Τεχνική Αναφορά Εργασίας

Βαρβούτας Κωνσταντίνος (2336)

Μουλαδένιος Κωνσταντίνος (2379)

Αθανασιάδης Γεώργιος (2331)

ΜΕΡΟΣ 1 : Συλλογή Δεδομένων

Μέσω του twitter streaming API λαμβάνουμε τα επίκαιρα tweets των χρηστών. Δημιουργούμε έναν λογαριασμό στο twitter αν δεν έχουμε και στο Application Manager του λογαριασμού μας δημιουργούμε ένα καινούριο app (create new app). Αυτό το κάνουμε για να αποκτήσουμε τα keys με τα οποία θα μπορέσουμε να κάνουμε streaming τα tweets των χρηστών.

Στο επόμενο στάδιο δημιουργούμε μία main class (TwitterApp.java)στο netbeans η οποία θα λαμβάνει τα tweets μέσω streaming και θα τα αποθηκεύει σε μία βάση δεδομένων. Χρησιμοποιούμε το twitter4j , μία βιβλιοθήκη java η οποία μας βοηθάει να υλοποιήσουμε αυτές τις λειτουργίες. Μέσω λοιπόν ενός ConfigurationBuilder εισάγουμε τα κλειδιά που αποκτήσαμε προηγουμένως και εγκαθιστάμε μία σύνδεση. Επιπλέον βρίσκουμε το top trend της ημέρας μέσω του twitter4j και κάνουμε streaming των tweets με βάση τη αυτή τη λέξη κλειδί . Για την αποθήκευση των tweets σε μορφή json χρησιμοποιούμε τη βάση MongoDB και εισάγουμε ένα – ένα τα tweets σε μία collection της βάσης . Τα αποθηκευμένα tweets έχουν την εξής μορφή :

```
_id: ObjectID('584a98000ecf481c4474f7df')
▸ extended_entities: Object
  in_reply_to_status_id_str: null
  in_reply_to_status_id: null
  created_at: "Fri Dec 09 11:39:43 +0000 2016"
  in_reply_to_user_id_str: null
  source: "<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>"
▸ retweeted_status: Object
  retweet_count: 0
  retweeted: false
  geo: null
  filter_level: "low"
  in_reply_to_screen_name: null
  is_quote_status: false
  id_str: "807187969039880193"
  in_reply_to_user_id: null
  favorite_count: 0
  id: 807187969039880193
  text: "RT @_theboulron: I'm just leave this here... #4YourEyezOnly https://t.co/MjGKO0SqUu"
  place: null
  lang: "en"
  favorited: false
  possibly_sensitive: false
  coordinates: null
  truncated: false
  timestamp_ms: "1481283583303"
▸ entities: Object
  contributors: null
▸ user: Object
```

Τρέχουμε τον κώδικα αυτό 4 φορές σε διαφορετικές χρονικές στιγμές και δημιουργούμε 4 διαφορετικές collections (myTweetCol, myTweetCol2, myTweetCol3, myTweetCol4) έτσι ώστε να έχουμε πληθώρα διαφορετικών tweets.

ΜΕΡΟΣ 2 : Ανάλυση Δεδομένων

Δημιουργούμε μία νέα main κλάση (TweetsProcessing.java) . Για κάθε συλλογή που έχουμε αποθηκευμένη στην κλάση θα δημιουργήσουμε μία ξεχωριστή συλλογή η οποία θα περιέχει για κάθε tweet τις οντότητες που αναγνωρίσαμε ότι υπάρχουν, όπως για παράδειγμα user, hashtag, mentions, retweets (Συλλογές : separateEntities, separateEntities2, separateEntities3, separateEntities4) . Η δομή των documents σε αυτή τη συλλογή είναι ως εξής :

```
_id: ObjectID('5877a08b21eaa61bf07c2a54')
user: "DirtGotNoHoes"
timestamp: "1481283583303"
hashtag: "4YourEyezOnly"
```

```
_id: ObjectID('5877a08b21eaa61bf07c2a55')
user: "DirtGotNoHoes"
timestamp: "1481283583303"
url: "https://twitter.com/_theboulron/status/807120061177675777/photo/1 "
```

```
_id: ObjectID('5877a08b21eaa61bf07c2a56')
user: "DirtGotNoHoes"
timestamp: "1481283583303"
mentioned_users: "_theboulron "
```

