ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

Τμήμα Πληροφορικής

ΕΠΛ 371 – Προγραμματισμός Συστημάτων

ΑΣΚΗΣΗ 2 – Web 2.0 Mash-up Engine στο Bash

Διδάσκων: Δημήτρης Ζεϊναλιπούρ

Υπεύθυνος Εργασίας: Παύλος Αντωνίου

Ημερομηνία Ανάθεσης: Παρασκευή, 14/02/2014
Ημερομηνία Παράδοσης: Τετάρτη, 5/03/2014 (19 μέρες)

(η λύση να υποβληθεί σε zip μέσω του Moodle και ο κώδικας να να παραδοθεί εκτυπωμένος στο εργαστήριο)

http://www.cs.ucy.ac.cy/courses/EPL371

I. Στόχος Άσκησης

Στόχος αυτής της άσκησης είναι η εξοικείωση με προχωρημένες τεχνικές προγραμματισμού στο κέλυφος Bash, και η εκτίμηση της ευκολίας με την οποία μπορεί κανείς να δημιουργήσει ένα σύνθετο σύστημα μέσω προγραμμάτων ωφελιμότητας (system utilities). Συγκεκριμένα, σε αυτή την άσκηση θα έχετε την ευκαιρία να κάνετε χρήση των εξής εννοιών που έχετε διδαχθεί στα πλαίσια του μαθήματος: εντολή *πίνακες, συνθήκες ελέγχου, δομές επανάληψης, κανονικές εκφράσεις, επεξεργαστές ροών (sed, awk) και χρήση συναρτήσεων με τα προαναφερθέντα.*

Το θέμα της άσκησης είναι η υλοποίηση ενός προγράμματος συνδυασμού περιεχομένων από ιστοσελίδες του Ιστού 2.0 (Web 2.0 Mashup). Ειδικότερα, στην εργασία αυτή θα ασχοληθούμε με την ανάκτηση και ανάλυση αρχείων τύπου json (json files) από το σύστημα κοινωνικής δικτύωσης twitter.com και από τη νέα υπηρεσία ανταλλαγής μηνυμάτων rayzit.com, που αναπτύχθηκε από το Εργαστήριο DMSL (dmsl.cs.ucy.ac.cy) του Τμήματος Πληροφορικής στο Πανεπιστήμιο Κύπρο. Η ανάκτηση δεδομένων θα γίνεται χρησιμοποιώντας την εντολή κελύφους curl. Οι λειτουργίες του προγράμματος σας και το αναμενόμενο αποτέλεσμα περιγράφονται αναλυτικότερα στην συνέχεια.

Την τελευταία δεκαετία έχει σημειωθεί παγκοσμίως μια τεράστια έκρηξη στην παραγωγή δεδομένων. Το ποσοστό της πληροφορίας που ανταλλάσσεται μέσω του Internet αυξάνεται όλο και περισσότερο. Η αύξηση αυτή των δεδομένων στο Internet, οφείλεται στην αλλαγή του τρόπου παραγωγής και διαχείρισής τους. Με την εμφάνιση του Web 2.0, ο χρήστης, εκτός από καταναλωτής, γίνεται και παραγωγός πληροφορίας. Το Internet δεν χρησιμεύει απλά για πρόσβαση σε δεδομένα. Οι web εφαρμογές δίνουν τη δυνατότητα στους χρήστες, να αλληλεπιδρούν μεταξύ τους και να μοιράζονται πληροφορίες. Οι πληροφορίες αυτές δεν είναι κατ' ανάγκη αρχεία κειμένου. Κείμενο, φωτογραφίες, video κι άλλα είδη αρχείων μεταφορτώνονται διαρκώς στο Διαδίκτυο.

Τα κοινωνικά δίκτυα αποτελούν κατ' εξοχήν παράδειγμα εφαρμογής του Web 2.0. Χαρακτηριστικά παραδείγματα κοινωνικών δικτύων είναι το Facebook και το Twitter. Με βάση τα επίσημα στατιστικά, σήμερα (2014), το Facebook αριθμεί πάνω από 1 δισεκατομμύριο ενεργούς χρήστες, και το Twitter έχει πάνω από 200 εκατομμύρια. Επίσης, για να έχουμε μια ιδέα του πλήθους των δεδομένων, που μπορούν να παράξουν αυτοί οι χρήστες, αναφέρουμε, ότι στο Facebook

ανεβαίνουν κατά μέσο όρο πάνω από 250 εκατομμύρια φωτογραφίες τη μέρα και στο Twitter δημοσιεύονται κατά μέσο όρο περίπου 340 εκατομμύρια tweets την μέρα.

