



**ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ**

**ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ**

**ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ**

**Διακυβέρνηση Πληροφοριακών Συστημάτων  
ΑΠΑΛΛΑΚΤΙΚΗ ΕΡΓΑΣΙΑ**

Παναγιώτης Καραμολέγκος

ΑΜ : Ε17065

[p.karamolegos@yahoo.gr](mailto:p.karamolegos@yahoo.gr)

## Περιεχόμενα

Λίστα Εικόνων	ii
1. Θεματολογία Εφαρμογής Ανάλυσης Δεδομένων	1
2. Αρχιτεκτονική Εφαρμογής	2
2.1. Εργαλεία που χρησιμοποιήθηκαν	2
2.2. Υλοποιημένη Αρχιτεκτονική	4
3. Εγχειρίδιο Χρήσης Εφαρμογής	7
3.1. Εγκατάσταση Εφαρμογής	7
3.1.1. Απαραίτητες προϋποθέσεις Εγκατάστασης	7
3.1.2. Διαδικασία Εγκατάστασης Εφαρμογής	7
3.2. Εκτέλεση Εφαρμογής	12
3.2.1. Άνοιγμα Εφαρμογής	12
3.2.2. Αρχική Σελίδα	12
3.2.3. Data Collection phase	15
3.2.4. AVRO phase	18
3.2.5. Map Reduce Job phase	20
3.2.6. Preprocessing phase	21
3.2.7. Clustering phase	23
3.2.8. End phase	25
4. Αποτελέσματα Εφαρμογής	26
4.1. Keywords και συλλεχθέντα δεδομένα	26
4.2. Flume Agent configuration file	27
4.3. AVRO Schema	28
4.4. Αποτελέσματα MapReduce	29
4.5. Προ-επεξεργασία δεδομένων	30
4.6. Ανάλυση συλλεχθέντων δεδομένων	31
5. Επόμενα Βήματα	34
6. Αναφορές	35

## Λίστα Εικόνων

Εικόνα 1: Εργαλεία Ανάλυσης Δεδομένων .....	2
Εικόνα 2: Twitter API.....	3
Εικόνα 3: Εργαλεία στην οποία βασίστηκε ο προγραμματισμός της Εφαρμογής .....	3
Εικόνα 4: Αρχιτεκτονική Εφαρμογής .....	4
Εικόνα 5: Work Flow Πληροφοριακού Συστήματος.....	6
Εικόνα 6: Εγκατάσταση - Βήμα 1 .....	8
Εικόνα 7: Εγκατάσταση - Βήμα 2 .....	8
Εικόνα 8: Εγκατάσταση - Βήμα 3 .....	9
Εικόνα 9: Εγκατάσταση - Βήμα 4 .....	10
Εικόνα 10: Εγκατάσταση - Βήμα 9 .....	11
Εικόνα 11: Εκτέλεση της Εφαρμογής .....	12
Εικόνα 12: Αρχική Διεπαφή.....	13
Εικόνα 13: Ολοκλήρωση πρώτης φόρμας.....	14
Εικόνα 14: Data Collection phase .....	16
Εικόνα 15: Δεδομένα Flume .....	16
Εικόνα 16: Παρουσίαση συλλεγμένων tweets .....	17
Εικόνα 17: AVRO phase .....	18
Εικόνα 18: Avro Deserialized Data .....	19
Εικόνα 19: Serialized και Deserialized Avro Δεδομένα .....	19
Εικόνα 20: HDFS Map Reduce αποτελέσματα .....	20
Εικόνα 21: Map Reduce Job phase.....	20
Εικόνα 22: Preprocessing phase.....	21
Εικόνα 23: Προεπεξεργασμένα Δεδομένα .....	21
Εικόνα 24: Πρώτο Preprocessing Βήμα .....	22
Εικόνα 25: Δεύτερο Preprocessing Βήμα .....	22
Εικόνα 26: Clustering phase .....	23
Εικόνα 27: Αποτελέσματα Clustering .....	24
Εικόνα 28: Clustering Δεδομένα.....	24
Εικόνα 29: Εμφάνιση οργάνωσης της εφαρμογής στο HDFS .....	25
Εικόνα 30: End phase .....	25
Εικόνα 31: Topic HDFS κατάλογος .....	25
Εικόνα 32: Συλλεχθέντα δεδομένα .....	26
Εικόνα 33: Preprocessed αρχείο .....	31

## 1. Θεματολογία Εφαρμογής Ανάλυσης Δεδομένων

Η εργασία που αναπτύχθηκε ως απαλλακτική του μαθήματος «Διακυβέρνηση Πληροφοριακών Συστημάτων», σκοπεύει στην κατασκευή ενός συστήματος (με όνομα Twitter Data Analyzer) το οποίο μπορεί να βοηθήσει στην ανάλυση ενός αναλυτή δεδομένων σε γενικότερα προβλήματα τα οποία τον απασχολούν.

Μέσω του συστήματος που έχει κατασκευαστεί, ένας αναλυτής δεδομένων μπορεί να επιλέξει όσες λέξεις επιθυμεί ώστε να τις αναζητήσει στο Twitter 1% firehose Source, όπως επίσης να συλλέξει όσα tweets επιθυμεί για τα πειράματά του. Παράλληλα, του παρέχεται και η ικανότητα να κάνει πολλαπλά πειράματα, με διαφορετικές παραμέτρους κάθε φορά, τα οποία αποθηκεύονται οργανωμένα στον HDFS κατάλογο.

Για τον σκοπό της εργασίας, επιλέχθηκε ο τομέας του marketing και την εύρεση συσχετίσεων ανάμεσα σε υλικά τα οποία πωλούνται στο ευρύ κοινό. Έχοντας την ικανότητα συλλογής αφηρημένων μεγάλων δεδομένων από μία πηγή όπως το Twitter, κάθε επιχείρηση θα μπορεί να λάβει τις αντίστοιχες πρωτοβουλίες. Παραδείγματος χάρη, θα μπορούσε να ανακαλύψει αν συμφέρει να βάλει σε προσφορά κάποιο προϊόν του, στην περίπτωση που γνωρίζει ότι θα πουλήσει παραπάνω, λόγο του κοινού στο οποίο απευθύνεται, με σκοπό την αύξηση των κερδών του. Ένας πειραματισμός επάνω σε αυτό το πρόβλημα πραγματοποιείται στην **Ενότητα 4**.

Η ανάλυση μεγάλων δεδομένων που θα εφαρμοστεί, αποτελείται από του εξής τύπους:

- **Diagnostic analysis:** Θα γίνει ανάλυση σε εμφανιζόμενα clusters δεδομένων και συνεπώς θα μπορούν να βγουν συμπεράσματα για την ύπαρξη συσχέτισης κάποιων δεδομένων
- **Prescriptive analysis:** Μέσω της γνώσης που θα αποκτά ο αναλυτής από την διαγνωστική (Diagnostic) ανάλυση δεδομένων, θα μπορεί να λάβει μελλοντικές αποφάσεις επάνω στα προϊόντα τα οποία τον ενδιαφέρουν.

## 2. Αρχιτεκτονική Εφαρμογής

### 2.1. Εργαλεία που χρησιμοποιήθηκαν

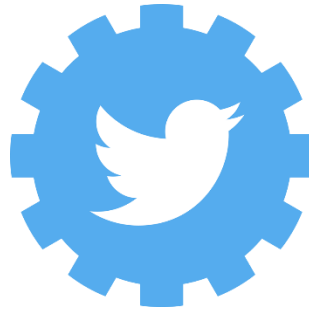
Η Εφαρμογή βασίζεται σε εργαλεία τα οποία είναι χρήσιμα στο περιβάλλον της ανάλυσης δεδομένων, αυτά τα εργαλεία είναι τα εξής:

- Apache Hadoop: Framework με βιβλιοθήκες οι οποίες επιτρέπουν την διαμοιρασμένη επεξεργασία μεγάλων συνόλων δεδομένων σε συμπλέγματα (clusters) υπολογιστικών μηχανών [1]. Από αυτό το εργαλείο, χρησιμοποιήθηκαν ποιο συγκεκριμένα:
  - Apache Hadoop HDFS (Hadoop Distributed File System): Σύστημα, το οποίο είναι εξαιρετικά ανεκτικό σε σφάλματα, για την αποθήκευση αρχείων σε σύνθετες hardware (υλικό το οποίο δεν είναι οικονομικά ακριβό) [2].
  - Apache Hadoop YARN (Yet Another Resource Negotiator): Εργαλείο το οποίο διαχειρίζεται τις διαδικασίες τις οποίες θέλει να εκτελέσει κάποιος χρήστης, πραγματοποιώντας κατανεμημένη εκτέλεση των εκτελούμενων εφαρμογών στους διαθέσιμους πόρους του συστήματος [3].
  - Apache Hadoop MapReduce: Framework για την διευκόλυνση προγραμματισμού Map Reduce διαδικασιών, οι οποίες περιέχουν μεγάλες ποσότητες δεδομένων [4].
- Apache AVRO: Σύστημα κωδικοποίησης αρχείων, ώστε να υπάρχουν σε μορφή συμπυκνωμένη και γρήγορα αναγνώσιμη από την γλώσσα η οποία το χρησιμοποιεί [5].
- Apache Flume: Υπηρεσία για την συλλογή και μεταφορά μεγάλων ποσοτήτων δεδομένων, με αξιόπιστο και κατανεμημένο τρόπο [6].
- Apache Kafka: Σύστημα το οποίο είναι ικανό να λάβει και να μεταφέρει, μεγάλες ποσότητες από streaming δεδομένα [7].
- Apache Mahout: Framework το οποίο υπάρχει για την διευκόλυνση, μαθηματικών στατιστικών και αναλυτών δεδομένων, στην υλοποίηση αλγορίθμων οι οποίοι έχουν να κάνουν με μηχανική μάθηση. Το Mahout βασίζεται στην γραμμική άλγεβρα καθώς επίσης μπορεί να εκτελέσει τις διαδικασίες του με κατανεμημένο τρόπο. [8].



Εικόνα 1: Εργαλεία Ανάλυσης Δεδομένων

Εκτός των εργαλείων που εμφανίζονται στην **Εικόνα 1**. Χρησιμοποιήθηκε και το Twitter API (**Εικόνα 2**) με σκοπό την επίτευξη της συλλογής των Tweet που είναι αποθηκευμένα εντός των Βάσεων Δεδομένων του.



Εικόνα 2: Twitter API

Για την λήψη αποφάσεων στην λειτουργικότητα της εφαρμογής, όπως και για την εμφάνισή της στον χρήστη και την ικανότητα της εφαρμογής να τρέχει σε hosted περιβάλλον χρησιμοποιήθηκαν τα εξής εργαλεία:

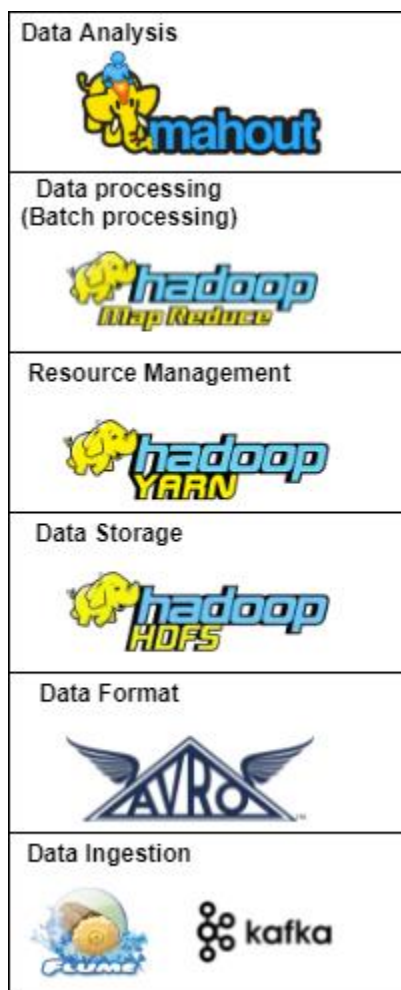
- Java
- JSP
- HTML
- JS
- CSS
- Apache Tomcat



Εικόνα 3: Εργαλεία στην οποία βασίστηκε ο προγραμματισμός της Εφαρμογής

## 2.2. Υλοποιημένη Αρχιτεκτονική

Στην **Εικόνα 4**, φαίνεται η αρχιτεκτονική με την οποία κατασκευάστηκε το σύστημα της εφαρμογής που έχει παραδοθεί.



Εικόνα 4: Αρχιτεκτονική Εφαρμογής

Περίληπτικά κατά την περισυλλογή των δεδομένων χρησιμοποιούνται τα εργαλεία Flume και Kafka (Flafka). Το Kafka παίρνει τα δεδομένα από το Twitter API και τα τοποθετεί σε ένα topic, ενώ στην συνέχεια χρησιμοποιεί το flume ως source για να τα τοποθετήσει στο HDFS (ως αποθήκη δεδομένων) το οποίο λειτουργεί ως sink.

