία αναφέρεται κάποιο φάρμακο, τα αποστέλλουμε στο επόμενο Bolt. Μαζί με τα πεδία που έχουμε λάβει από το Spout, προσθέτουμε το όνομα του φαρμάκου που εντοπίσαμε καθώς και το όνομα από την παρενέργεια. Αυτά τα πεδία αποστέλλονται στο επόμενο Bolt.

Το επόμενο Bolt είναι το "ClassyBolt", στο οποίο γίνεται το Sentiment Analysis πάνω στα tweets που λαμβάνει από το "FirstBolt". Έτσι κάθε tweet κατηγοριοποιείται ως Θετικό, Αρνητικό ή Ουδέτερο. Η κατηγοριοποίηση του κάθε tweet, μαζί με τα υπόλοιπα πεδία που έχουμε λάβει από το προηγούμενο Bolt αποστέλνονται σε δύο Bolt. Το "SQLBolt" και το "PrepHDFSBolt".

Στο "SQLBolt" λαμβάνουμε όλα τα πεδία από το "ClassyBolt" και τα ανεβάζουμε σε μία βάση δεδομένων, για να μπορούμε να ανατρέχουμε σε αυτά ανά πάσα στιγμή.

Παράλληλα με τη Βάση Δεδομένων, εγγράφουμε τα δεδομένα μας και σε ένα αρχείο στο HDFS. Οι λόγοι για την αποθήκευση των δεδομένων και σε αρχείο είναι: αφενός για την ύπαρξη backup σε περίπτωση σφάλματος στην λειτουργία της Βάσης και

επίσης μπορούμε να διαχειριστούμε τα δεδομένα καλύτερα, όταν είναι σε μορφή αρχείου, όταν τα κατανείμουμε ως training data για την περαιτέρω εκπαίδευση του classifier. Στο "PrepHDFSBolt", δεν κάνουμε κάτι άλλο πέρα από το να βάλουμε όλα τα πεδία που λαμβάνουμε από το "ClassyBolt" σε ένα ενιαίο String και αυτό το στέλνουμε στο "HDFSBolt" το οποίο εγγράφει στο αρχείο εξόδου το String αυτό.

Για να μπορούμε να χρησιμοποιήσουμε στην τοπολογία μας το Twitter API, χρειάστηκε να δημιουργήσουμε και να δηλώσουμε μία εφαρμογή στο αντίστοιχο site του Twitter Developers. Έτσι, έχουμε στη διάθεση μας 4 "κλειδιά" τα οποία θα τα χρησιμοποιήσουμε για την σύνδεση της εφαρμογής μας στο API.

5.2 Αναλυτική περιγραφή των στοιχείων της τοπολογίας

5.2.1 Η μέθοδος Main

Στην κλάση Main υλοποιούμε όλα τα αρχικά στάδια που χρειάζεται η τοπολογία μας για να μπορέσουμε να την εκτελέσουμε. Εκτός από τον ακριβή ορισμό των Spout και Bolt, των αριθμό των task για κάθε από αυτά και την ομαδοποίηση ροών, όπου είναι απαραίτητα για το Storm, υλοποιούμε και τα αρχικά στάδια για τα υπόλοιπα εργαλεία που χρησιμοποιούμε στην εργασία μας.

Ρυθμίσεις HDFS για το διάβασμα αρχείου

Για να μπορέσουμε να διαβάσουμε τα αρχεία που θα χρειαστούμε όταν η τοπολογία μας ανέβει στον cluster, πρέπει να παρέχουμε όλες τις απαραίτητες ρυθμίσεις για το HDFS ώστε να έχει η τοπολογία μας πρόσβαση σε αυτό. Έτσι, για κάθε αρχείο που θέλουμε να διαβάσουμε, πρέπει να παρέχουμε το ακριβές Path του αρχείου μας στο σύστημα HDFS, που έχουμε εκ των προτέρων εισάγει καθώς και το Path του Master του HDFS που υπάρχει στον cluster του εργαστηρίου SoftNet.

Ορισμός φίλτρων για το Twitter API

Στην Main ορίζουμε και τις παραμέτρους του φίλτρου που θα περάσουμε στο Spout για να φιλτράρουμε τα εισερχόμενα tweets.

Track Parameter: Ορίζουμε την παράμετρο του φίλτρο με τους όρους που θέλουμε να υπάρχουν σε κάθε tweet. Στην εργασία μας οι όροι αυτοί είναι μία λίστα με παρενέργειες. Το φίλτρο αυτό το υλοποιούμε ως εξής: διαβάζουμε το αρχείο όπου έχουμε καταχωρημένες ανά σειρά τις παρενέργειες και μετατρέπουμε τα περιεχόμενα του αρχείου σε έναν πίνακα από String. Τον πίνακα αυτόν τον ορίζουμε ως την παράμετρο με όρους στο φίλτρο. Αυτό γίνεται χρησιμοποιώντας τις ρυθμίσεις για το HDFS που περιγράψαμε παραπάνω.

Language Parameter: Ορίζουμε την επιθυμητή γλώσσα που πρέπει να είναι τα tweets. Στην εργασία μας, ορίζουμε ότι τα tweets που θα παίρνουμε θα είναι μόνο στα Αγγλικά.

Ρύθμιση Bolt για δημιουργία αρχείου στο HDFS

Για τη δημιουργία αρχείου στο σύστημα HDFS, έχει δημιουργηθεί ένα component από χρήστη του Storm με το οποίο μπορούμε πολύ εύκολα να δημιουργήσουμε ένα Bolt για αυτή τη δουλειά [19]. Το μόνο που πρέπει να παρέχουμε, είναι απλά κάποιες παράμετροι, οι οποίες είναι:

Όνομα αρχείου: παρέχουμε ουσιαστικά το πρόθεμα του αρχείου όπου θα δημιουργήσουμε, καθώς το πλήρες όνομα του αρχείου είναι της παρακάτω μορφής:

```
{prefix}{componentId}-{taskId}-{rotationNum}-{timestamp}{extension}
```

με το προεπιλεγμένο extension να είναι το .txt

Path Αρχείου: ορίζουμε το ακριβές Path στο οποίο θα δημιουργηθεί το αρχείο στο HDFS.

Format Αρχείου: ορίζουμε τον τρόπο με τον οποίο θα χωρίζονται τα καινούργια στοιχεία που έρχονται κάθε φορά στο Bolt. Επιλέξαμε κάθε καινούργιο στοιχείο να εγγράφεται σε καινούργια γραμμή.

Πολιτική Rotation: Μπορούμε να ελέγξουμε πότε θα γίνονται rotate τα αρχεία. Το αφήσαμε στην default τιμή του.

Τέλος, πρέπει να βάλουμε ως όρισμα το ακριβές Path του Master του HDFS όπου έχουμε εγκαταστήσει στον cluster του εργαστηρίου SoftNet.