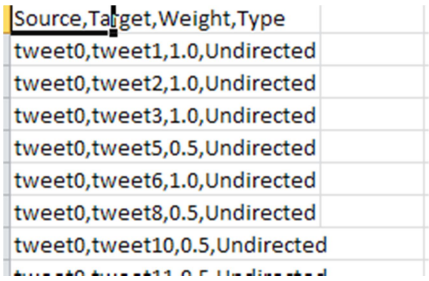
Αυτό το επιτυγχάνουμε με διαχωρισμό του κειμένου στα αντίστοιχα tokens και έτσι προκύπτει η αναγνώριση των οντοτήτων που περιέχει το Tweet. Αυτές οι οντότητες αποθηκεύονται στην ίδια βάση (mongodb) για να διατηρήσουμε πάλι τη μορφή αναπαράστασης json. Στη συνέχεια από αυτές τις συλλογές επαναληπτικά για κάθε συλλογή παίρνουμε τα έγγραφα που περιέχουν hashtag και τα αποθηκεύουμε σε ξεχωριστή συλλογή hashtagAll, το ίδιο κάνουμε και για τα mentioned(mentionedAll), τα url (urlAll) και τα retweeted tweets(retweetedAll). Λόγω μειωμένης επεξεργαστικής δύναμης και για να μπορέσουμε να φορτώσουμε αργότερα τα αποτελέσματα στο gephi οι συλλογές αυτές περιέχουν μόνο 250 από την κάθε συλλογή . Για παράδειγμα η hashtagAll περιέχει 250 documents από το separateEntities, 250 από το seperateEntities2, 250 από το

seperateEntities3 και 250 από το seperateEntities4 άρα συνολικά περιέχει 1000 documents με hashtag από διάφορα Tweets και το ίδιο γίνεται και για τις υπόλοιπες συλλογές . Αυτός ο διαχωρισμός γίνεται με αποτέλεσμα να διευκολύνουμε τη διαδικασία δημιουργίας πινάκων ομοιότητας.

ΜΕΡΟΣ 3 : Υπολογισμός Ομοιότητας

Σε αυτή τη φάση παίρνουμε τα έγγραφα από τις συλλογές hashtagAll, mentionedAll, urlAll και retweetedAll που δημιουργήσαμε ακριβώς για αυτό το σκοπό και υπολογίζουμε για κάθε χαρακτηριστικό τον πίνακα ομοιότητας του. Αφαιρούμε όμως τις οντότητες που εμφανίζονται σε παραπάνω από το 50% των tweets το οποίο θα βελτιώσει την ακρίβεια των αποτελεσμάτων. Επίσης δεν εισάγουμε τα tweets για τα οποία η ομοιότητα είναι μικρότερη από 0.2 . Για τον υπολογισμό της ομοιότητας των tweets (ως προς hashtag, url κλπ) χρησιμοποιούμε το μέτρο ομοιότητας jaccard για τον οποίο βρήκαμε πληροφορίες στη wikipedia (https://en.wikipedia.org/wiki/Jaccard_index).

Κάθε πίνακας εξάγεται σε ένα αρχείο csv . Επίσης δημιουργούμε έναν πίνακα στον οποίο συνδυάζονται τα παραπάνω αποτελέσματα σε ένα συνολικό μέτρο ομοιότητας. Η δομή των πινάκων είναι η εξής:



Source,Target,Weight,Type
tweet0,tweet1,1.0,Undirected
tweet0,tweet2,1.0,Undirected
tweet0,tweet3,1.0,Undirected
tweet0,tweet5,0.5,Undirected
tweet0,tweet6,1.0,Undirected
tweet0,tweet8,0.5,Undirected
tweet0,tweet10,0.5,Undirected

Όπου περιγράφεται η συσχέτιση των tweets και η μεταξύ τους ομοιότητα (ως βάρος ακμής)