Εφαρμόζουμε μορφοποίηση δεδομένων από απλό κείμενο (Plain text) σε μορφή AVRO (AVRO serialization και αντιστρόφως πίσω στην αρχική τους μορφή (AVRO deserialization).

Για τις διαδικασίες των βιβλιοθηκών του Hadoop χρησιμοποιείται το YARN με σκοπό την καλύτερη κατεύθυνση και διαχείριση των workloads των εφαρμογών που εκτελούνται.

Στην υλοποιημένη εφαρμογή, γίνονται επίσης διαδικασίες επεξεργασίας και ανάλυσης δεδομένων στις οποίες γίνεται χρήση των εργαλείων Hadoop MapReduce και Mahout αντίστοιχα.

Στην **Εικόνα 5**, φαίνεται μία ποιο αναλυτική και low level εξήγηση, ως work flow των διαδικασιών που πραγματοποιούνται στο παραδοτέο πληροφοριακό σύστημα. Στο πάνω μέρος φαίνεται το

user Interface και στο κάτω το HDFS και τα δεδομένα που αποθηκεύονται σε αυτό. Επίσης ακολουθείτε το εξής color coding στην πληροφορία της εικόνας:

- Με κόκκινο χρώμα απεικονίζονται οι διαδικασίες που έχουν πραγματοποιηθεί μέσω Java προγραμμάτων.
- Με πορτοκαλί χρώμα τα εργαλεία των οποίων χρησιμοποιούνται οι βιβλιοθήκες τους.
- Με μπλε χρώμα τα αποτελέσματα των διεπαφών της εφαρμογής.
- Με μαύρο χρώμα τα εργαλεία που τρέχουν στο παρασκήνιο για την επίτευξη κάποιου σκοπού της εφαρμογής.
- Με πράσινο χρώμα εμφανίζονται τα δεδομένα τα οποία τελικά αποθηκεύονται εντός του HDFS.

Η Εικόνα αυτή, διαβάζεται από τα αριστερά προς τα δεξιά και ξεκινάει από την εμφάνιση των λέξεων που θέλει ο χρήστης να αναζητήσει. Ύστερα, χρησιμοποιώντας το Flume και ενός Kafka Producer, συλλέγεται το πλήθος των tweets που θέλει ο αναλυτής δεδομένων και αποθηκεύονται στο HDFS.

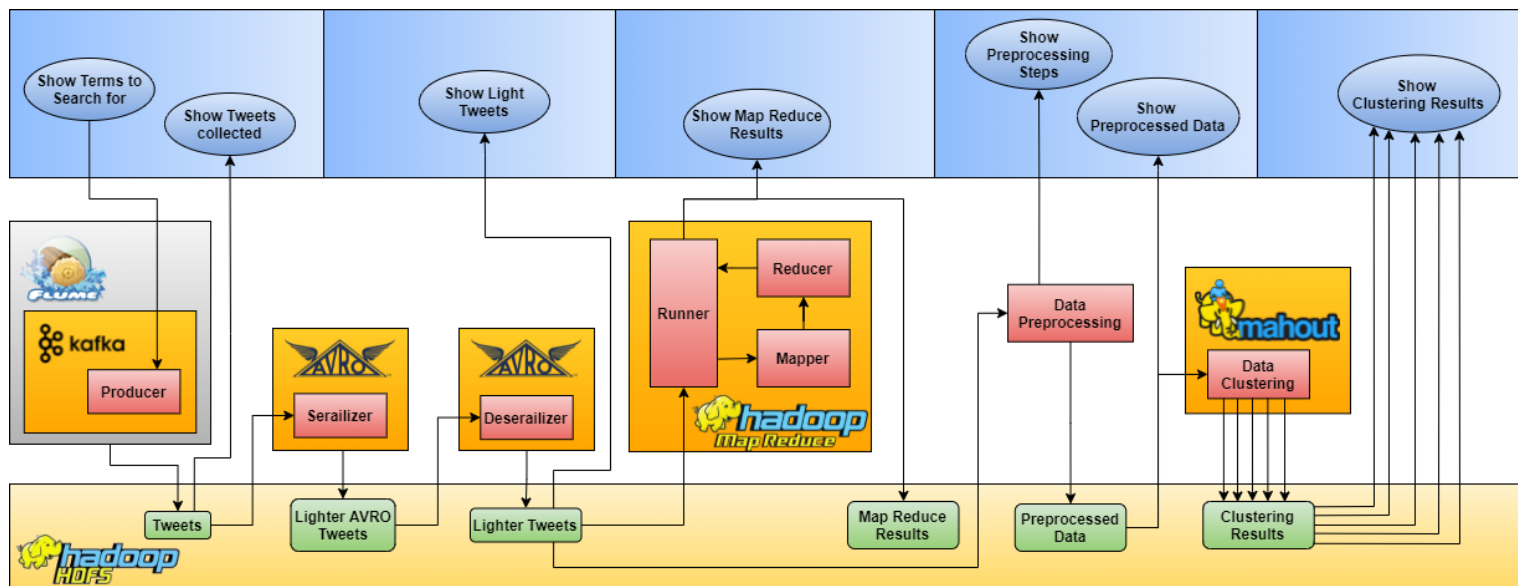
Συνεχίζοντας, γίνεται AVRO Serialization των αποθηκευμένων δεδομένων. Η εφαρμογή αυτή διαχειρίζεται μόνο το tweet του κάθε χρήστη και συνεπώς το AVRO schema έχει προγραμματιστεί να κρατάει μόνο το όνομα αυτού, το tweet και το timestamp του. Συνεπώς, αυτά είναι και τα δεδομένα που αποθηκεύονται στο HDFS ως «Lighter AVRO Tweets» και εμφανίζονται μετέπειτα στον τελικό χρήστη. Μετά από αυτήν την διαδικασία, γίνεται AVRO Deserialization των «ελαφρύτερων» δεδομένων και αποθηκεύονται σε πλέον plain text μορφή στο Data Storage της εφαρμογής.

Μετά, γίνεται χρήση των «ελαφρών» δεδομένων για μία διαδικασία Map Reduce, με την βοήθεια των αντίστοιχων βιβλιοθηκών του Hadoop, η οποία μετράει το πλήθος των φορών που υπάρχουν οι αναζητούμενες λέξεις εντός των tweet που έχουν συλλεχτεί. Εκτός από την αποθήκευση αυτών των αποτελεσμάτων στον χρήστη, γίνεται και αποθήκευσή τους στο HDFS.

Τα δεδομένα προεπεξεργάζονται με σκοπό την ανάλυσή τους. Η διεπαφή της εφαρμογής θα εμφανίσει τα βήματα της προεπεξεργασίας, καθώς και τα στάδια των δεδομένων κατά την διάρκεια αυτής της διαδικασίας, μέχρι την τελική τους παρουσίαση ως προεπεξεργασμένα δεδομένα και αποθήκευσή τους στο File System που χρησιμοποιεί η εφαρμογή.



Σε τελευταίο στάδιο, η διεπαφή της εφαρμογής, θα επιτρέψει στον τελικό χρήστη να κάνει όσα διαφορετικά πειράματα επιθυμεί, αποθηκεύοντας και παρουσιάζοντας όλα τα τελικά αποτελέσματα της K-means clustering ανάλυσης. Ο χρήστης μπορεί να διαλέξει να παραμετροποιήσει το clustering του ως προς το μέτρο απόστασης, το πλήθος των clusters, το πλήθος των επαναλήψεων και τη σύγκλιση Delta.



Εικόνα 5: Work Flow Πληροφοριακού Συστήματος

### 3. Εγχειρίδιο Χρήσης Εφαρμογής

#### 3.1. Εγκατάσταση Εφαρμογής

##### 3.1.1. Απαραίτητες προϋποθέσεις Εγκατάστασης

Για την εγκατάσταση του «Twitter Data Analyzer» πρέπει ο χρήστης να χρησιμοποιεί λειτουργικό σύστημα Windows και να κατέχει τα παρακάτω προγράμματα ήδη εγκατεστημένα στον υπολογιστή του:

- Apache Hadoop v3.1.0
  - Το HDFS και το YARN θα πρέπει να είναι εκτελέσιμα.
  - Θα πρέπει να υπάρχει μία μεταβλητή συστήματος με όνομα «HADOOP\_HOME» τοποθετημένη απευθείας στον κατάλογο του Hadoop.
- Apache Flume v1.9.0
  - Το Flume θα πρέπει να είναι εκτελέσιμο.
  - Θα πρέπει να υπάρχει μια μεταβλητή συστήματος με όνομα «FLUME\_HOME» τοποθετημένη απευθείας στον κατάλογο του Flume.
  - Ο κατάλογος του Flume θα πρέπει να είναι τοποθετημένος απευθείας στον C drive του υπολογιστή του χρήστη.
  - Θα πρέπει ο κατάλογος του Flume να έχει την εξής ονομασία «apache-flume-1.9.0-bin».
- Apache Kafka v2.13-2.7.0
  - Το Kafka θα πρέπει να είναι εκτελέσιμο.
  - Θα πρέπει να υπάρχει μια μεταβλητή συστήματος με όνομα «KAFKA\_HOME» τοποθετημένη απευθείας στον κατάλογο του Kafka.
- Apache Mahout 0.12.0
  - Το Mahout θα πρέπει να είναι εκτελέσιμο.
- Eclipse IDE for Enterprise Java Developers (μη υποχρεωτικό - εάν πληρείται με άλλο τρόπο)
- Apache Tomcat v9.0 Server (μη υποχρεωτικό - εάν πληρείται με άλλο τρόπο)
  - Ο Tomcat θα πρέπει να είναι εγκατεστημένος εντός του Eclipse.
  - Ο Tomcat θα πρέπει να τρέχει επάνω στο localhost.

Θα πρέπει ο χρήστης που χρησιμοποιεί την εφαρμογή να έχει πρόσβαση απευθείας στον C drive του υπολογιστή, καθώς η εφαρμογή θα τοποθετήσει ένα νέο directory εντός του, για να μπορέσει να διαχειριστεί ορισμένες λειτουργίες της. Συνεπώς, εφόσον ζητηθεί από την εφαρμογή (μέσω Windows alert message), ο χρήστης θα πρέπει να έχει την δυνατότητα να της παραχωρήσει πρόσβαση.

##### 3.1.2. Διαδικασία Εγκατάστασης Εφαρμογής

Ο παραδοτέος φάκελος της εφαρμογής αυτής περιέχει τον εξής κατάλογο:

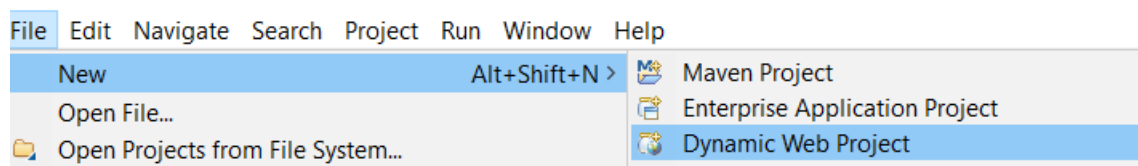
my\_files/dynamic\_web\_project/ISG\_final

Ο κατάλογος ISG\_final είναι και το Java dynamic web project της υλοποιημένης εφαρμογής. Μέσα του περιέχει, όλα τα αναγκαία jars για την εκτέλεσή της, όπως και όλα τα interfaces και java κλάσεις οι οποίες χρειάζονται για την εγκατάστασή της.

Αν και υπάρχουν πολύ τρόποι για την προσθήκη του project εντός του eclipse, θα αναφερθεί παρακάτω ο πιο ασφαλής από αυτούς, με τον οποίο η εφαρμογή θα έχει τις περισσότερες πιθανότητες να εκτελεστεί σωστά.

### **Βήμα 1**

- Κατασκευή νέου Java Dynamic Web Project στο Eclipse (**Εικόνα 6**).
- Εισαγωγή Project Name (Για τα βήματα αυτά θα δοθεί το όνομα Twitter\_Data\_Analyzer).
- Πατάμε το κουμπί Next.
- Πατάμε το κουμπί Next.
- Ενεργοποιούμε το πεδίο «Generate web.xml deployment descriptor».