Classifier training

Εκπαιδεύουμε τον classifier που μας παρέχει το εργαλείο LingPipe για να μπορεί να γίνει το Sentiment Analysis στα tweets. Πρώτα ορίζουμε τον αριθμό από τις διαφορετικές κατηγορίες συναισθημάτων που μπορεί να εκφράζονται σε ένα tweet. Ορίσαμε αυτές να είναι 3: τα θετικά tweets, τα αρνητικά tweets και τα ουδέτερα. Στην συνέχεια, ορίζουμε τα αρχεία για την εκπαίδευση. Για κάθε διαφορετική κατηγορία έχουμε ένα αρχείο που έχει tweets της αντίστοιχης κατηγορίας. Τα tweets σε κάθε αρχείο και για κάθε κατηγορία, τα έχουμε τοποθετήσει με βάση την δική μας κρίση. Ορίζουμε επίσης και το επίπεδο στο οποίο θα ορίσουμε τα n-gram. Στα tutorials που είναι διαθέσιμα, αναφέρεται ένα εύρος τιμών από 7 μέχρι 12. Επιλέξαμε το χαμηλότερο, λόγω και των μικρό αριθμό λέξεων που υπάρχει σε κάθε tweet. Αφού διαβαστούν τα αρχεία, με τον τρόπο που έχουμε αναλύσει και παραπάνω, δημιουργείται το μοντέλο για τον classifier. Στο, τέλος μετατρέπουμε το μοντέλο μας σε αρχείο για να μπορέσει να διαβαστεί από το αντίστοιχο Bolt για να γίνει το Sentiment Analysis, μέσω του HDFS.

Ορισμός της τοπολογίας

Ορίζουμε τον πλήρη χάρτη της τοπολογίας μας. Έτσι έχουμε:

Στο TwitterSpout περνάμε ως ορίσματα τα 4 "κλειδιά" ασφαλείας που έχουμε διαθέσιμα για την εφαρμογή μας που θα συνδέεται με το Twitter API. Επίσης περνάμε και ως όρισμα το φίλτρο που ορίσαμε παραπάνω για να λαμβάνουμε τα tweet που θέλουμε. Το Spout θα τρέχει σε 1 task, καθώς παραπάνω task θα προκαλεί συνεχείς διακοπές στη σύνδεση με το Twitter API. Στη συνέχεια, ενώνουμε όλα τα υπόλοιπα Bolt μεταξύ τους, σύμφωνα με την περιγραφή που έχουμε δώσει παραπάνω.

Για την ομαδοποίηση των ροών μεταξύ των Bolt, επιλέξαμε την τυχαία ομαδοποίηση (shuffle grouping). Επιλέξαμε την ομαδοποίηση αυτή καθώς κάθε διαφορετικό tuple που αποστέλνεται μεταξύ των Bolt, περιέχει πληροφορία για διαφορετικό tweet. Έτσι, δεν υπάρχει κάποια σχέση που να ενώνει τα διαφορετικά tuples και δε χρειάζεται να υλοποιήσουμε κάποια άλλη ομαδοποίηση.

Τέλος, ορίζουμε και τον αριθμό των workers που θα καταλάβουμε στον Storm cluster του εργαστηρίου. Για τις ανάγκες της τοπολογίας μας, ορίσαμε 2 workers.

5.2.2 TwitterSpout

Όπως έχουμε αναφέρει, στο Spout μας συνδεόμαστε με το Streaming API του Twitter για να έχουμε μια συνεχόμενη ροή από δημόσια tweets.

Αρχικά, για να μπορέσουμε να υλοποιήσουμε τη σύνδεση πρέπει να την πιστοποιήσουμε. Γι' αυτό το λόγο, παίρνουμε τα ορίσματα που έχει περάσει η Main, που είναι τα 4 κλειδιά και χρησιμοποιώντας την αντίστοιχη συνάρτηση που προσφέρει το Twitter4j, πιστοποιούμε την σύνδεση μας. Αφού γίνει αυτό, υλοποιούμε την συνάρτηση της βιβλιοθήκης όπου δημιουργείται η σύνδεση μας και μπορούμε να λαμβάνουμε ροές από tweets. Για να λαμβάνουμε τα tweet που θέλουμε, χρησιμοποιούμε την αντίστοιχη συνάρτηση που παίρνει ως όρισμα το φίλτρο που έχουμε περάσει, επίσης από την Main, και έτσι δεν λαμβάνουμε τυχαία tweets αλλά αυτά που έχουμε ορίσει. Κάθε tweet που λαμβάνουμε το βάζουμε κατευθείαν σε μία LinkedBlockingQueue και κάθε φορά που καλείται η συνάρτηση nextTuple(), τραβάει τα δεδομένα από εδώ.

Το κάθε tweet που αποθηκεύεται στη LinkedBlockingQueue είναι της μορφής Status. Κάθε Status δεν περιλαμβάνει μόνο το περιεχόμενο ενός Tweet αλλά και άλλες πληροφορίες γι' αυτό όπως ποιος το δημιούργησε, εάν έχει γίνει Retweet, το μοναδικό ID κάθε tweet και διάφορα άλλα. Κάθε τέτοιο χαρακτηριστικό μπορούμε να το πάρουμε, χρησιμοποιώντας τις αντίστοιχες συναρτήσεις.

Κάθε tuple που στέλνουμε από το Spout στο FirstBolt περιλαμβάνει τα εξής πεδία:

ID: Σε κάθε tweet που λαμβάνουμε του προσθέτουμε και ένα μοναδικό ID για να ξέρουμε πόσα Status έχουμε λάβει και σε ποια σειρά έχουμε λάβει κάθε Status. Κάθε φορά που τρέχουμε την τοπολογία από την αρχή, η αρίθμηση για τα ID ξεκινάει από την αρχή.

Date: Στέλνουμε την ημερομηνία και ώρα την οποία λάβαμε το tweet.

Tweet: Το περιεχόμενο του Tweet.

URL: Το ακριβές URL περιλαμβάνει δύο διαφορετικά κομμάτια. Ένα αριθμό που δείχνει τον χρήστη ο οποίος έκανε το post και επίσης έναν αριθμό που είναι το μοναδικό ID του κάθε tweet, πχ
https://twitter.com/267497187/status/693126977260167168

5.2.3 FirstBolt

Το FirstBolt αποτελεί το πρώτο στάδιο επεξεργασίας των εισερχόμενων tweets από το Spout. Όπως έχουμε αναφέρει, πραγματοποιείται ένα δεύτερο επίπεδο φιλτραρίσματος στο πεδίο με το περιεχόμενο κάθε Tweet. Εκεί ελέγχουμε αν περιλαμβάνει κάποιο φάρμακο και στην περίπτωση αυτή το προωθούμε στο επόμενο Bolt.