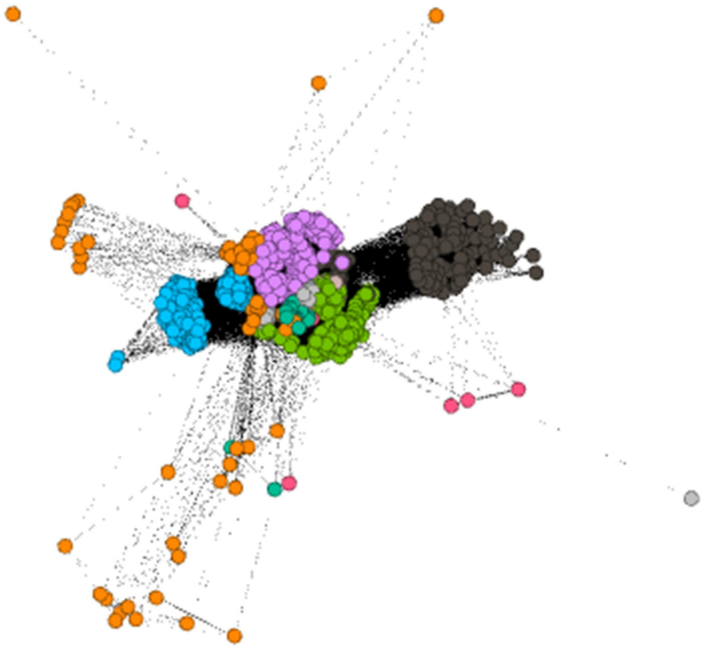
ΜΕΡΟΣ 4 : Gephi

Εισάγουμε τους πίνακες ομοιότητας για κάθε χαρακτηριστικό στο gephi . Πηγαίνουμε στο Overview και βλέπουμε τον γράφο που έχει δημιουργήσει. Στη συνέχεια τρέχουμε τους αλγορίθμους για να υπολογίσουμε το density του γράφου , το Average Clustering Coefficient και το diameter. Το density είναι η αναλογία που συγκρίνει

πόσοι σύνδεσμοι υπάρχουν στο γράφο με τους συνδέσμους που θεωρητικά θα μπορούσαν να υπάρξουν στον ίδιο γράφο. Στην ουσία μας δίνει της αλληλεπιδράσεις των tweets . Μεγαλύτερο density έχει ο συνδυαστικός πίνακας all καθώς τα Tweets έχουν περισσότερες συνδέσεις μεταξύ τους σε σχέση με τα άλλα χαρακτηριστικά. Το average clustering coefficient μετρά κατά πόσο οι γείτονες ενός κόμβου με τους οποίους συνδέεται με ακμές , είναι συνδεδεμένοι μεταξύ τους. Η μεγαλύτερη τιμή για αυτό το μέτρο αντιστοιχεί στον retweeted γράφο .

Στη συνέχεια θέλουμε να βρούμε τις κοινότητες που δημιουργούνται στους γράφους . Για να επιτευχθεί αυτό τρέχουμε τον αλγόριθμο Modularity με resolution 0.6 για κάθε γράφο.

Έπειτα θέλουμε κάθε κοινότητα να έχει διαφορετικό χρώμα. Αυτό το κάνουμε επιλέγοντας node->attribute->modularity class->apply . Για να οπτικοποιήσουμε καλύτερα τα αποτελέσματα επιλέγουμε το layout Yifan Hu και πατάμε run. Έτσι μας εμφανίζει τις κοινότητες με πιο εμφανή τρόπο.



Τέλος για την σύγκριση των δομών των κοινοτήτων χρειάζεται να υπολογίσουμε τον δείκτη nmi για κάθε ζεύγος χαρακτηριστικών . Αυτός

ο δείκτης υπολογίζει την κοινή πληροφορία που υπάρχει ανάμεσα σε δύο vectors. Αρχικά εξάγουμε για κάθε γράφο τους πίνακες που περιγράφουν την ανάθεση των Tweets σε κλάσεις . Πηγαίνουμε στο data laboratory και επιλέγουμε export table με επιλογή του modularity class έτσι μας εξάγει ένα csv αρχείο το οποίο περιέχει διακριτούς αριθμούς οι οποίοι αντιπροσωπεύουν την κλάση στην οποία ανήκει ένα tweet . Αφού εξάγουμε όλους τους πίνακες υλοποιούμε σε Java ένα αλγόριθμο υπολογισμού του δείκτη nmi (NMI.java) . Η υλοποίηση αυτή παίρνει ως είσοδο δύο csv tables και τα συγκρίνει μεταξύ τους με βάση τη συχνότητα εμφάνισης των κλάσεων σε κάθε αρχείο . Οι πληροφορίες πάρθηκαν από http://mathworld.wolfram.com/MutualInformation.html

και http://mathworld.wolfram.com/Entropy.html

Οι πληροφορίες που εξάγονται για κάθε ζεύγος φαίνονται στον παρακάτω πίνακα :

	hashtag	mentioned	retweeted	url	all
hashtag	1.0	0.6430	0.6612	0.6967	0.4292
mentioned	0.6430	1.0	0.8271	0.8021	0.3315
retweeted	0.6612	0.8271	1.0	0.8493	0.3453
url	0.6967	0.8021	0.8493	1.0	0.3425
all	0.4292	0.3315	0.3453	0.3425	1.0

Η τιμές εκφράζουν πόσο όμοιοι είναι οι γράφοι ως προς τις κοινότητες που περιέχουν .