**Εικόνα 6: Εγκατάσταση - Βήμα 1**

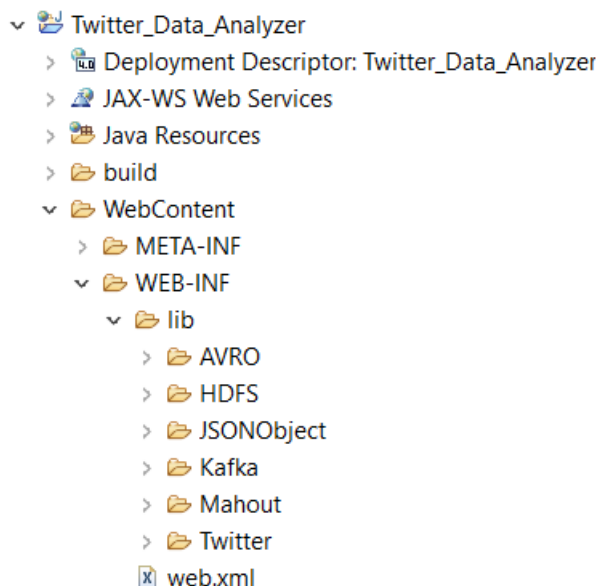
### **Βήμα 2**

- Από τον παραδοτέο φάκελο, μετακινούμε όλο το περιεχόμενο του καταλόγου **(1)**, εντός του καταλόγου του Java Project μας **(2)**.

my\_files\dynamic\_web\_project\ISG\_final\WebContent\WEB-INF\lib **(1)**

Twitter\_Data\_Analyzer\WebContent\WEB-INF\lib **(2)**

Θα πρέπει το Project μας να έχει καταλήξει όπως και η **Εικόνα 7**.



**Εικόνα 7: Εγκατάσταση - Βήμα 2**

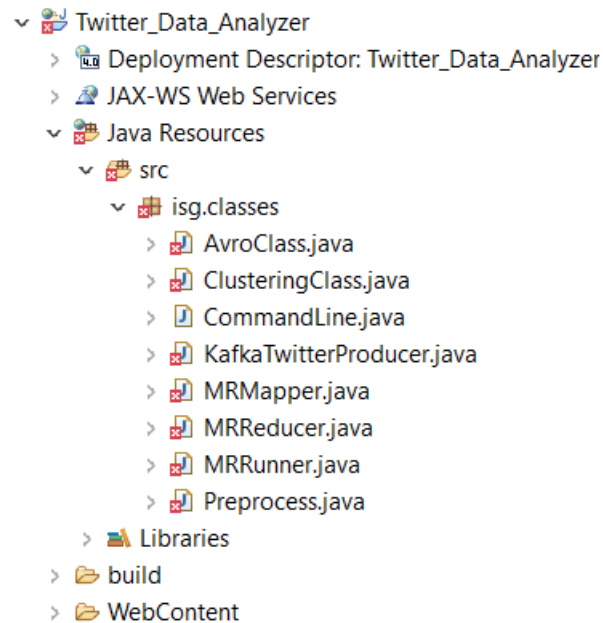
### **Βήμα 3**

- Από τον παραδοτέο φάκελο, μετακινούμε όλο το περιεχόμενο του καταλόγου **(3)**, εντός του καταλόγου του Java Project μας **(4)**.

my\_files\dynamic\_web\_project\ISG\_final\src **(3)**

Twitter\_Data\_Analyzer\src **(4)**

Θα πρέπει το Project μας να έχει καταλήξει όπως και η **Εικόνα 8**.



**Εικόνα 8: Εγκατάσταση - Βήμα 3**

### **Βήμα 4**

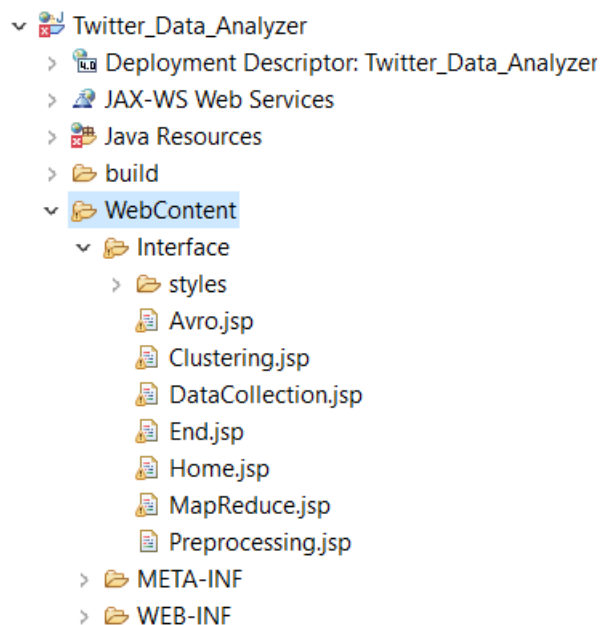
- Από τον παραδοτέο φάκελο, μετακινούμε όλο το περιεχόμενο του καταλόγου Interface (Με τον φάκελο Interface) **(5)**, εντός του καταλόγου του Java Project μας **(6)**.

my\_files\dynamic\_web\_project\ISG\_final\WebContent\Interface **(5)**

Twitter\_Data\_Analyzer\WebContent **(6)**

Θα πρέπει το Project μας να έχει καταλήξει όπως και η **Εικόνα 9**.

Πλέον έχουμε τοποθετήσει όλα τα απαραίτητα αρχεία για να εκτελεστεί η εφαρμογή μας. Όμως πρέπει να ακολουθήσουμε περαιτέρω διαδικασίες για να κάνουμε την εφαρμογή εκτελέσιμη.



Εικόνα 9: Εγκατάσταση - Βήμα 4

### **Βήμα 5**

- Δεξί κλικ επάνω στο Java Project.
- Build Path → Configure Build Path → Libraries → Add JARS... → Twitter\_Data\_Analyzer → WebContent → WEB-INF → lib.
- Προσθέτουμε όλα τα JARS των υποφακέλων του lib καταλόγου.
- Πατάμε το Add JARS μέχρι να προσθέσουμε όλα τα JARS που χρειάζονται. (Χρήσιμο Key Shortcut είναι το CTRL+SHIFT για την επιλογή πολλαπλών αρχείων στην σειρά).
- Επιλέγουμε «Apply and Close».

Αυτό το βήμα λόγω της υψηλής επεξεργασίας που θα πραγματοποιήσει στο background μπορεί να αργήσει να λειτουργήσει – Θα έχει λειτουργήσει όταν θα έχει φύγει το κόκκινο error από το Java Project που χρησιμοποιούμε.

### **Βήμα 6**

- Δεξί κλικ επάνω στο Java Project.
- Επιλέγουμε «Properties».
- Επιλέγουμε «Deployment Assembly».
- Add → Java Build Path Entries → Next → Επιλέγουμε όλα τα Jars που έχουν εμφανιστεί. (Χρήσιμο Key Shortcut είναι το CTRL+A για την επιλογή όλων των περιεχομένων).
- Επιλέγουμε «Finish».
- Επιλέγουμε «Apply and Close».

## **Βήμα 7**

- Κάνουμε κλικ επάνω στο project μας
- Επάνω στο Eclipse Toolbar επιλέγουμε «Project»
- Επιλέγουμε «Clean...»
- Αφαιρούμε το κουμπί «Clean all projects»
- Επιλέγουμε μόνο το project μας
- Επιλέγουμε «Clean»

Πλέον το Project μας έχει εγκατασταθεί, αλλά μένουν ακόμα μερικά βήματα ώστε να γνωρίζουμε ότι ο server μας είναι ικανός να τρέξει την εφαρμογή μας.

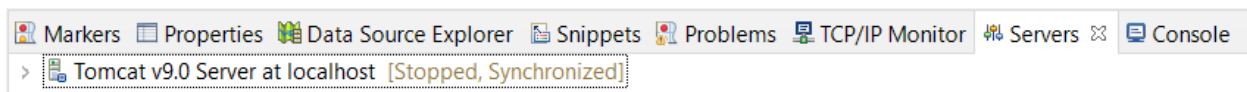
## **Βήμα 8**

Σιγουρευόμαστε ότι το Eclipse μας στις επιπλέον πληροφορίες του (το παράθυρο στο οποίο υπάρχει η κονσόλα), κατέχει το tab «Servers» Αν δεν υπάρχει τότε ακολουθούμε το **Βήμα 9**, αλλιώς πάμε κατευθείαν στο **Βήμα 10**.

## **Βήμα 9**

- Επάνω στο Eclipse Toolbar επιλέγουμε «Window»
- Show view → Servers

Το μορφή του παραθύρου επιπλέον πληροφοριών του Eclipse θα πρέπει να είναι παρόμοια με της **Εικόνας 10**.



***Εικόνα 10: Εγκατάσταση - Βήμα 9***

## **Βήμα 10**

- Κάνουμε double κλικ επάνω στον Tomcat server στο tab Servers
- Βρίσκουμε και πατάμε την κεφαλίδα «Timeouts»
- Αυξάνουμε το πεδίο «Start (in seconds)» (Αυτό το πεδίο πρέπει να αυξηθεί αρκετά, καθώς η εφαρμογή περιέχει πολλά JARS και συνεπώς είναι αρκετά απαιτητική. Είναι πιθανό λοιπόν, ο Server να μην προλάβει να ανοίξει και συνεπώς, να μην επιτρέψει στην εφαρμογή να λειτουργήσει. Αργά μηχανήματα θα πρέπει να έχουν την τιμή αυτού του πεδίου στα 180 δευτερόλεπτα τουλάχιστον.)

## **Βήμα 11**

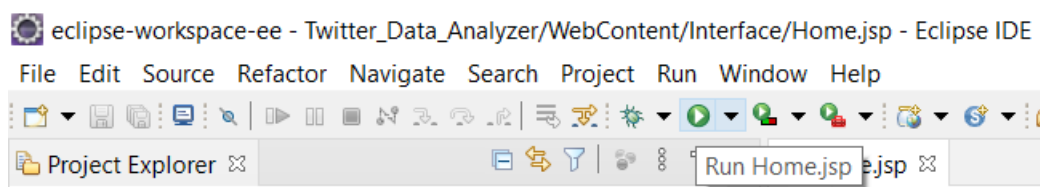
- Δεξί κλικ επάνω στον Tomcat Server
- Επιλέγουμε «Add and Remove...»
- Σιγουρευόμαστε ότι θα τρέξουν στον Server μόνο τα Project που είναι ανάγκη να τρέξουν (Το Java Project που δημιουργήθηκε κατά την διάρκεια αυτών των βημάτων θα πρέπει να τοποθετηθεί στα Configured projects – τα δεξιά δηλαδή)
- Επιλέγουμε «Finish»

Πλέον είμαστε έτοιμοι να εκτελέσουμε την εφαρμογή!

## 3.2. Εκτέλεση Εφαρμογής

### 3.2.1. Άνοιγμα Εφαρμογής

- Για να εκτελεστεί η εφαρμογή, περιηγηθείτε στο Java Project στο παρακάτω αρχείο: Twitter\_Data\_Analyzer\WebContent\Interface\Home.jsp
- Στην συνέχεια πατήστε εντός του αρχείου Home.jsp.
- Πατήστε το κουμπί «Run» του Eclipse το οποίο, όταν πλέον κάνετε hover από πάνω του, θα σας λέει «Run Home.jsp» (**Εικόνα 11**).
- Επιλέξτε τον Tomcat και στην συνέχεια το κουμπί «Finish». (Αυτό το βήμα είναι αρκετά απαιτητικό και μπορεί να καθυστερήσει να ολοκληρωθεί).
- Περιμένετε μέχρι ο Server σας να ανοίξει και να τρέξει το αρχείο. (Το Eclipse θα εμφανίσει μία πειραματική διεπαφή όταν γίνει αυτό).
- Ανοίξτε έναν browser της επιλογής σας (Π.χ. Google Chrome).
- Περιηγηθείτε στο παρακάτω link:  
[http://localhost:8080/Twitter\\_Data\\_Analyzer/Interface/Home.jsp](http://localhost:8080/Twitter_Data_Analyzer/Interface/Home.jsp)



*Εικόνα 11: Εκτέλεση της Εφαρμογής*

### 3.2.2. Αρχική Σελίδα

Το υπόλοιπο της **Ενότητας 3** θα χρησιμοποιηθεί για να εξηγήσει στον χρήστη τον τρόπο χρήσης της υλοποιημένης εφαρμογής αλλά και για να δείξει την διαδικασία που ακολουθήθηκε για την ανάλυση δεδομένων η οποία αναφέρθηκε στην Ενότητα 1. Τα αποτελέσματα της ανάλυσης που θα λάβει μέρος σε αυτήν την Ενότητα, θα είναι και αυτά που θα παρουσιαστούν στην **Ενότητα 4**.

Αφού ανοίξουμε την Εφαρμογή (**Ενότητα 3.2.1**), εμφανίζεται μία διεπαφή, όπως αυτή της **Εικόνας 12**. Σε αυτήν, τοποθετούμε τα κλειδιά που μας παρέχει το Twitter API και στην συνέχεια επιλέγουμε το πλήθος των tweets που θέλουμε να συλλέξουμε, όπως και το Topic επάνω στο οποίο θέλουμε να κάνουμε την ανάλυσή μας.