Αρχικά, περνάμε τη λίστα με τα φάρμακα μέσα στο Bolt για να μπορούμε να υλοποιήσουμε το φιλτράρισμα που θέλουμε. Αυτό γίνεται στην μέθοδο Prepare(), δηλαδή κατά την αρχικοποίηση του Bolt. Χρησιμοποιούμε την ίδια μεθοδολογία που χρησιμοποιήσαμε και στην μέθοδο Main με το αρχείο με τις παρενέργειες. Έχουμε βάλει ένα αρχείο με φάρμακα στο HDFS και υλοποιώντας τα ίδια βήματα με την Main, μετατρέπουμε το αρχείο αυτό σε έναν πίνακα με String για να μπορεί να χρησιμοποιείται εύκολα στην μέθοδο execute(). Με παρόμοιο τρόπο, μετατρέπουμε και το αρχείο με τις παρενέργειες που χρησιμοποιήσαμε και στην Main σε πίνακα ώστε να μπορούμε να εντοπίσουμε σε κάθε tweet ποια παρενέργεια περιέχει.

Κάθε φορά που εκτελείται η συνάρτηση execute() στο Bolt, επεξεργαζόμαστε το περιεχόμενο του κάθε tweet που μας στέλνει το Spout. Πριν επεξεργαστούμε περαιτέρω το κάθε tweet, ελέγχουμε εάν περιλαμβάνει κάποιο link σε κάποια ιστοσελίδα. Αυτό το κάνουμε διότι παρατηρήσαμε ότι η συντριπτική πλειοψηφία των tweets που έχουν links, είναι διαφημιστικού περιεχομένου και επαναλαμβάνονται συνέχεια με αποτέλεσμα να μαζεύουμε πάρα πολλά tweets που στην ουσία μας είναι άχρηστα. Έτσι, κάθε tweet που έχει link δεν το επεξεργαζόμαστε καθόλου.

Για τα tweets που επεξεργαζόμαστε ακολουθούμε την εξής διαδικασία:

- ***Αφαιρούμε άχρηστους χαρακτήρες:*** Αφαιρούμε από το περιεχόμενο κάθε tweet τους χαρακτήρες που δεν είναι γράμματα. Αυτό γίνεται για να μπορέσουμε στην συνέχεια να ελέγχουμε για το αν υπάρχει κάποιο φάρμακο σε αυτό, γεγονός που θα είναι δύσκολο να γίνει αν δεν τους αφαιρέσουμε. Π.χ. τέτοιοι χαρακτήρες είναι (#, !, @, \n, \t, …)
- ***Χωρίζουμε ανά λέξη:*** Αφού αφαιρέσουμε τους άχρηστους χαρακτήρες, χωρίζουμε το "καθαρό" πλέον περιεχόμενο ανά λέξη για να μπορούμε μετά να συγκρίνουμε.
- ***Ελέγχουμε για ομοιότητα:*** Συγκρίνουμε κάθε λέξη που προκύπτει από τα παραπάνω με όλα τα στοιχεία από τον πίνακα με τα φάρμακα και ελέγχουμε αν υπάρχει κάποιο φάρμακο και σε αυτή την περίπτωση το κρατάμε για να το στείλουμε στο επόμενο Bolt.

Εάν βρούμε κάποιο από τα φάρμακα που έχουμε στη λίστα, τότε ξανά-ελέγχουμε το tweet για να βρούμε την παρενέργεια που υπάρχει στο tweet. Για αυτό χρησιμοποιούμε τον πίνακα με τις παρενέργειες που έχουμε και όταν την εντοπίσουμε την κρατάμε για να την στείλουμε και αυτή στο επόμενο Bolt.

Στα tuple που στέλνουμε από το FirstBolt στο επόμενο Bolt περιέχονται:

- Τα πεδία που λάβαμε από το Spout: Αυτά είναι ***ID, Date, Tweet, URL***
- ***Drug Name***: Το όνομα του φαρμάκου που εντοπίσαμε στο περιεχόμενο του Tweet.
- ***Side Effect:*** Το όνομα της παρενέργειας που εντοπίσαμε στο περιεχόμενο του Tweet.

5.2.4 ClassyBolt

Στο Bolt αυτό πραγματοποιείται το Sentiment Analysis πάνω στα tweets που έρχονται από το FirstBolt.

Κατά την αρχικοποίηση του Bolt, υλοποιούμε τον classifier που θα χρησιμοποιήσουμε για την κατηγοριοποίηση των tweets. Έτσι, διαβάζουμε το μοντέλο που έχουμε δημιουργήσει στην Main και το έχουμε εξάγει ως αρχείο. Για να διαβάσουμε το αρχείο με το μοντέλο που είναι στο HDFS, χρησιμοποιούμε την ίδια διαδικασία που ακολουθήσαμε και στο FirstBolt. Αφού διαβάσουμε το αρχείο-μοντέλο, το εισάγουμε στον classifier και έτσι μπορούμε να τον χρησιμοποιήσουμε.

Σε κάθε tuple που έρχεται στο Bolt, χρησιμοποιούμε μόνο το πεδίο με το περιεχόμενο του Tweet. Δεν πραγματοποιούμε καμία επεξεργασία στο κείμενο, όπως κάναμε στο FirstBolt διότι πολλές φορές χρησιμοποιούνται οι ειδικοί χαρακτήρες για να ενισχύσουν ένα συναίσθημα. Έτσι κάθε φορά, στον classifier περνάμε για κατηγοριοποίηση ακριβώς όπως είναι το περιεχόμενο του tweet. Ο classifier διαβάζει

κάθε φορά το κείμενο, πραγματοποιεί την κατηγοριοποίηση αυτού βάσει του μοντέλου που του έχουμε βάλει και αυτόματα αποφασίζει σε ποια κατηγορία συναισθήματος ανήκει και του βάζει αυτή την ταμπέλα.

Έτσι τα πεδία του tuple που στέλνει το Bolt αυτό στα επόμενα Bolt είναι:

Τα πεδία που λάβαμε από το FirstBolt: Αυτά είναι ***ID, Date, Tweet, URL, Drug Name, Side Effect***

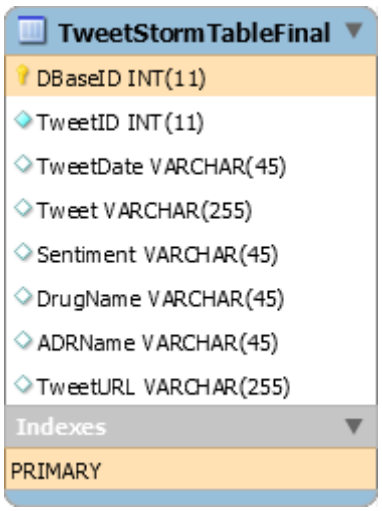
Sentiment: Η κατηγορία συναισθήματος που έβαλε ο classifier το περιεχόμενο του tweet.

5.2.5 SQLBolt

Στο SQLBolt αποθηκεύουμε όλα τα πεδία που έχουν έρθει από το ClassyBolt σε έναν πίνακα σε μία Βάση Δεδομένων για να μπορούμε στη συνέχεια να τα εξετάσουμε. Το Bolt αυτό δε στέλνει tuples σε κάποιο άλλο Bolt.

Η Βάση Δεδομένων στην οποία έχουμε δημιουργήσει τον πίνακα με τα δεδομένα μας είναι στην MySQL, γεγονός που μας επιτρέπει να μπορούμε να τον επεξεργαστούμε με ευκολία, καθώς η MySQL είναι ευρέως διαδεδομένη. Για τη σύνδεση στη Βάση, χρησιμοποιούμε τον driver JDBC και τον αντίστοιχο connector που υπάρχει για την Java.