Καταλαβαίνουμε επομένως, ότι ανά πάσα στιγμή, στα κοινωνικά δίκτυα παράγεται πλήθος νέας πληροφορίας. Αναλύοντας επομένως το περιεχόμενο αυτών των ιστοσελίδων, ή όπως λέμε διαφορετικά, αναλύοντας τον "ιστό πραγματικού χρόνου" (real-time Web), μπορούν να εξαχθούν ιδιαίτερα χρήσιμες πληροφορίες σχετικά με τις συμπεριφορές και τα συναισθήματα των χρηστών. Οι πληροφορίες, που μπορούμε να εξάγουμε μπορεί να είναι σχετικές με τη δημοτικότητα ενός ατόμου ή ενός προϊόντος, αλλά μπορεί να είναι και αρκετά πιο σύνθετες, όπως η πρόβλεψη ενός εκλογικού αποτελέσματος. Έχοντας ως κίνητρο τα παραπάνω, κατανοώντας την ανάγκη για διαχείριση του τεράστιου όγκου των δεδομένων που υπάρχει σήμερα, αλλά και αναγνωρίζοντας τη σημασία των πληροφοριών, που ρέουν στα κοινωνικά δίκτυα, **στην άσκηση αυτή, θα αναπτύξουμε ένα πρόγραμμα, το οποίο να χρησιμοποιεί δεδομένα του Twitter για την εξαγωγή πληροφοριών από αυτά**.

Το Rayzit είναι μια τεχνολογία ανταλλαγής μηνυμάτων που επιτρέπει την ανταλλαγή ερωτήσεων και ιδεών με τους πλησιέστερους (γεωγραφικά) χρήστες ανεξάρτητα από το πόσο μακριά και ποιοι είναι οι χρήστες αυτοί. Χρησιμοποιώντας ένα συνδυασμό σύγχρονων υπολογιστικών τεχνικών και εννοιών crowdsourcing, το Rayzit εισαγάγει μια ευχάριστη εμπειρία κοινωνικής αλληλεπίδρασης!

Στοχος της εργασιας

II. Εισαγωγή στο JSON

Το JSON (JavaScript Object Notation) είναι ένα ελαφρύ πρότυπο ανταλλαγής δεδομένων σε μορφή κειμένου (text). Είναι εύκολο για τους ανθρώπους να το διαβάσουν και γράψουν. Είναι εύκολο για τις μηχανές να το αναλύσουν (parse) και να το παράγουν (generate). Είναι βασισμένο πάνω σε ένα υποσύνολο της γλώσσας προγραμματισμού JavaScript, Standard ECMA-262 Έκδοση 3η - Δεκέμβριος 1999. Το JSON είναι ένα πρότυπο κειμένου το οποίο είναι τελείως ανεξάρτητο από γλώσσες προγραμματισμού αλλά χρησιμοποιεί πρακτικές (conventions) οι οποίες είναι γνωστές στους προγραμματιστές της οικογένειας προγραμματισμού C, συμπεριλαμβανομένων των C, C++, C#, Java, JavaScript, Perl, Python, και πολλών άλλων. Αυτές οι ιδιότητες κάνουν το JSON μια ιδανική γλώσσα προγραμματισμού ανταλλαγής δεδομένων.

Το JSON είναι χτισμένο σε δύο δομές:

- Μια συλλογή από ζευγάρια ονομάτων/τιμών. Σε διάφορες γλώσσες προγραμματισμού, αυτό αντιλαμβάνεται ως ένα object, καταχώριση, δομή, λεξικό, πίνακα hash (hash table), λίστα κλειδιών, ή associative πίνακα.

- Μία ταξινομημένη λίστα τιμών. Στις περισσότερες γλώσσες προγραμματισμού, αυτό αντιλαμβάνεται ως ένας πίνακας (array), διάνυσμα, λίστα, ή ακολουθία.