Ύστερα, επιλέγουμε το πλήθος των λέξεων που θέλουμε να αναζητήσουμε, μέσω του Twitter API και πατάμε το κουμπί «Get keyword fields». Το κουμπί, θα εμφανίσει κάτω από αυτό, όσα πεδία επιθυμούμε, ώστε να γράψουμε σε αυτά τους όρους προς αναζήτηση. Στην συνέχεια μπορούμε να πατήσουμε το κουμπί «Go» ώστε η εφαρμογή να ξεκινήσει να μαζεύει δεδομένα (**Εικόνα 13**).

Μετά το πάτημα του τελευταίου κουμπιού, θα εμφανιστούν διάφορα CMD παράθυρα τα οποία αφορούν την ομαλή εκτέλεση του Kafka, HDFS, YARN και Flume. Ο χρήστης δεν πρέπει να κλείσει κανένα από αυτά τα παράθυρα. Όταν έχει γίνει η συλλογή δεδομένων, τότε η διεπαφή που βλέπει ο χρήστης θα αλλάξει και θα έχει πλέον τίτλο «Data Collection phase» (**Εικόνα 14**). Ανάλογα το πλήθος των δεδομένων που θέλει να μαζέψει ο χρήστης, μπορεί να χρειαστεί πολύς χρόνος για να φορτώσει η επόμενη σελίδα!

Οι κωδικοί του Twitter API είναι κρυμμένοι, συνεπώς ο κάθε χρήστης θα πρέπει να παρέχει τους δικούς του. Για την εκτέλεση αυτής της ανάλυσης το όνομα του topic είναι «Marketing» και έχει

γίνει επιλογή συλλογής 2000 συνολικά tweets. Οι λέξεις οι οποίες έχουν επιλεχτεί, αφορούν προϊόντα τα οποία αγοράζονται συχνά από το ευρύ κοινό. Πιο συγκεκριμένα αυτά τα προϊόντα είναι:

- Χαλάκι πόρτας
- Ακουστικά
- Πετσέτα
- Σέλες
- Βιβλιοθήκη
- Στήριγμα βιβλίων
- Μπάλα
- Βιβλίο
- Σκούπα
- AirPods

The image shows a web application titled "Twitter Data Analyzer". It has a black header with the title in white. Below the header, there is a light blue section with instructions: "Give your data and preferences below and then click the 'Go' button. After you press the button, depending on the amount of tweets that you want to collect, it may take a long time to load the next page! Do not close any CMD windows from the ones that will be opened by this application unless if the application tells you to do it!". The main form area has a light blue background and contains several input fields and a button. The fields are labeled: "Consumer Key:", "Consumer Secret:", "Token:", "Secret:", "Amount of Tweets to collect (positive integer):", "Your Topic Name:", "Amount of Keywords to search for (positive integer):", and "Keyword 1:". The "Amount of Keywords to search for" field contains the number "1". There is a green button labeled "Get keyword fields" below the keyword search field. At the bottom of the form is a large green button labeled "Go".

Εικόνα 12: Αρχική Διεπαφή



# Twitter Data Analyzer

Give your data and preferences below and then click the "Go" button.  
After you press the button, depending on the amount of tweets that you want to collect, it may take a long time to load the next page!  
Do not close any CMD windows from the ones that will be opened by this application unless if the application tells you to do it!

Consumer Key:

Consumer Secret:

Token:

Secret:

Amount of Tweets to collect (positive integer):

Your Topic Name:

Amount of Keywords to search for (positive integer):

Keyword 1:

Keyword 2:

Keyword 3:

Keyword 4:

Keyword 5:

Keyword 6:

Keyword 7:

Keyword 8:

Keyword 9:

Keyword 10:

Εικόνα 13: Ολοκλήρωση πρώτης φόρμας

Πριν εμφανιστεί στον χρήστη η νέα διεπαφή (της **Εικόνας 14**), αρκετές διαδικασίες εκτελέστηκαν με αυτοματοποιημένο τρόπο για την διευκόλυνσή του. Αυτές είναι οι εξής:

- Εκτέλεση του Kafka Zookeeper και Kafka Broker.
- Εκτέλεση του HDFS και του YARN.
- Αναμονή 30 δευτερολέπτων ώστε τα παραπάνω να έχουν προλάβει να ξεκινήσουν.
- Έξοδος του safemode από το NameNode.
- Προσθήκη ενός καταλόγου απευθείας στην ρίζα του HDFS με το όνομα του Topic που έχει δώσει ο χρήστης.
- Αναμονή 30 δευτερολέπτων ώστε τα παραπάνω να έχουν προλάβει να εκτελεστούν.
- Δημιουργία του Flume Agent (twitter\_flume.conf) απευθείας μέσα στο path που φαίνεται παρακάτω. Ο Agent αυτός γνωρίζει το Kafka Topic που θα χρησιμοποιηθεί και την τοποθεσία που θα αφήσει τα δεδομένα που θα λάβει, εντός του νέου καταλόγου στο HDFS, με σκοπό την οργάνωση των αρχείων του χρήστη.  
C:\apache-flume-1.9.0-bin\conf
- Αναμονή 30 δευτερολέπτων ώστε να έχει προλάβει να ξεκινήσει ομαλά ο Flume Agent.
- Απόκτηση των δεδομένων που θέλει ο χρήστη μέσω του συνδυασμού Flafka.
- Αναμονή 10 δευτερολέπτων ώστε να έχει προλάβει να τερματίσει ομαλά ο Kafka Producer.

### 3.2.3. Data Collection phase

Στην εμφάνιση της δεύτερης διεπαφής, ο χρήστης μπορεί πλέον να κλείσει το CMD παράθυρο το οποίο τρέχει τον Flume Agent (το τελευταίο παράθυρο που του άνοιξε η εφαρμογή), πατώντας CTRL+C για να τερματίσει ομαλά και να τοποθετήσει σωστά τα δεδομένα στο HDFS. (Στην περίπτωση που ο χρήστης επιθυμεί, μπορεί να κλείσει και τα CMD που αφορούν τον Kafka Zookeeper και τον Kafka Broker, αλλά δεν είναι αναγκαίο για την ολοκλήρωση της εφαρμογής).

Από αυτήν την διεπαφή και ύστερα, όλα τα αποτελέσματα του χρήστη θα αποθηκεύονται εντός του HDFS. Η διεπαφή, θα του δίνει το link στο οποίο έχει αποθηκεύσει τα δεδομένα του κάθε φορά, ώστε να μην υπάρχει ανάγκη να ψάχνει τους καταλόγους του HDFS για να τα βρει.

Στην **Εικόνα 15**, φαίνεται η αποθήκευση των συλλεχθέντων tweets στο HDFS. Ο χρήστης, μπορεί να βάλει το όνομα του Flume αρχείου του (για το πείραμα της **Εικόνα 15** είναι FlumeData.1625750434949) και να λάβει τα tweets του απευθείας στο κάτω μέρος της διεπαφής η οποία παρουσιάζεται μπροστά του, πατώντας το κουμπί «VIEW TWEETS» (**Εικόνα 16**). Στην περίπτωση που το πλήθος των δεδομένων είναι μεγάλο, είναι πιθανό να καθυστερήσει να φορτώσει η σελίδα. Το κάθε tweet θα εμφανίζεται αριθμημένο μαζί με ολόκληρο το περιεχόμενο που λήφθηκε από το Twitter API.

Στην συνέχεια, πατώντας το κουμπί «FORMAT TWEETS IN AVRO» κατευθυνόμαστε στην επόμενη φάση της ανάλυσης, η οποία είναι η μορφοποίηση των δεδομένων σε AVRO.

# Data Collection phase

**To continue, stop the CMD window that was running the Flume Agent by pressing CTRL+C until it stops (The last CMD that opened), then you can close this CMD window.**

Insert the name of the file that Flume created in the HDFS directory below:

[Tweeter data](#)

(You want something like: FlumeData.1625350796794 - Remember to Refresh the HDFS so you will not get a .tmp file!)

Name of the Flume file (with the suffix):

Press this button to view the tweets that you collected:

VIEW TWEETS

When ready, press the button below to save your tweets in AVRO format and go to the next page, so you can deserialize them also.

FORMAT TWEETS IN AVRO

[Εικόνα 14: Data Collection phase](#)

## Browse Directory

<input type="text" value="/Marketing/mytweets"/>									Go!			
Show	25	entries	Search: <input type="text"/>									
<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name				
<input type="checkbox"/>	-rw-r--r--	USER	supergroup	12.21 MB	Jul 08 18:53	1	128 MB	FlumeData.1625759400521				

[Εικόνα 15: Δεδομένα Flume](#)

**Your Data:**

TWEET NUMBER 1

[illegible]

**Εικόνα 16: Παρουσίαση συλλεγμένων tweets**

### 3.2.4. AVRO phase

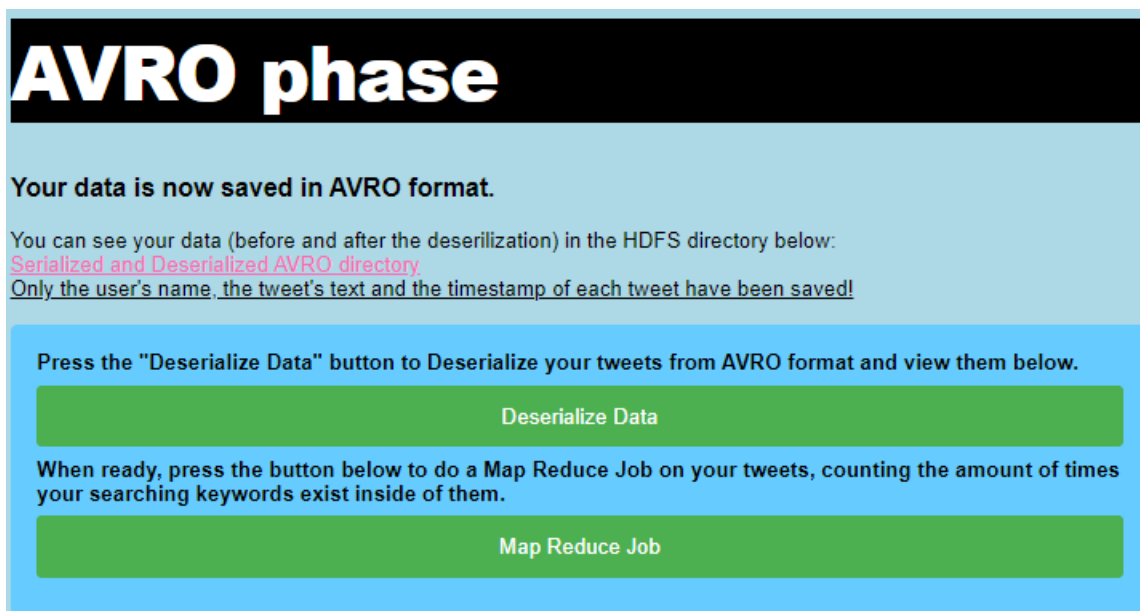
Ο χρήστης στην επόμενη διεπαφή που του εμφανίζεται (**Εικόνα 17**), ενημερώνεται ότι από τα δεδομένα του, πλέον έχουν κρατηθεί ορισμένα πεδία μόνο και ότι έχουν μορφοποιηθεί σε AVRO. Πατώντας στο κουμπί «Deserialize Data», γίνεται μορφοποίηση των δεδομένων ξανά, σε κανονικό κείμενο και αποτυπώνονται στην διεπαφή (**Εικόνα 18**). Όπως και στην προηγούμενη σελίδα, ο χρήστης μπορεί να βρει κατευθείαν σύνδεσμο για το HDFS κατάλογο, στον οποίο είναι αποθηκευμένα τα σειριοποιημένα και αντίστροφα σειριοποιημένα αρχεία του (**Εικόνα 19**).

Πατώντας το κουμπί «Map Reduce Job» θα πραγματοποιηθεί μία διαδικασία Map Reduce (Παραπάνω στην **Ενότητα 3.2.5**) και η εφαρμογή θα ξεκινήσει να κατευθύνεται στην επόμενη της φάση.

Κατά την διάρκεια του AVRO phase, η εφαρμογή εκμεταλλεύεται έναν φάκελο που κατασκευάζει η ίδια στο παρακάτω μονοπάτι:

C:\TwitterAnalyzer

Αυτός ο φάκελος χρησιμοποιείται, ώστε ο χρήστης να μην χρειάζεται να εγκαταστήσει το Avro σχήμα αλλά να εγκατασταθεί αυτόματα, όπως επίσης για την αποθήκευση των deserialized δεδομένων, με τρόπο που θα τα χρησιμοποιεί αυτοματοποιημένα σε επόμενη φάση, η διαδικασία Map Reduce.