Τον πίνακα στην Βάση τον έχουμε δημιουργήσει πριν τρέξουμε την τοπολογία μας και το schema του είναι το εξής (Εικόνα 5.2):



Εικόνα 5.2 Ο Πίνακας των Δεδομένων μας

Τα Columns που έχει κάθε σειρά του πίνακά μας στην Βάση Δεδομένων είναι:

DBaseID(INT): Είναι το Primary Key του πίνακα μας και είναι AUTO INCREMENT. Διασφαλίζουμε έτσι ότι κάθε σειρά του πίνακα θα έχει από ένα μοναδικό ID που αναφέρεται στον πίνακα.

TweetID(INT): Είναι το μοναδικό ID κάθε tweet που έχουμε επεξεργαστεί. Δεν μπορεί να αποτελέσει το Primary Key στον Πίνακα διότι κάθε φορά του τρέχουμε την τοπολογία μας από την αρχή, τα ID αναθέτονται στα tweet πάλι από την αρχή. Έτσι, μπορεί να έχουμε κάποια στιγμή δύο διαφορετικά tweet, από διαφορετικές φορές που έχουμε τρέξει την τοπολογία, και να έχουν το ίδιο ID.

TweetDate(VARCHAR): Το πεδίο με την ημερομηνία και ώρα που συλλέξαμε το κάθε tweet.

Tweet(VARCHAR): Το πεδίο με το περιεχόμενο του κάθε tweet.

Sentiment(VARCHAR): Το πεδίο με την κατηγορία συναισθήματος που κατατάξαμε το κάθε tweet.

DrugName(VARCHAR): Το πεδίο με το όνομα του φαρμάκου που εντοπίσαμε σε κάθε tweet.

ADRName(VARCHAR): Το πεδίο με την παρενέργεια που εντοπίσαμε σε κάθε tweet.

TweetURL(VARCHAR): Το πεδίο με το πλήρες URL για κάθε tweet.

Όλα τα παραπάνω πεδία, εκτός από το DBaseID, τα λαμβάνουμε από το ClassyBolt. Για να εισάγουμε στον Πίνακα, χρησιμοποιούμε την μέθοδο **PreparedStatement**. Με αυτόν τον τρόπο, κάθε φορά που εκτελείται η execute, πραγματοποιούνται τα εξής βήματα:

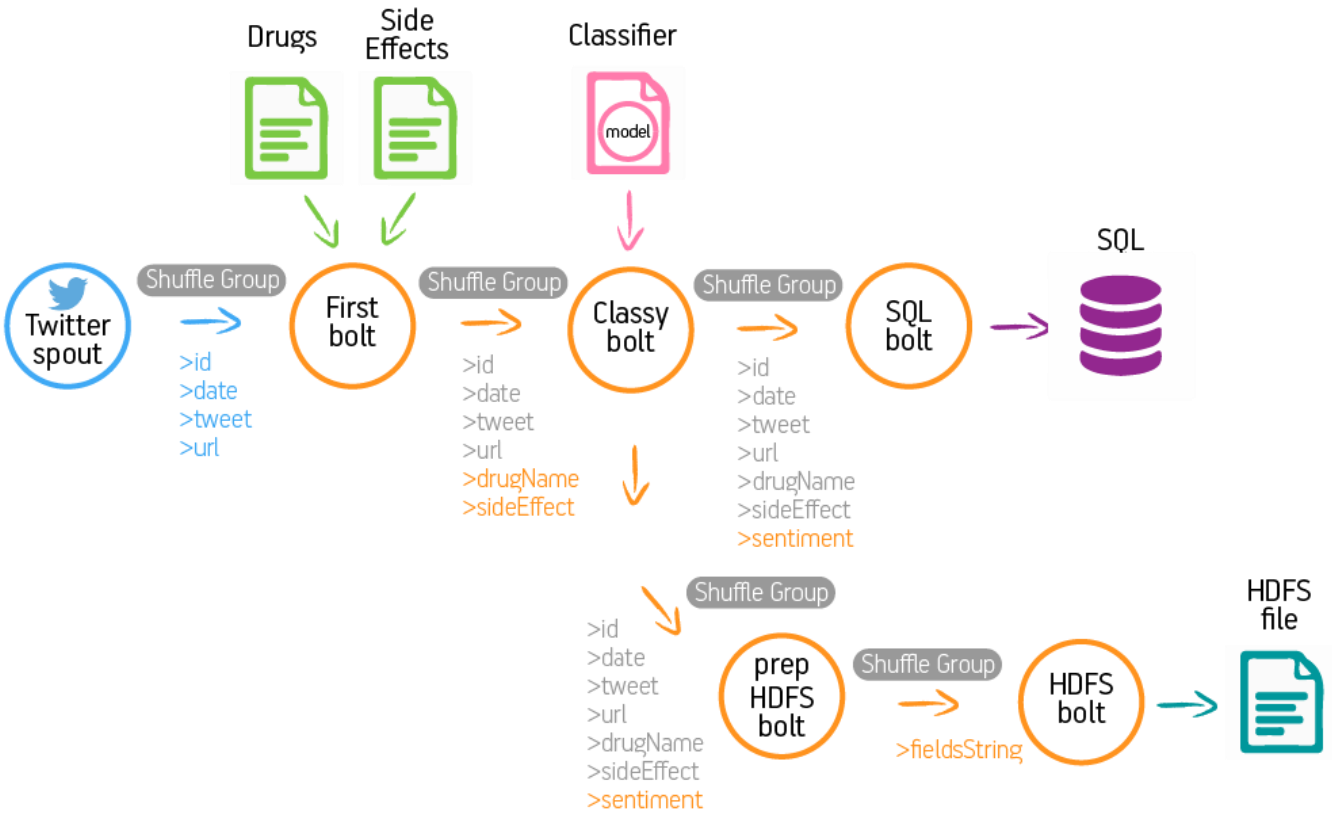
1) Ανοίγουμε ένα connection στην Βάση Δεδομένων
2) Δημιουργούμε ένα στιγμιότυπο από το PreparedStatement.
3) Του περνάμε το query, με το οποίο θα περάσει τα πεδία που έχουμε λάβει στα αντίστοιχα πεδία του Πίνακα.
4) Εκτελούμε το PreparedStatement.
5) Τερματίζουμε το PreparedStatement και την σύνδεση στη Βάση Δεδομένων.

5.2.6 PrepHDFSBolt

Το Bolt αυτό αποτελεί το πρώτο στάδιο για να γράφουμε τα δεδομένα μας σε αρχείο στο HDFS του cluster. Επειδή το component που έχουμε για την δημιουργία του αρχείου εγγράφει κατευθείαν τα tuples που του έρχονται, παρεμβάλλεται το Bolt αυτό για να τοποθετήσει τα πεδία κάθε tuple με έναν τρόπο που να είναι εύκολο να διαβαστούν αφού εγγραφούν στο αρχείο.