[Εικόνα 17: AVRO phase](#)

Below is the saved information for your tweets deserialized from AVRO format:

TWEET NUMBER 1

```
{ "username": "Stephen Dreyse", "tweet": "RT @Goodkinds: House FIRE [Financial Independence, Retire Early]: How to Be a Red\u00fa2013Hot Real Estate Millionaire with a Wealth of Time and M\u00fa2026", "timestamp": 1625759388674 }
```

TWEET NUMBER 2

```
{ "username": "ðŸŒšGrace ðŸŒš", "tweet": "RT @AuntieJo14: A rather large petri dish for #COVID19 \n\nThe U.K. will have tens of thousands of new cases every day by the end of this mon\u2026", "timestamp": 1625759388677 }
```

TWEET NUMBER 3

```
{ "username": "Gregory Norminton", "tweet": "RT @GDRNorminton: I teach (delightful) creative writing students, to whom the @McrrWritingSchl lecturers may seem to have 'arrived'. Yet her!u2026", "timestamp": 1625759388891 }
```

TWEET NUMBER 4

```

{"username": "Kevin The Truth 8Y ±", "tweet": "RT @tacaboutpics: Austin Texas weekend of August 20th\\n\\nHouston Texas the week of August 23rd\\n\\nBook your photoshoots 8Y ± https://t.co/qIRCVII7u2026", "timestamp": 1625759388957}

```

TWEFT NUMBER 5

```
{ "username": "Jack", "tweet": "RT @SocanalysisHQ: How many centre forwards in world football receive this ball on the half turn, have the awareness of the run and execute w2026", "timestamp": 1625759388979 }
```

TWEET NUMBER 6

```
{ "username": "BLUEISTHECOLOUR", "tweet": "@SkySportsPL I agree carragher knows ball", "timestamp": 1625759389104 }
```

TWFEFT NUMBER 7

[illegible]

TWEET NUMBER 8

[illegible]

TWEET NUMBER 9

```
{ "username": "Neil Matouka", "tweet": "And lastly, by promoting the idea to wait 10-20 years, you\u2019re creating climate denial (see @MichaelEMann book) and\u2026", "timestamp": "1625759389275" }
```

TWEET NUMBER 10

[illegible]

TWEET NUMBER 11

```
{ "username": "PrinterGoBrrr", "tweet": "RT @JaniceDean: lu2019CWe want to do with gun violence with what we just did with COVID." @NYGovCuomo \n\nOver 9,000 loaded guns sent to nursing ho\nlu2026", "timestamp": 1625759389447 }
```

TWEET NUMBER 12

```
{username: "rhongnimdō", "text": "ceō- ceō- ce", "tweet": "RT @AlwaysWithJihyo: When sana accidentally hit momo's forehead with the ping pong ball and it bounces off just like that ðŸ™¸ https://t.co/XGlu2026", "timestamp": 1625759389492}
```

TWEET NUMBER 13

f"/username": "Veritas" "tweet": "@RateDan0 @Macero3000 @iann0 @inklekr @DonTunni1900 Kommt darauf an wie sich Dinge unser Spiel

### Εικόνα 18: Avro Deserialized Data

## Browse Directory

Show  entries

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	<input type="checkbox"/>
<input type="checkbox"/>	-rw-r--r--	<a href="#">USER</a>	<a href="#">supergroup</a>	322.29 KB	Jul 08 16:39	<a href="#">3</a>	128 MB	<a href="#">avrotweets.avro</a>	<input type="checkbox"/>
<input type="checkbox"/>	-rw-r--r--	<a href="#">USER</a>	<a href="#">supergroup</a>	428.55 KB	Jul 08 17:00	<a href="#">3</a>	128 MB	<a href="#">deserttweets.txt</a>	<input type="checkbox"/>

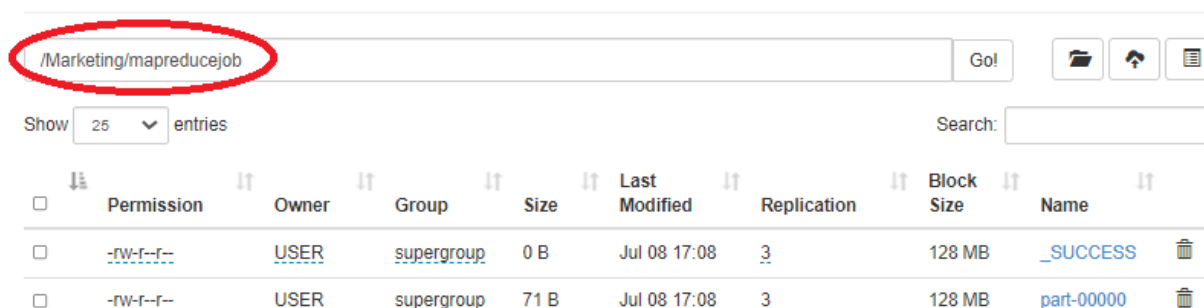
**Εικόνα 19: Serialized και Deserialized Αγρο Δεδομένα**

### 3.2.5. Map Reduce Job phase

Στην διαδικασία Map Reduce, χρησιμοποιούνται τα deserialized δεδομένα του φακέλου TwitterAnalyzer που έχει κατασκευάσει η εφαρμογή, απευθείας στον C drive, με σκοπό να μετρηθεί το πλήθος των λέξεων από αυτές που αναζητά ο αναλυτής εντός του συνόλου των tweets. Στην συνέχεια αφού αποθηκευτούν τα αποτελέσματα στο HDFS (**Εικόνα 20**), εμφανίζεται η διεπαφή μαζί με αυτά, όπως επίσης και έναν σύνδεσμο τους στο HDFS (**Εικόνα 21**).

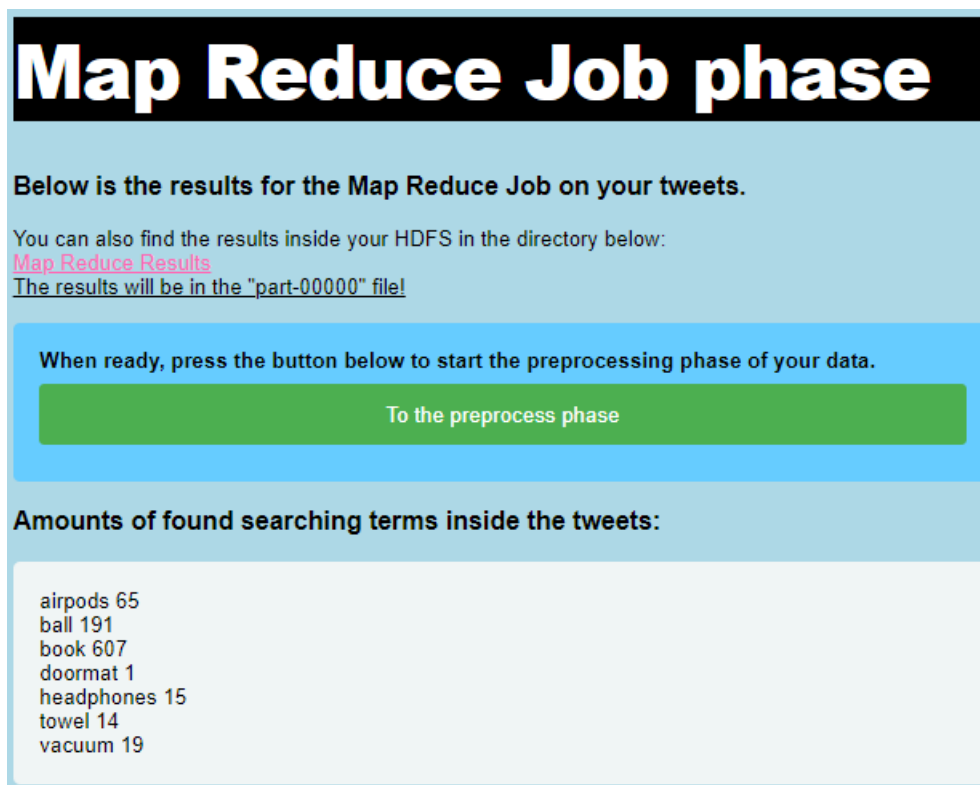
Πατώντας το κουμπί «To the preprocess phase», ο αναλυτής κατευθύνεται στην επόμενη φάση της εφαρμογής.

## Browse Directory



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	USER	supergroup	0 B	Jul 08 17:08	3	128 MB	_SUCCESS
-rw-r--r--	USER	supergroup	71 B	Jul 08 17:08	3	128 MB	part-00000

[Εικόνα 20: HDFS Map Reduce αποτελέσματα](#)



## Map Reduce Job phase

Below is the results for the Map Reduce Job on your tweets.

You can also find the results inside your HDFS in the directory below:  
[Map Reduce Results](#)  
 The results will be in the "part-00000" file!

When ready, press the button below to start the preprocessing phase of your data.

To the preprocess phase

Amounts of found searching terms inside the tweets:

```
airpods 65
ball 191
book 607
doormat 1
headphones 15
towel 14
vacuum 19
```

[Εικόνα 21: Map Reduce Job phase](#)



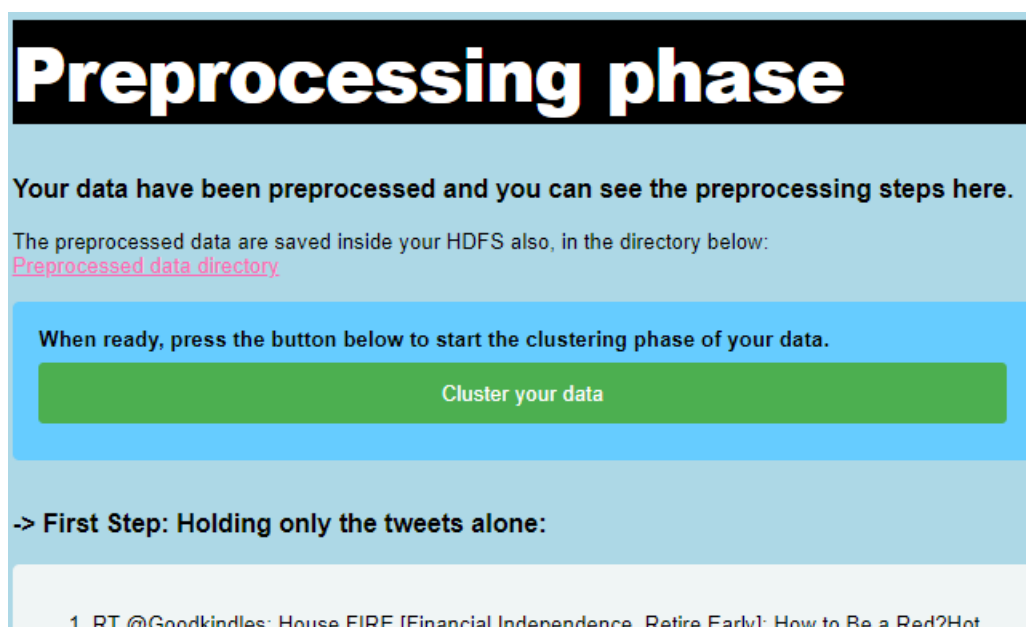
### 3.2.6. Preprocessing phase

Στην φάση της προεπεξεργασίας, η εφαρμογή προεπεξεργάζεται τα δεδομένα που έχουν γίνει Deserialized από την μορφοποίηση AVRO και αφού τα αποθηκεύσει στο HDFS, εμφανίζει στον χρήστη την διεπαφή της (**Εικόνα 22**), δείχνοντάς του τα βήματα που ακολούθησε. Ο χρήστης, όπως και στις προηγούμενες φάσεις, λαμβάνει και έναν σύνδεσμο για τον κατάλογο εντός του HDFS, στον οποίο έχουν αποθηκευτεί τα προεπεξεργασμένα δεδομένα του (**Εικόνα 23**). (Για να δει τα βήματα της προεπεξεργασίας ο χρήστης πρέπει να κάνει scroll down).

Τα βήματα προεπεξεργασίας είναι τα εξής:

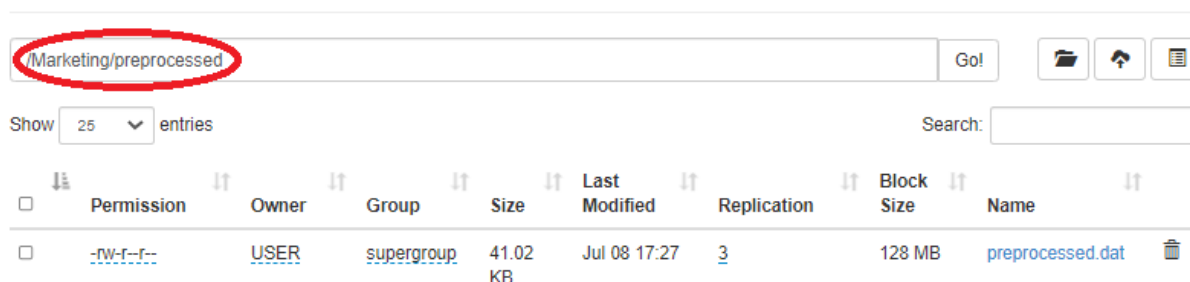
- Διατήρηση μόνο του tweet από κάθε record (**Εικόνα 24**).
- Μεταμόρφωση κάθε tweet σε ένα array ακεραίων. Κάθε ακέριος αναπαριστά το πλήθος των φορών που υπάρχει η αντίστοιχη λέξη προς αναζήτηση, εντός του text αυτού. (**Εικόνα 25**).

Όταν ο αναλυτής δεδομένων θελήσει, μπορεί να πατήσει το κουμπί «Cluster your data», ώστε να προχωρήσει στην φάση του Clustering των δεδομένων του.



[Εικόνα 22: Preprocessing phase](#)

## Browse Directory



[Εικόνα 23: Προεπεξεργασμένα Δεδομένα](#)



-> First Step: Holding only the tweets alone:

- [illegible]

### Εικόνα 24: Πρώτο Preprocessing Βήμα

-> Second Step: Counted the amount of times each searching term is inside of each tweet:

Tweet#	term	door mat	headphones	towel	saddle	bookcase	bookend	ball book	vacuum	air pods
1		0	0	0	0	0	0	0	0	0
2		0	0	0	0	0	0	0	0	0
3		0	0	0	0	0	0	0	0	0
4		0	0	0	0	0	0	0	0	0
5		0	0	0	0	0	0	1	0	0
6		0	0	0	0	0	0	1	0	0
7		0	0	0	0	0	0	1	0	0
8		0	0	0	0	0	0	0	0	0
9		0	0	0	0	0	0	0	0	0
10		0	0	0	0	0	0	1	0	0
11		0	0	0	0	0	0	0	0	0
12		0	0	0	0	0	0	1	0	0
13		0	0	0	0	0	0	0	0	0
14		0	0	0	0	0	0	0	0	0
15		0	0	0	0	0	0	1	0	0
16		0	0	0	0	0	0	1	0	0
17		0	0	0	0	0	0	1	0	0
18		0	0	0	0	0	0	0	0	0
19		0	0	0	0	0	0	0	0	0
20		0	0	0	0	0	0	0	0	0

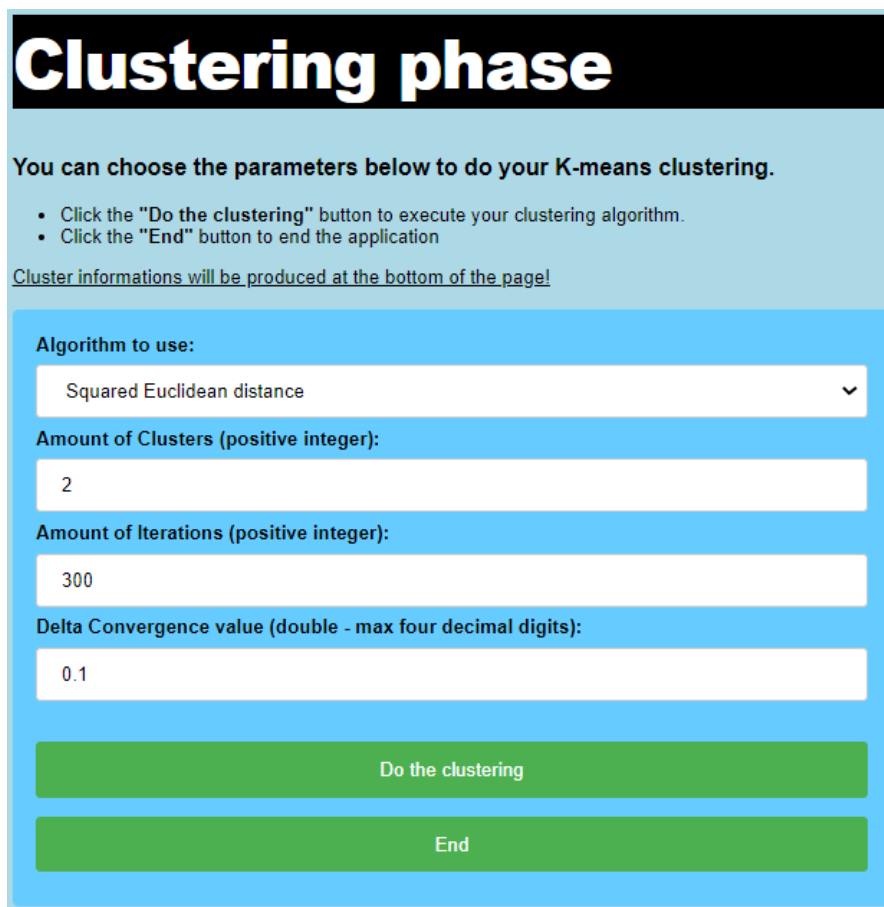
### Εικόνα 25: Δεύτερο Preprocessing Βήμα

### 3.2.7. Clustering phase

Στην φάση αυτή, παρουσιάζεται στον χρήστη, μία σελίδα με μία φόρμα που καλείτε να συμπληρώσει (στην **Εικόνα 26** φαίνεται η φόρμα συμπληρωμένη). Εκτός από την φόρμα, ο χρήστης κατέχει πλέον και ένα κουμπί με όνομα «End» το οποίο πατώντας το, τον οδηγεί στην λήξη της εφαρμογής (**Ενότητα 3.2.8**).

Η φόρμα, ζητάει από τον αναλυτή να δώσει τις παραμέτρους ενός K-means αλγορίθμου. Αφού συμπληρώσει όλα τα πεδία, μπορεί να πατήσει το κουμπί «Do the Clustering» και θα εκτελεστεί η συσταδοποίηση των προεπεξεργασμένων δεδομένων του. Τα αποτελέσματα αυτής, θα τοποθετηθούν με οργανωμένο τρόπο στο HDFS, με σκοπό ο χρήστης να μπορεί να εκτελέσει πολλά πειράματα, τα οποία θα συγκεντρώνονται στη αποθήκη δεδομένων, στους αντίστοιχους καταλόγους του Topic του. Ο Χρήστης μέχρι να πατήσει το κουμπί «End» θα μπορεί να κάνει όσα πειράματα επιθυμεί.

Αφού αποθηκευτούν τα αποτελέσματα στο HDFS, εμφανίζονται στην διεπαφή της εφαρμογής (**Εικόνα 27**). Αυτά τα αποτελέσματα θα βρίσκονται κάτω από την φόρμα για κάθε πείραμα που πραγματοποιεί ο χρήστης. Συνεπώς μπορούν να βρεθούν μετά από scroll down. Επίσης, η εφαρμογή παρέχει στον αναλυτή δεδομένων το νέο σύνδεσμο στον οποίο αποθηκεύει τα δεδομένα των clustering για κάθε πείραμα (**Εικόνα 28**). Στην **Εικόνα 29**, εμφανίζονται οι κατάλογοι του HDFS μετά από έναν αριθμό συνεχόμενων πειραμάτων, ώστε να φαίνεται ο τρόπος με τον οποίο οργανώνει η εφαρμογή τα πειράματα εντός της αποθήκης δεδομένων.



**Clustering phase**

You can choose the parameters below to do your K-means clustering.

- Click the "Do the clustering" button to execute your clustering algorithm.
- Click the "End" button to end the application

[Cluster informations will be produced at the bottom of the page!](#)

Algorithm to use:  
Squared Euclidean distance

Amount of Clusters (positive integer):  
2

Amount of Iterations (positive integer):  
300

Delta Convergence value (double - max four decimal digits):  
0.1

Do the clustering

End

[Εικόνα 26: Clustering phase](#)

End

## Below are the results for your clustering!




Those results have been saved in the HDFS directory below:  
[Clustering number 1](#)  
 You can do as many clustering tests you want and the results will be printed here and saved always in the above directory.  
 (for every new test, the above link will change to the new HDFS directory for your test).

**Results:**







```
{ "identifier": "VL-911", "r": [{"1": 0.087}, {"6": 0.256}, {"7": 0.431}, {"8": 0.071}, {"9": 0.092}], "c": [{"1": 0.007}, {"6": 0.068}, {"7": 0.235}, {"8": 0.005}, {"9": 0.009}], "n": 1987 }
Weight : [props - optional]: Point:
1.0 : [distance=0.05975909339240545]: []
1.0 : [distance=0.05975909339240545]: []
1.0 : [distance=0.05975909339240545]: []
1.0 : [distance=0.05975909339240545]: []
1.0 : [distance=0.9238758523254704]: [{"6": 1.0}]
1.0 : [distance=0.9238758523254704]: [{"6": 1.0}]
1.0 : [distance=0.590710276079874]: [{"7": 1.0}]
1.0 : [distance=0.05975909339240545]: []
1.0 : [distance=0.05975909339240545]: []
1.0 : [distance=0.590710276079874]: [{"7": 1.0}]
1.0 : [distance=0.05975909339240545]: []
1.0 : [distance=0.9238758523254704]: [{"6": 1.0}]
1.0 : [distance=0.05975909339240545]: []
1.0 : [distance=0.05975909339240545]: []
1.0 : [distance=0.590710276079874]: [{"7": 1.0}]
1.0 : [distance=0.590710276079874]: [{"7": 1.0}]
1.0 : [distance=0.590710276079874]: [{"7": 1.0}]
1.0 : [distance=0.05975909339240545]: []
```

Εικόνα 27: Αποτελέσματα Clustering

## Browse Directory

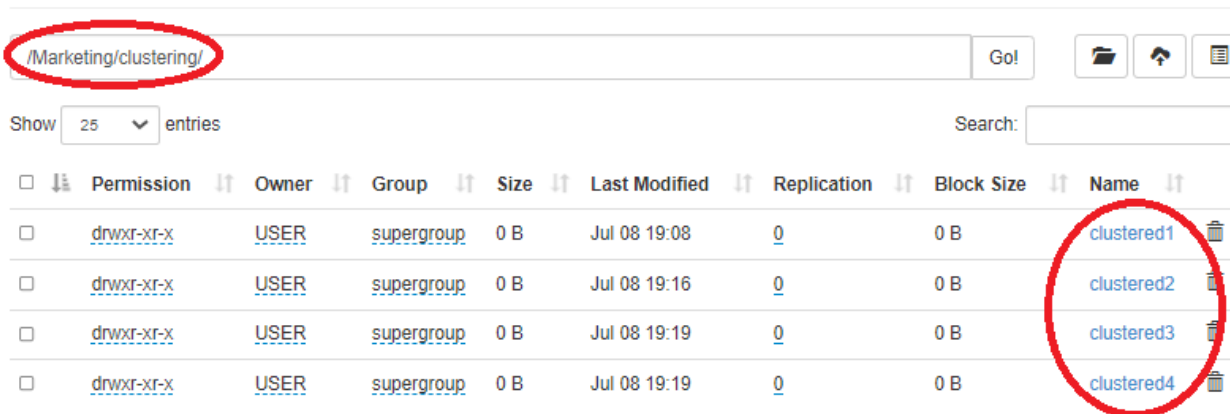
[/Marketing/clustering/clustered1](#) Go!   