Έτσι, το Bolt αυτό λαμβάνει όλα τα πεδία από το ClassyBolt, τα τοποθετεί σε μία σειρά μέσα σε ένα String και στην συνέχεια κάνει emit αυτό το String στο Bolt που τα εγγράφει στο αρχείο. Το String αυτό έχει την εξής μορφή:

TweetID | Date | Tweet | Sentiment | Drug Name | Side_Effect | URL



Εικόνα 5.3 Αναλυτικό σχήμα της τοπολογίας

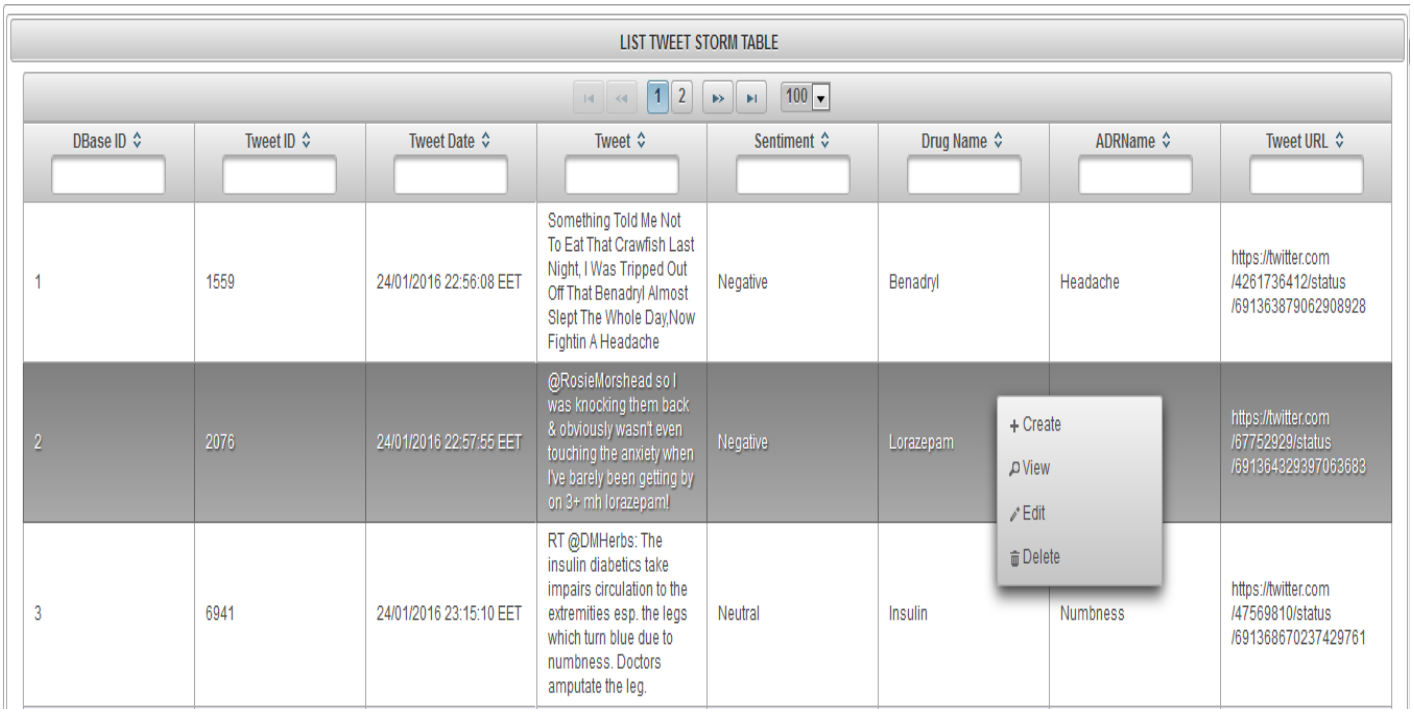
5.3 CRUD Interface

Τα δεδομένα που εισάγουμε στην Βάση Δεδομένων πρέπει να μπορούμε με κάποιο τρόπο να τα βλέπουμε για να μπορούμε να βγάλουμε και τα συμπεράσματα που θέλουμε. Για τον λόγο αυτό, χρειαζόμαστε κάποια πλατφόρμα με την οποία να μπορούμε με εύκολο τρόπο να συνδεόμαστε στη Βάση μας και να βλέπουμε τα περιεχόμενα του Πίνακα που έχουμε δημιουργήσει, είτε συγκεντρωτικά όταν σταματήσει να τρέχει η τοπολογία είτε και ανά πάσα στιγμή ενώ τρέχει. Ένα τέτοιο είδος πλατφόρμας αποτελούν τα CRUD Interfaces.

Ο όρος CRUD προκύπτει από τις λέξεις Create, Read, Update, Delete. Αυτές οι λέξεις παραπέμπουν στις βασικές συναρτήσεις που υλοποιούνται στις εφαρμογές που βασίζονται στο σχεσιακό μοντέλο, όπως παράδειγμα η SQL. Συγκεκριμένα στην SQL, οι λειτουργίες αυτές αντιστοιχούν στις μεθόδους INSERT, SELECT, UPDATE, DELETE. Επίσης, ο όρος CRUD σχετίζεται συχνά και με τις γραφικές διεπαφές που έχουν διάφορες εφαρμογές. Μέσω των διεπαφών αυτών μπορεί ο χρήστης να εφαρμόσει αυτές τις λειτουργίες με ευκολία στην Βάση Δεδομένων που είναι συνδεδεμένη με αυτές. Για την ακόμα πιο εύκολη δημιουργία τέτοιων διεπαφών, υπάρχουν διάφοροι generators με τους οποίους μπορεί ένας χρήστης να δημιουργήσει μία διεπαφή για την Βάση του.

Μετά από σχετική έρευνα στο διαδίκτυο, αποφασίσαμε να χρησιμοποιήσουμε το PrimeFaces CRUD Generator. Το εργαλείο αυτό, έχει υλοποιηθεί από την εταιρία Primefaces και αποτελεί plugin για το IDE NetBeans. Αφού εγκαταστήσαμε το NetBeans στην συνέχεια προσθέσαμε και τον εν λόγω plugin. Το μόνο που χρειάστηκε να ρυθμίσουμε είναι η διεύθυνση και οι κωδικοί για να συνδεθούμε στη Βάση Δεδομένων του SoftNet. Στην συνέχεια, αφού εντοπιστεί ο Πίνακας που έχουμε δημιουργήσει μέσα στη Βάση με αυτοματοποιημένο τρόπο δημιουργείται το Interface με το οποίο μπορούμε να διαχειριστούμε τον Πίνακα μας. Μας δίνεται η δυνατότητα να επιλέξουμε ποιες από τις λειτουργίες θέλουμε να είναι διαθέσιμες για τον χρήστη. Πιο συγκεκριμένα μπορούμε να επιλέξουμε ποια από τις επιλογές CREATE-READ-UPDATE-DELETE θα εμφανίζονται, τη δυνατότητα να ταξινομούνται τα αποτελέσματα με βάση τα πεδία του πίνακα όπως και επιλογές για σελιδοποίηση.