Show  entries Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	USER	supergroup	0 B	Jul 08 19:07	0	0 B	KmeansOutputData	
<input type="checkbox"/>	-rw-r--r--	USER	supergroup	194 B	Jul 08 19:08	3	128 MB	_policy	
<input type="checkbox"/>	drwxr-xr-x	USER	supergroup	0 B	Jul 08 19:08	0	0 B	clusteredPoints	
<input type="checkbox"/>	drwxr-xr-x	USER	supergroup	0 B	Jul 08 19:08	0	0 B	clusters-0	
<input type="checkbox"/>	drwxr-xr-x	USER	supergroup	0 B	Jul 08 19:08	0	0 B	clusters-1-final	
<input type="checkbox"/>	drwxr-xr-x	USER	supergroup	0 B	Jul 08 19:08	0	0 B	random-seeds	

Εικόνα 28: Clustering Δεδομένα

## Browse Directory



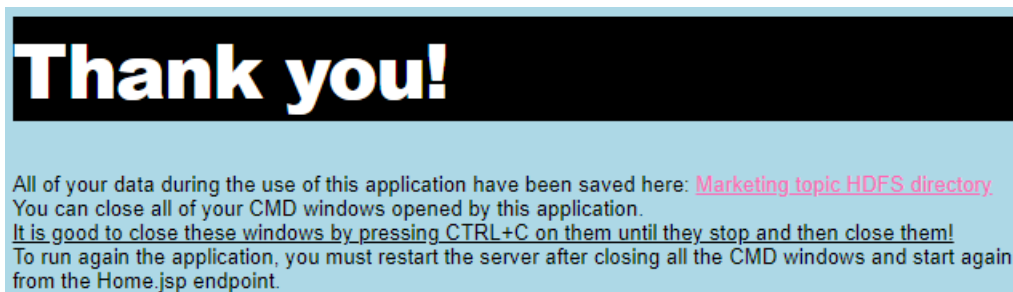
The screenshot shows the HDFS Browse Directory interface. The path `/Marketing/clustering/` is entered in the address bar and circled in red. Below the address bar, there is a "Go!" button and icons for file operations. A "Show" dropdown is set to "25" entries. A search bar is present on the right. The main area displays a table of files with columns: Permission, Owner, Group, Size, Last Modified, Replication, Block Size, and Name. The files listed are `clustered1`, `clustered2`, `clustered3`, and `clustered4`, all owned by `USER` and `supergroup`. The names are circled in red.

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
<code>drwxr-xr-x</code>	<code>USER</code>	<code>supergroup</code>	0 B	Jul 08 19:08	0	0 B	<code>clustered1</code>
<code>drwxr-xr-x</code>	<code>USER</code>	<code>supergroup</code>	0 B	Jul 08 19:16	0	0 B	<code>clustered2</code>
<code>drwxr-xr-x</code>	<code>USER</code>	<code>supergroup</code>	0 B	Jul 08 19:19	0	0 B	<code>clustered3</code>
<code>drwxr-xr-x</code>	<code>USER</code>	<code>supergroup</code>	0 B	Jul 08 19:19	0	0 B	<code>clustered4</code>

[Εικόνα 29: Εμφάνιση οργάνωσης της εφαρμογής στο HDFS](#)

### 3.2.8. End phase

Όταν ο χρήστης προχωρήσει στην τελική φάση, εμφανίζεται μία διεπαφή (**Εικόνα 30**) στην οποία ο χρήστης μπορεί να διαβάσει τον τρόπο με οποίο μπορεί να εκτελέσει την εφαρμογή για να ξανακάνει ανάλυση. Τέλος, η διεπαφή αυτή παρέχει έναν σύνδεσμο στον αναλυτή για ολόκληρο τον HDFS κατάλογο που έχουν αποθηκευτεί τα δεδομένα του (**Εικόνα 31**).

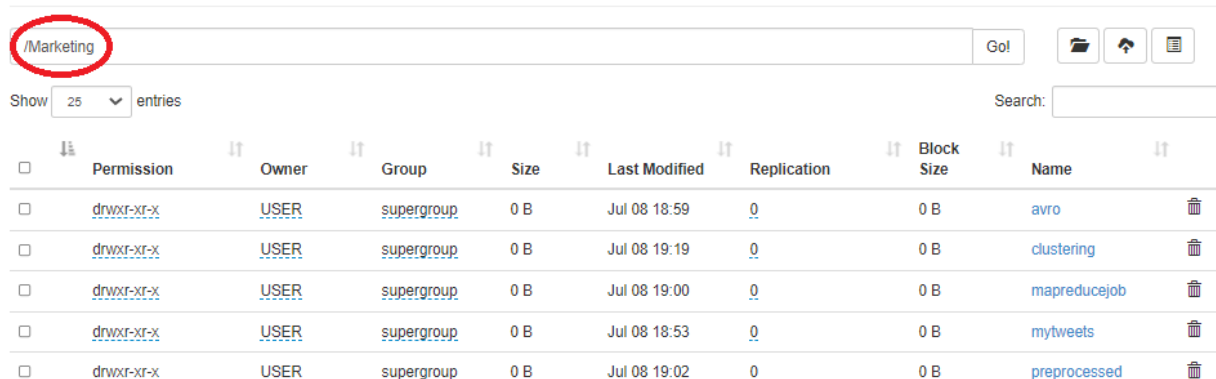


The screenshot shows a "Thank you!" message in a black box. Below it, a light blue box contains the following text:

All of your data during the use of this application have been saved here: [Marketing topic HDFS directory](#). You can close all of your CMD windows opened by this application. It is good to close these windows by pressing CTRL+C on them until they stop and then close them! To run again the application, you must restart the server after closing all the CMD windows and start again from the Home.jsp endpoint.

[Εικόνα 30: End phase](#)

## Browse Directory



The screenshot shows the HDFS Browse Directory interface. The path `/Marketing` is entered in the address bar and circled in red. Below the address bar, there is a "Go!" button and icons for file operations. A "Show" dropdown is set to "25" entries. A search bar is present on the right. The main area displays a table of files with columns: Permission, Owner, Group, Size, Last Modified, Replication, Block Size, and Name. The files listed are `avro`, `clustering`, `mapreducejob`, `mytweets`, and `preprocessed`, all owned by `USER` and `supergroup`.

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
<code>drwxr-xr-x</code>	<code>USER</code>	<code>supergroup</code>	0 B	Jul 08 18:59	0	0 B	<code>avro</code>
<code>drwxr-xr-x</code>	<code>USER</code>	<code>supergroup</code>	0 B	Jul 08 19:19	0	0 B	<code>clustering</code>
<code>drwxr-xr-x</code>	<code>USER</code>	<code>supergroup</code>	0 B	Jul 08 19:00	0	0 B	<code>mapreducejob</code>
<code>drwxr-xr-x</code>	<code>USER</code>	<code>supergroup</code>	0 B	Jul 08 18:53	0	0 B	<code>mytweets</code>
<code>drwxr-xr-x</code>	<code>USER</code>	<code>supergroup</code>	0 B	Jul 08 19:02	0	0 B	<code>preprocessed</code>

[Εικόνα 31: Topic HDFS κατάλογος](#)

## 4. Αποτελέσματα Εφαρμογής

Η Ενότητα αυτή, βασίζεται στις διαδικασίες που πραγματοποιήθηκαν στην **Ενότητα 3.2** και αφορά το πρόβλημα το οποίο αναπτύχθηκε στην **Ενότητα 1**. Θα αποδοθεί μία καλύτερη «ματιά» στα αποτελέσματα και στα δεδομένα τα οποία συλλέχτηκαν, καθώς επίσης θα εξηγηθούν μερικά από τα χρησιμοποιημένα τμήματα των εργαλείων που αξιοποιήθηκαν.

#### 4.1. Keywords και συλλεχθέντα δεδομένα

Έγινε επιλογή 10 λέξεων-κλειδιών, οι οποίες είναι σχετικές με συχνά αγοραζόμενα προϊόντα ανά τον κόσμο τα τελευταία χρόνια. Αυτές οι λέξεις είναι οι εξής:

- doormat
- headphones
- towel
- saddle
- bookcase
- bookend
- ball
- book
- vacuum
- airpods

Συλλέχθηκαν συνολικά 2000 tweets και στην **Εικόνας 32**, φαίνονται τρία από αυτά. Το περιεχόμενο αυτών των δεδομένων, είναι πολύ μεγάλο και δυστυχώς δεν μπορεί να αναπαρασταθεί ολόκληρο για κάθε tweet.

[illegible]

### Εικόνα 32: Συλλεχθέντα δεδομένα



## 4.2. Flume Agent configuration file

Ο Flume Agent (με όνομα KafkaAgent) της εφαρμογής κατασκευάζεται αυτόματα, ανάλογα το Topic το οποίο θέλει να ασχοληθεί ο χρήστης και στην συνέχεια τοποθετείται στον φάκελο του %FLUME\_HOME% με σκοπό να εκτελεστεί για την συλλογή των δεδομένων. Παρακάτω, βλέπουμε τον κώδικα που υπάρχει στο Flume Agent που δημιουργείται κάθε φορά, όπως επίσης φαίνονται με **κόκκινο χρώμα** οι μεταβαλλόμενες τιμές ανά Flume Agent.

```
KafkaAgent.sources = source1
```

```
KafkaAgent.channels = channel1
```

```
KafkaAgent.sinks = sink1
```

```
KafkaAgent.sources.source1.type = org.apache.flume.source.kafka.KafkaSource
```

```
KafkaAgent.sources.source1.kafka.bootstrap.servers = localhost:9092
```

```
KafkaAgent.sources.source1.kafka.topics = Marketing
```

```
KafkaAgent.sources.source1.kafka.consumer.group.id = flume
```

```
KafkaAgent.sources.source1.channels = channel1
```

```
KafkaAgent.sources.source1.interceptors = i1
```

```
KafkaAgent.sources.source1.interceptors.i1.type = timestamp
```

```
KafkaAgent.sources.source1.kafka.consumer.timeout.ms = 100
```

```
KafkaAgent.channels.channel1.type = memory
```

```
KafkaAgent.channels.channel1.capacity = 10000
```

```
KafkaAgent.channels.channel1.transactionCapacity = 1000
```

```
KafkaAgent.sinks.sink1.type = hdfs
```

```
KafkaAgent.sinks.sink1.hdfs.path = hdfs://localhost:9000/Marketing/mytweets
```

```
KafkaAgent.sinks.sink1.hdfs.rollInterval = 0
```

```
KafkaAgent.sinks.sink1.hdfs.rollSize = 0
```

```
KafkaAgent.sinks.sink1.hdfs.rollCount = 0
```

```
KafkaAgent.sinks.sink1.hdfs.fileType = DataStream
```

```
KafkaAgent.sinks.sink1.channel = channel1
```

Όπως φαίνεται στον κώδικα παραπάνω ο Agent αποτελείται από μία πηγή (source1), ένα κανάλι (channel1) και έναν αποδέκτη δεδομένων (sink1).

Στον Agent, η πηγή συνδέεται με το Topic «Marketing» του Kafka, το κανάλι κατοχυρώνει ένα κομμάτι της μνήμης του μηχανήματος για την λειτουργία του, ενώ ο αποδέκτης δεδομένων είναι μία αποθήκη δεδομένων HDFS στο μονοπάτι «hdfs://localhost:9000/Marketing/mytweets». Ο κώδικας αυτός βάζει όλα τα δεδομένα που συλλέγονται σε ένα μόνο αρχείο εντός του HDFS, με σκοπό τα δεδομένα να είναι οργανωμένα και μαζεμένα σε ένα σημείο.

### 4.3. AVRO Schema

Για την κωδικοποίηση και αποκωδικοποίηση των συλλεχθέντων δεδομένων, χρησιμοποιήθηκε το παρακάτω AVRO Schema [\[9\]](#) (Το οποίο τοποθετείτε από την εφαρμογή αυτόματα στο μονοπάτι «C:\TwitterAnalyzer»).

```
{
  "type" : "record",
  "name" : "Tweet",
  "namespace" : "com.miguno.avro",
  "fields" : [ {
    "name" : "username",
    "type" : "string",
    "doc" : "Name of the user account on Twitter.com"
  }, {
    "name" : "tweet",
    "type" : "string",
    "doc" : "The content of the user's Twitter message"
  }, {
    "name" : "timestamp",
    "type" : "long",
    "doc" : "Unix epoch time in seconds"
  } ],
  "doc" : "A basic schema for storing Twitter messages"
}
```

Το παραπάνω σχήμα θεωρεί κάθε tweet ως ένα record και αποθηκεύει τρεις ιδιότητές του. Το όνομα χρήστη του tweet (username) ως αλφαριθμητικό, το περιεχόμενο που υπάρχει εντός του tweet αυτού (tweet) ως αλφαριθμητικό και το timestamp του κάθε tweet (timestamp) ως έναν αριθμό μεγάλου μεγέθους.

#### 4.4. Αποτελέσματα MapReduce

Όπως παρουσιάστηκε και στην **Εικόνα 21**, τα αποτελέσματα της Map Reduce διαδικασίας ήταν τα παρακάτω:

Λέξη	Πλήθος εμφανίσεων στα 2000 tweets
airpods	65
ball	191
book	607
doormat	1
headphones	15
towel	14
vacuum	19

Τα αποτελέσματα, για την καλύτερη οπτικοποίησή τους, φαίνονται στον παρακάτω πίνακα ταξινομημένα, μαζί με τις υπόλοιπες λέξεις οι οποίες δεν εμφανίστηκαν σε κανένα tweet.