Τέλος, αξίζει να σημειώσουμε ότι το Primefaces CRUD Generator βασίζεται σε τεχνολογίες της Java που υπάρχουν για σχεδιασμό web εφαρμογών, όπως τα JSF και EJB. Έτσι, μπορεί ένας χρήστης να παρέμβει στον κώδικα του Interface, αντίστοιχα με μία εφαρμογή web, και να προσθέσει επιπλέον λειτουργικότητα.



Εικόνα 5.4 Στιγμιότυπο από το CRUD Interface με το οποίο βλέπουμε τα περιεχόμενα της Βάσης

6. Αποτελέσματα - Συμπεράσματα

Για να δοκιμάσουμε να τρέξουμε την τοπολογία μας στον cluster του εργαστηρίου, κάναμε εκτεταμένο debugging στον κώδικά μας. Αυτό έγινε πρώτα στο επίπεδο του υπολογιστή μας, για να ελέγξουμε τα διαφορετικά components που χρησιμοποιούμε, και στην συνέχεια στο επίπεδο του cluster, όπου έπρεπε να ελέγξουμε για τις διαφορετικές ρυθμίσεις που έπρεπε να γίνουν έτσι ώστε να μπορούμε να τρέχουμε χωρίς προβλήματα την τοπολογία.

Τα αποτελέσματα που έχουμε μετά από την εκτέλεση της τοπολογίας μας και τα αντίστοιχα συμπεράσματα, μπορούμε να τα κατατάξουμε σε δύο κατηγορίες. Η μία κατηγορία αφορά τα αποτελέσματα που είχαμε από την χρησιμοποίηση του Twitter API και τα συμπεράσματα που μπορούμε να βγάλουμε από αυτό. Η δεύτερη αφορά τα αποτελέσματα από το Sentiment Analysis που κάναμε πάνω στα δεδομένα μας και τα αντίστοιχα συμπεράσματα.

Για να μπορέσουμε να έχουμε μια καλή εκτίμηση για τα αποτελέσματα, αφήνουμε την τοπολογία μας να τρέξει για αρκετές μέρες στον cluster ώστε να έχουμε έναν ικανοποιητικό αριθμό από αυτά. Κρίναμε ως ικανοποιητικό αριθμό tweets, για να τα εξετάσουμε και να βγάλουμε κάποια αποτελέσματα, έναν αριθμό κοντά στα **1500**. Με βάση αυτά τα δεδομένα θα εξάγουμε τα συμπεράσματά μας.

6.1 Twitter API

Όπως έχουμε αναφέρει ήδη, η δημόσια χρήση του Twitter Streaming API έχει κάποιους περιορισμούς όσον αφορά το ποσοστό δεδομένων στο οποίο έχουμε πρόσβαση.

Σημειώνουμε ότι οι πηγές για να γεμίσουμε τις λίστες με τις παρενέργειες και με τα φάρμακα είναι το site του FDA [8], το drugs.com καθώς και η πλατφόρμα SIDER[9]. Η λίστα με τις παρενέργειες περιλαμβάνει 400 όρους, μιας και τόσο είναι το όριο για το API και η λίστα με τα φάρμακα περιλαμβάνει περίπου 4000 φάρμακα.

6.1.1 Αποτελέσματα από το Twitter API

Ο ρυθμός των tweets που λαμβάνει το Spout και τα στέλνει στο FirstBolt, δηλαδή αυτά που περιέχουν κάποια παρενέργεια είναι, κατά μέσο όρο, περίπου 320.000 ανά ημέρα.

Λέμε κατά μέσο όρο διότι δεν είναι ίδιος ο αριθμός των tweets που λαμβάνουμε κάθε μέρα, οπότε βγάζουμε τον μέσο όρο από τον συνολικό αριθμό των tweets που έχουμε λάβει στο διάστημα κάποιων ημερών.

Από τα 320.000 ανά ημέρα tweet που φτάνουν στο FirstBolt, κρατάμε, πάλι κατά μέσο όρο τα 340. Δηλαδή, τόσα tweet περιέχουν και παρενέργεια και όνομα φαρμάκου. Η αναλογία, όπως φαίνεται, είναι κοντά στο 1/1000. Αντίστοιχα αναφέρουμε και εδώ κατά μέσο όρο.

Παραθέτουμε έναν πίνακα με ενδεικτικά tweets που λαμβάνουμε και αυτά που κρατάμε τελικά για επεξεργασία:

Tweets	Περιέχουν ονομασία φαρμάκου
@LlNZZMORGAN It came so out of nowhere I almost had a stroke	OXI
Chills... #BatmanvSuperman	OXI
Zika virus?? Lassa fever.. Dang whose name is next?	OXI
Already having separation anxiety from @96_Hamilton	OXI
Shit. Do you want me to get ur excedrin migraine and water?	NAI
I've got a fever, and the only thing that can cure it is a gram of Tylenol and rest probably.	NAI
I can't feel my face I'm on adderall, nausea	NAI

Πίνακας 6.1.1.1 Ενδεικτικό δείγμα από διαφορετικά tweets που λαμβάνουμε

Μελετώντας τα tweets που προκύπτουν μετά από το φιλτράρισμα από το FirstBolt, μπορούμε να εντοπίσουμε κάποια επιπλέον στατιστικά στοιχεία.

Οι παρενέργειες οι οποίες αναφέρονται περισσότερο στα tweets των χρηστών είναι:

Ονομασία Παρενέργειας	Ποσοστό Εμφάνισης(%)
Anxiety	26,9
Headache	21,2
Migraine	8,1
Fever	5,3
Nausea	3,4

Πίνακας 6.1.1.2 Ποσοστά Εμφάνισης Παρενεργειών

Τα φάρμακα τα οποία αναφέρονται περισσότερο στα tweet των χρηστών είναι:

Ονομασία Φαρμάκου	Ποσοστό Εμφάνισης(%)
Xanax	14,2
Aspirin	10,7
Ibuprofen	10,4
Tylenol	6,9
Paracetamol	6,1

Πίνακας 6.1.1.3 Ποσοστά Εμφάνισης Φαρμάκων

6.1.2 Συμπεράσματα από το Twitter API

Κάποια πρώτα συμπεράσματα που μπορούμε από τα παραπάνω στοιχεία είναι τα εξής:

- Ο αριθμός των tweets που λαμβάνουμε είναι σχετικά χαμηλός. Αυτό οφείλεται σε μεγάλο ποσοστό στον περιορισμό του Public API του Twitter καθώς επίσης και στον περιορισμό του αριθμού των όρων με τους οποίους φιλτράρουμε τα tweets που λαμβάνουμε.

- Ο αριθμός των tweets που λαμβάνουμε μετά το φιλτράρισμα με τα φάρμακα είναι πολύ μικρός σε σχέση με τον αριθμό που στέλνει το Spout. Αυτό οφείλεται κυρίως σε δύο παράγοντες: αφενός κάποιες παρενέργειες (π.χ. burning, nightmares, …) χρησιμοποιούνται από τους χρήστες και για άλλης φύσης tweets, τα οποία δεν μας ενδιαφέρουν αλλά τα συλλέγουμε. Αφετέρου, επειδή αποκλείουμε tweets που περιέχουν υπερσυνδέσμους, λόγω ότι είναι διαφημιστικά, έναν μεγάλος αριθμός από tweets τα οποία περιέχουν ονομασία από φάρμακο δεν τον κρατάμε.

- Οι περισσότερες παρενέργειες που αναφέρονται στα tweets των χρηστών αφορούν γενικές περιγραφές (π.χ. anxiety, headache, pain) ενώ αντίστοιχα και τα φάρμακα που αναφέρονται είναι πάρα πολύ κοινά (π.χ. Aspirin, Xanax, …). Το γεγονός αυτό είναι αναμενόμενο και μπορεί να μην προσφέρει πολύ σημαντικά ευρήματα στους ερευνητές αποτελεί όμως μία σημαντική παράπλευρη πτυχή για έρευνα.

6.2 Sentiment Analysis

Τα tweets στα οποία κάνουμε Sentiment Analysis είναι αυτά που στέλνονται από το FirstBolt στο ClassyBolt. Σημειώνουμε ότι χρησιμοποιούμε τον classifier όπου έχουμε δημιουργήσει στη Main, μετά την εκπαίδευση από τα δεδομένα όπου του έχουμε εισάγει, και τον διαβάζουμε ως αρχείο στο Bolt.

6.2.1 Αποτελέσματα από Sentiment Analysis

Τον σημαντικότερο ρόλο στην προσέγγιση που χρησιμοποιεί ο classifier, δηλαδή στην machine learning προσέγγιση, έχουν τα δεδομένα που χρησιμοποιούμε για την εκπαίδευση του. Για κάθε κατηγορία συναισθήματος που έχουμε, βάζουμε tweets που έχουμε ήδη συλλέξει με βάση το υποκειμενικό μας κριτήριο. Αυτό σημαίνει ότι ένας άλλος άνθρωπος μπορεί κάποια δεδομένα να τα κατηγοριοποιούσε διαφορετικά. Με αυτό το δεδομένο, παραθέτουμε έναν Πίνακα με ενδεικτικά tweet για κάθε διαφορετική κατηγορία:

Δείγμα από tweet για εκπαίδευση	Κατηγορία Συναισθήματος
Excedrin is a gift from God when you have a headache	POSITIVE
I LOVE this Effexor! MAJORLY kills my libido.	POSITIVE
Finally feeling back to normal after 3 days of a migraine. Thank God for Zofran for helping me make it through the festivities!	POSITIVE
That nap was on point.... Cymbalta did that shit cuz I dont take naps...ever	POSITIVE
Levaquin sucks. Blinding headaches. Vomiting. Diarrhea. Time for my next dose.	NEGATIVE
sometimes it never ends. So sorry. Topomax is miserable with parasthesia, cognitive problems, kidney stones.	NEGATIVE
my mom used Rivaroxaban and it gave her a terrible time with weakness and muscle pain.	NEGATIVE
Zoloft was just as bad as Effexor. It made me more depressed and more suicidal. I don't want it.	NEGATIVE
Viagra was developed with the intention of relieving chest pain and its side effect was accidental.	NEUTRAL
So now botox is being used for migraine headaches?	NEUTRAL
I heard cheese cake was about the same as tramadol for helping with pain....might explain my weight gain ha	NEUTRAL

Πίνακας 6.2.1.1 Ενδεικτικό δείγμα από τα training Data

Κατά την τελευταία φορά όπου τρέξαμε την τοπολογία μας, είχαμε χρησιμοποιήσει για την εκπαίδευση του classifier περίπου 1000 tweets, τα οποία τα είχαμε συλλέξει από προηγούμενες φορές όπου τρέξαμε την τοπολογία μας. Τα tweets αυτά, τα κατανείμαμε στις αντίστοιχες κατηγορίες ως εξής:

- 500 tweets στην κατηγορία Αρνητικά

- 450 tweets στην κατηγορία Ουδέτερα

- 150 tweets στην κατηγορία Θετικά

Έχοντας εκπαιδεύσει τον classifier με αυτό τον αριθμό δεδομένων, τα ποσοστά κάθε κατηγορίας στα tweets που συλλέξαμε κατά την τελευταία φορά όπου τρέξαμε την τοπολογία μας είναι:

Κατηγορία συναισθήματος	Ποσοστό (%)
Positive	9,8
Negative	53,5
Neutral	36,7

Πίνακας 6.2.1.2 Ποσοστά κατηγοριοποίησης του classifier στα tweets

Για να εξετάσουμε την ακρίβεια του classifier, εξετάζουμε μόνοι μας τα tweets που έχουμε λάβει και την κάθε κατηγορία στην οποία έχουν κατηγοριοποιηθεί για να δούμε σε πόσα έχει γίνει σωστή κατηγοριοποίηση και σε ποια όχι. Σημειώνουμε ότι εξετάζουμε μόνο το αν ο classifier μας, έκανε σωστή κατηγοριοποίηση σε κάθε tweet ή όχι. Τα υπόλοιπα στοιχεία που μπορούμε να ελέγξουμε στα γενικότερα πλαίσια του text classification, δεν τα εξετάζουμε στα πλαίσια αυτής της Διπλωματικής Εργασίας.

Το γενικό ποσοστό της ακρίβειας (accuracy) του classifier μετρήθηκε στο 68,4%.

Πιο συγκεκριμένα σε κάθε κατηγορία το αντίστοιχο ποσοστό ακρίβειας είναι:

Κατηγορία	Accuracy (%)
Positive	63,2
Negative	72,4
Neutral	69,6

Πίνακας 6.2.1.3 Ποσοστά ακρίβειας του classifier στις επιμέρους κατηγορίες

6.2.2 Συμπεράσματα από Sentiment Analysis

Κάποια πρώτα συμπεράσματα που μπορούμε να βγάλουμε πάνω στα παραπάνω αποτελέσματα είναι:

- Ένας παράγοντας που οδηγεί στα χαμηλά ποσοστά ακρίβειας του classifier είναι ότι τα δεδομένα που παρέχουμε για την εκπαίδευση του classifier είναι πολύ λίγα. Αυτό διότι επιλέξαμε να παρέχουμε για εκπαίδευση μόνο tweet που έχουμε ήδη συλλέξει και είναι σχετικά με το ειδικό θέμα που εξετάζουμε. Όσο περισσότερο τρέχουμε την τοπολογία μας και μαζεύουμε περισσότερα tweets για την εκπαίδευση τόσο περισσότερο θα αυξάνεται το ποσοστό της ακρίβειας.

- Ένας άλλος παράγοντας για τα χαμηλά ποσοστά ακρίβειας είναι η χρήση της γλώσσας που χρησιμοποιείται στο Twitter. Έχουμε συχνά λάθος χρήση της Αγγλικής γλώσσας, ορθογραφικά λάθη, χρήση ιδιωματισμών ενώ πολλές φορές εκφράζονται ανάμεικτα συναισθήματα. Επίσης πολλές φορές τα μηνύματα είναι πολύ μικρά σε μήκος. Όλοι αυτοί οι παράγοντες κάνουν δύσκολη την κατηγοριοποίηση ενός tweet σε σχέση με ένα κανονικό κομμάτι ενός κειμένου.