Λέξη	Πλήθος εμφανίσεων στα 2000 tweets
book	607
ball	191
airpods	65
vacuum	19
headphones	15
towel	14
doormat	1
saddle	0
bookend	0
bookcase	0

Όπως φαίνεται παραπάνω υπάρχει ένα μεγαλύτερο ενδιαφέρον στο κοινό για βιβλία όπως και μπάλες, στην συνέχεια για ηλεκτρικά ήδη μόδας όπως airpods και μετά είδη καθαρισμού σπιτιού όπως σκούπες. Τα υπόλοιπα είδη εμφανίστηκαν από λίγες έως και καμία φορά, οπότε δεν ενδιαφέρουν τόσο το κοινό.



#### 4.5. Προ-επεξεργασία δεδομένων

Όπως φαίνεται στις **Εικόνες 24 και 25**, στην προεπεξεργασία των συλλεχθέντων δεδομένων έλαβαν μέρος τα παρακάτω βήματα:

- Συγκράτηση της πληροφορίας μόνο του περιεχόμενου κάθε tweet, χωρίς το όνομα χρήστη και το timestamp του.
- Αρίθμηση του πλήθους εμφάνισης της κάθε λέξης κλειδί για κάθε tweet και δημιουργία ενός αρχείου-πίνακα με την πληροφορία αυτήν.

Συνεπώς, από την αποκωδικοποιημένη AVRO πληροφορία, έγινε χρήση μόνο της ιδιότητας «tweet» από κάθε record, η οποία βρίσκεται στο πεδίο «text» του κάθε json αντικειμένου που παρέχει το Twitter API.

#### 4.6. Ανάλυση συλλεχθέντων δεδομένων

Για τα πειράματα της ανάλυσης των δεδομένων, κάθε πείραμα παίρνει ένα αρχείο του οποίου η μορφή φαίνεται στην **Εικόνα 33**. Το αρχείο που δίνεται ως είσοδος σε κάθε συσταδοποίηση, παράγεται από τον πίνακα που έχει κατασκευαστεί στην φάση της προεπεξεργασίας (**Εικόνα 25**).

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0
```

**Εικόνα 33: Preprocessed αρχείο**

Εκτός από το προαναφερθέν αρχείο, δίνονται και παράμετροι για τον K-means clustering αλγόριθμο κατευθείαν από τον αναλυτή. Στην ανάλυση των πειραμάτων που πραγματοποιήθηκαν, οι παράμετροι σε κάθε πείραμα (4 συνολικά πειράματα) ήταν οι εξής:

<b>Πείραμα 1</b>	
<b>Παράμετροι</b>	<b>Τιμές</b>
<b>Μέτρο απόστασης</b>	<b>Τετραγωνική Ευκλείδεια απόσταση</b>
<b>Πλήθος συστάδων</b>	<b>2</b>
<b>Αριθμός επαναλήψεων</b>	<b>300</b>
<b>Σύγκλιση Delta</b>	<b>0.1</b>

<b>Πείραμα 2</b>	
<b>Παράμετροι</b>	<b>Τιμές</b>
<b>Μέτρο απόστασης</b>	<b>Τετραγωνική Ευκλείδεια απόσταση</b>
<b>Πλήθος συστάδων</b>	<b>3</b>
<b>Αριθμός επαναλήψεων</b>	<b>300</b>
<b>Σύγκλιση Delta</b>	<b>0.1</b>

<b>Πείραμα 3</b>	
<b>Παράμετροι</b>	<b>Τιμές</b>
<b>Μέτρο απόστασης</b>	<b>Συνημιτονοειδής απόσταση</b>
<b>Πλήθος συστάδων</b>	<b>4</b>
<b>Αριθμός επαναλήψεων</b>	<b>300</b>
<b>Σύγκλιση Delta</b>	<b>0.1</b>

<b>Πείραμα 4</b>	
<b>Παράμετροι</b>	<b>Τιμές</b>
<b>Μέτρο απόστασης</b>	<b>Συνημιτονοειδής απόσταση</b>
<b>Πλήθος συστάδων</b>	<b>2</b>
<b>Αριθμός επαναλήψεων</b>	<b>300</b>
<b>Σύγκλιση Delta</b>	<b>0.1</b>

Μετά από κάθε πείραμα, εκτός από την αποθήκευση των αποτελεσμάτων στο HDFS (**Εικόνες 28 και 29**), γίνεται και παρουσίαση αυτών, κατευθείαν στην διεπαφή που βλέπει ο χρήστης (**Εικόνα 27**). Στην ανάλυση των πειραμάτων που πραγματοποιήθηκαν παρακάτω, φαίνονται τα αποτελέσματα τους. (Το πλήθος δεδομένων στον αλγόριθμο του Mahout πρέπει να προσθέτει ένα σημείο για κάθε Cluster και για αυτό τα δεδομένα προσμετρούνται στην τιμή <σύνολο tweets> + <σύνολο συστάδων>).

### Πείραμα 1

<b>Συστάδα 1</b>	
<b>Κέντρο</b>	<b>{"1":0.007},{ "6":0.068},{ "7":0.235},{ "8":0.005},{ "9":0.009}</b>
<b>Ακτίνα</b>	<b>{"1":0.087},{ "6":0.256},{ "7":0.431},{ "8":0.071},{ "9":0.092}</b>
<b>Πλήθος δεδομένων</b>	<b>1987</b>
<b>Συστάδα 2</b>	
<b>Κέντρο</b>	<b>{"2":1.0}</b>
<b>Ακτίνα</b>	<b>-</b>
<b>Πλήθος δεδομένων</b>	<b>15</b>

### Πείραμα 2

<b>Συστάδα 1</b>	
<b>Κέντρο</b>	<b>{"1":0.007},{ "2":0.007},{ "7":0.249},{ "8":0.005},{ "9":0.009}</b>
<b>Ακτίνα</b>	<b>{"1":0.089},{ "2":0.086},{ "7":0.44},{ "8":0.073},{ "9":0.095}</b>
<b>Πλήθος δεδομένων</b>	<b>1868</b>
<b>Συστάδα 2</b>	
<b>Κέντρο</b>	<b>-</b>
<b>Ακτίνα</b>	<b>-</b>
<b>Πλήθος δεδομένων</b>	<b>1</b>
<b>Συστάδα 3</b>	
<b>Κέντρο</b>	<b>{"6":1.015}</b>
<b>Ακτίνα</b>	<b>{"6":0.121}</b>
<b>Πλήθος δεδομένων</b>	<b>134</b>

**Πείραμα 3**

<b>Συστάδα 1</b>	
<b>Κέντρο</b>	{ "1":0.006},{ "2":0.007},{ "6":0.067},{ "7":0.233},{ "8":0.005},{ "9":0.008}
<b>Ακτίνα</b>	{ "1":0.086},{ "2":0.083},{ "6":0.255},{ "7":0.43},{ "8":0.071},{ "9":0.092}
<b>Πλήθος δεδομένων</b>	2001
<b>Συστάδα 2</b>	
<b>Κέντρο</b>	-
<b>Ακτίνα</b>	-
<b>Πλήθος δεδομένων</b>	1
<b>Συστάδα 3</b>	
<b>Κέντρο</b>	-
<b>Ακτίνα</b>	-
<b>Πλήθος δεδομένων</b>	1
<b>Συστάδα 4</b>	
<b>Κέντρο</b>	{ "6":1.0}
<b>Ακτίνα</b>	-
<b>Πλήθος δεδομένων</b>	1

**Πείραμα 4**

<b>Συστάδα 1</b>	
<b>Κέντρο</b>	{ "1":0.007},{ "2":0.007},{ "6":0.068},{ "7":0.235},{ "9":0.009}
<b>Ακτίνα</b>	{ "1":0.087},{ "2":0.084},{ "6":0.255},{ "7":0.431},{ "9":0.092}
<b>Πλήθος δεδομένων</b>	1991
<b>Συστάδα 2</b>	
<b>Κέντρο</b>	{ "8":1.0}
<b>Ακτίνα</b>	-
<b>Πλήθος δεδομένων</b>	11

Στα πειράματα 3 και 4, η απόφαση για Συνημιτονοειδή απόσταση δεν ήταν καλή. Σχεδόν όλα τα δεδομένα έχουν μαζευτεί σε ένα cluster και συνεπώς δεν μπορούν να παραχθούν χρήσιμα συμπεράσματα. Το πείραμα 1 ένα και 2 έχουν όμως ένα ενδιαφέρον. Στο 1, φαίνεται ότι υπάρχει κάποια διαχώριση σε δύο clusters, ενώ στο 2, παρουσιάζεται ότι παρόλο που ο αναλυτής ζήτησε να γίνει διάσπαση σε τρεις συστάδες, ο K-means κατέληξε ουσιαστικά μόνο με 2. Μάλιστα η τρίτη συστάδα φαίνεται να περιέχει μέσα της και ένα αρκετά «καλό» πλήθος από δεδομένα, τα οποία είναι γύρω από την 6<sup>η</sup> διάσταση του συνόλου που είναι η λέξη ball (ξεκινώντας από την αρίθμηση από το μηδέν). Συμπεραίνεται λοιπόν ότι το κοινό, δείχνει ένα ενδιαφέρον για την λέξη μπάλα, την οποία δεν την συσχετίζει σε αρκετές περιπτώσεις με προϊόντα τα οποία υπάρχουν στην αγορά. Αυτό μπορεί να συμβαίνει λόγω του αθλήματος του ποδοσφαίρου, το οποίο έχει ένα πολύ μεγάλο κοινό ανά τον κόσμο. Ένας καταστηματάρχης θα επωφεληθεί από την παραπάνω ανάλυση, αυξάνοντας τις τιμές για προϊόντα που σχετίζονται με αθλήματα όπως μπάλες ποδοσφαίρου με μπλουζές παικτών κάποιες αθλητικής ομάδας.

## 5. Επόμενα Βήματα

- Αυτοματοποίηση Flume Agent, ώστε να μην υπάρχει λόγος συγκεκριμένης ονομασίας και τοποθεσίας του Flume καταλόγου.
- Εύρεση τρόπου διαχείρισης λειτουργιών της εφαρμογής, χωρίς την δημιουργία και χρήση επιπλέον καταλόγου εντός του C drive, του υπολογιστή του χρήστη.
- Καλύτερη έκδοση της εφαρμογής ως προς την εγκατάστασή της. Με σκοπό να είναι πιο φιλική στον τελικό χρήστη.
- Προσθήκη ικανότητας της εφαρμογής να μην χρειάζεται να τρέχει μόνο προς τις επόμενες σελίδες. Θα μπορεί ο χρήστης να επιλέγει το topic που έχει αποθηκεύσει σε προηγούμενη χρήση της εφαρμογής, ώστε να το ξανά-αναλύσει.
- Καλύτερη χρήση Avro Serialization των αρχείων με σκοπό να υπάρχει νόημα της χρήσης αυτής της μορφοποίησης.
- Απόδοση ικανότητας στον χρήστη να επιλέγει το πως θα είναι το AVRO schema των δεδομένων που συλλέγει.
- Διατήρηση μεγαλύτερου περιεχομένου κάθε tweet, με σκοπό την πιο ουσιώδη προεπεξεργασία των δεδομένων και εύρεση καλύτερων και πιο στοχευμένων αποτελεσμάτων.
- Βελτίωση της διεπαφής του Preprocessing phase.
- Βελτίωση του στίλ των διεπαφών εν γένει.
- Αύξηση των βημάτων προεπεξεργασίας.
- Απόδοση ικανότητας στον χρήστη να επιλέγει τα βήματα προεπεξεργασίας των δεδομένων του.
- Απόδοση ικανότητας επιλογής στον χρήστη του Map Reduce αλγορίθμου.
- Απόδοση ικανότητας στον χρήστη πολλαπλής επιλογής αλγορίθμων για το clustering των δεδομένων του εκτός από τον K-means.
- Αποθήκευση των δεδομένων που παρουσιάζονται στο clustering του χρήστη και στο HDFS, καθώς το mahout αποθηκεύει στο HDFS αποτελέσματα σε μορφή η οποία δεν είναι αναγνώσιμη.
- Προσθήκη γραφικής οπτικοποίησης των αποτελεσμάτων του Clustering.
- Απόδοση ικανότητας τροποποίησης του περιεχομένου HDFS κατευθείαν μέσα από την εφαρμογή.
- Προσθήκη λογαριασμών στην εφαρμογή, ώστε ο κάθε αναλυτής να μπορεί να βλέπει μόνο την δικιά του ανάλυση.
- Απόδοση ικανότητας παραμετροποίησης, με όμορφο οπτικά τρόπο, του Flume Agent από τον χρήστη.

## 6. Αναφορές

- 1) [Apache Hadoop](#)
- 2) [Apache Hadoop HDFS](#)
- 3) [Apache Hadoop YARN](#)
- 4) [Apache Hadoop MapReduce](#)
- 5) [Apache AVRO](#)
- 6) [Apache Flume](#)
- 7) [Apache Kafka](#)
- 8) [Apache Mahout](#)
- 9) [AVRO schema](#)