- Ο αριθμός των Positive tweet που έχουμε για την εκπαίδευση του classifier είναι πολύ μικρότερος από τις άλλες δύο κατηγορίες. Αυτό το αποδίδουμε στην παρατήρηση ότι οι χρήστες του Twitter χρησιμοποιούν περισσότερο την πλατφόρμα για να εκφράσουν παράπονα για κάτι που τους απασχολεί, στην δικιά μας περίπτωση να διαμαρτυρηθούν για διάφορες παρενέργειες από φάρμακα.

-Λόγω του γεγονότος ότι υπάρχουν περισσότερα δεδομένα για τις κατηγορίες Negative και Neutral, ο classifier πετυχαίνει καλύτερα ποσοστά στο accuracy σε αυτές τις δύο κατηγορίες από τα Positive.

7. Μελλοντικές Επεκτάσεις

Στα πλαίσια αυτής της Διπλωματικής Εργασίας έγινε μία πρώτη προσπάθεια για τη δημιουργία μιας εφαρμογής που θα συλλέγει tweets σε πραγματικό χρόνο και αυτόματα θα τα κατηγοριοποιεί βάσει του συναισθήματος που εκφράζουν. Το περιεχόμενο από τα tweets αποτελείται από διάφορες παρενέργειες που προκαλούν φάρμακα που υπάρχουν στην αγορά.

Στους στόχους που είχαμε για υλοποίηση, κρίνουμε ότι είχαμε επιτυχία. Αυτό προκύπτει καθώς η εφαρμογή μπορεί να τρέξει για όσο διάστημα επιθυμούμε και παράγει τα αποτελέσματα που θέλουμε, έστω και σε αρχικό στάδιο. Παρόλα αυτά, μπορούμε να επεκτείνουμε την λειτουργικότητά της κάνοντας διάφορες αλλαγές.

Ένα πρώτο πεδίο που μπορούμε να επεκτείνουμε είναι να αυξήσουμε την ροή της εισερχόμενης πληροφορίας. Δηλαδή, να μπορέσουμε να αποκτήσουμε πρόσβαση στο 100% των διαθέσιμων tweet που δημοσιεύονται στο Twitter. Το αρνητικό για αυτό είναι ότι απαιτούνται χρήματα για να το επιτύχουμε. Μία σημαντική προσθήκη θα ήταν να αποκτήσουμε πρόσβαση και σε άλλες πηγές πληροφορίας τις οποίες θα μπορούμε να επεξεργαζόμαστε μαζί με τα δεδομένα από το Twitter. Οι πηγές αυτές θα μπορούσαν να αποτελούνε άλλα κοινωνικά δίκτυα, όπως π.χ. το Facebook, ή επίσης άλλες πηγές όπως πλατφόρμες από σχόλια που υπάρχουν σε πάρα πολλά site από τις οποίες μπορούμε να μαζεύουμε δεδομένα, όπως π.χ. το Disqus.

Ένα δεύτερο πεδίο όπου μπορούμε να κάνουμε αλλαγές είναι αυτό του Sentiment Analysis. Έτσι μπορούμε να χρησιμοποιήσουμε άλλες πλατφόρμες όπου προσφέρουν τέτοιες δυνατότητες, όπως π.χ. η πλατφόρμα του Stanford, και να πραγματοποιήσουμε συγκρίσεις στην διαφορά στην απόδοση τους. Επίσης, μπορούμε να εφαρμόσουμε και άλλες τεχνικές NLP πάνω στα δεδομένα που λαμβάνουμε, ώστε να μπορούμε να έχουμε επιπλέον αποτελέσματα για να μελετήσουμε.

Τέλος, μπορούμε να τρέξουμε την εφαρμογή μας για αναζήτηση δεδομένων που ανήκουν σε άλλες κατηγορίες ενδιαφέροντος, πέρα του πεδίου που είχαμε στην εργασία αυτή. Με τα εργαλεία που χρησιμοποιούμε, η συλλογή τέτοιων δεδομένων όσο και η εφαρμογή NLP τεχνικών σε αυτά είναι πολύ εύκολο να γίνει με πολύ μικρές αλλαγές.

8. Βιβλιογραφία

[1] http://storm.apache.org/

[2] S. Allen, M. Jankowski, P. Pathirana. Storm Applied: Strategies for real-time event processing. Greenwich, CT: Manning Publications Co.

[3] A. Jain, A. Nalya. Learning Storm. Packt Publishing.

[4] J.Leiviusky, G. Eisbruch, D. Simnassi. Getting Started with Storm. O'Reilly Media, Inc.

[5] https://dev.twitter.com/overview/api

[6] https://dev.twitter.com/streaming/overview

[7] http://twitter4j.org/en/index.html

[8] http://www.fda.gov/Drugs/default.htm

[9] http://sideeffects.embl.de/

[10] Elizabeth Sukkar. Searching social networks to detect adverse reactions. The Pharmaceutical Journal, 24 January 2015, Vol 294, No 7846

[11] Clark C. Freifeld, John S. Brownstein, Christopher M. Menone, Wenjie Bao, Ross Filice, Taha Kass-Hout, Nabarun Dasgupta. Digital Drug Safety Surveillance: Monitoring Pharmaceutical Products in Twitter. Drug Safety. Volume 37, Issue 5 , pp 343-350

[12] Abeed Sarker, Graciela Gonzalez. Portable automatic text classification for adverse drug reaction detection via multi-corpus training. Journal of Biomedical Informatics, Volume 53, February 2015, Pages 196–207

[13] Walaa Medhat, Ahmed Hassan, Hoda Korashy. Sentiment analysis algorithms and applications: A survey. Ain Shams Engineering Journal Volume 5, Issue 4, December 2014, Pages 1093–1113

[14] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up?: sentiment classification using machine learning techniques,” Proceedings of the ACL-02 conference on Empirical methods in natural language processing, vol.10, 2002, pp. 79-86.

[15] Κωσταντίνος Καράλας. Ανάλυση Διάθεσης με Συνεχή Χρονικά Μοντέλα. Σχολή ΗΜΜΥ Πολυτεχνείου Κρήτης. 2013

[16] MR. S. M. VOHRA, PROF. J. B. TERAIYA. A COMPARATIVE STUDY OF SENTIMENT ANALYSIS TECHNIQUES. JOURNAL OF INFORMATION, KNOWLEDGE AND RESEARCH IN COMPUTER ENGINEERING. VOLUME – 02, ISSUE – 02, pp 313-317

[17] http://alias-i.com/lingpipe/index.html

[18] http://nlp.stanford.edu/sentiment/index.html

[19] https://github.com/ptgoetz/storm-hdfs

[20] https://sourceforge.net/projects/nbpfcrudgen/

[21] http://hadoop.apache.org/