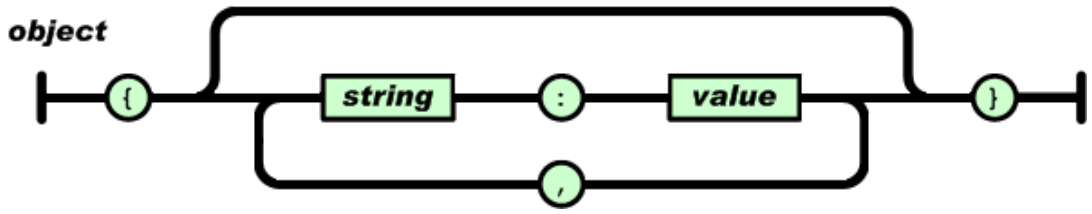
Αυτά είναι τα universal data structures. Ουσιαστικά όλες οι μοντέρνες γλώσσες προγραμματισμού τα υποστηρίζουν με τον έναν ή τον άλλον τρόπο. Λογικό είναι πως ένα πρότυπο δεδομένων το οποίο είναι εύκολα μεταβαλλόμενο με γλώσσες προγραμματισμού οι οποίες επίσης είναι βασισμένες σε αυτές τις δομές.

Στο JSON, παίρνουν αυτές τις μορφές:

Ένα αντικείμενο (object) είναι ένα άτακτο σύνολο από ζευγάρια ονόματων/τιμών. Ένα αντικείμενο (object) ξεκινάει με { (αριστερό άγκιστρο) και τελειώνει με } (δεξιό άγκιστρο). Κάθε

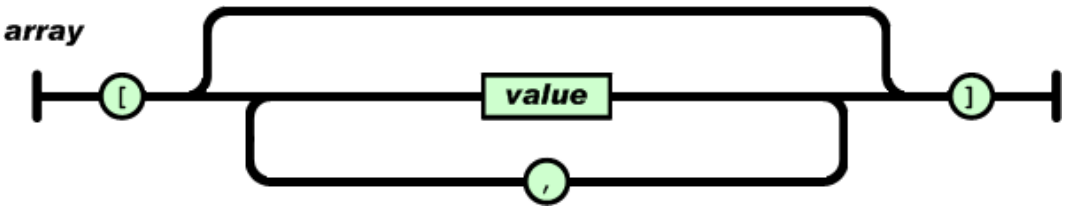
```
object
        {}
        { members }
members
        pair
        pair , members
pair
        string : value
array
        []
        [ elements ]
elements
        value
        value , elements
value
        string
        number
        object
        array
        true
        false
        null
```

όνομα ακολουθείται από : (άνω-κάτω τελεία) και τα ζευγάρια ονόματος/τιμής χωρίζονται από , (κόμμα).



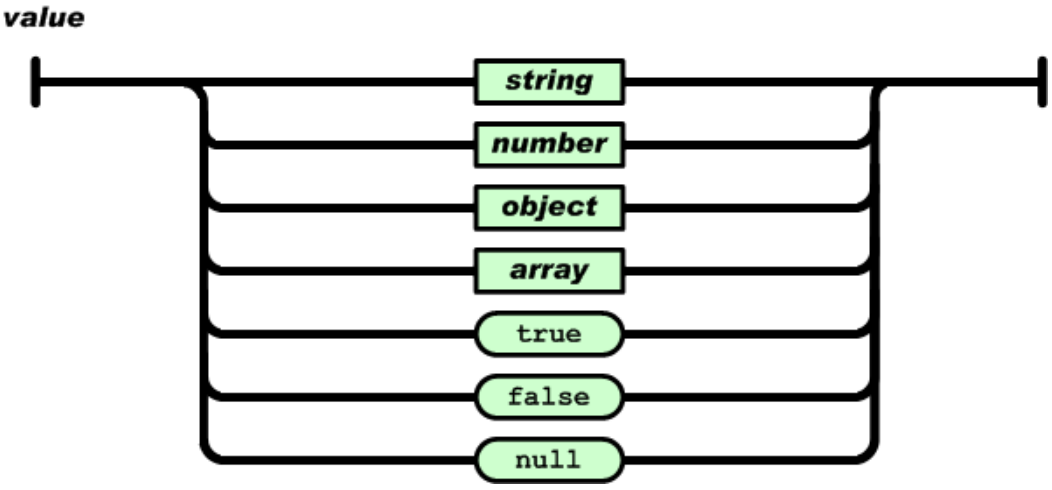
Εικόνα 1

Ένας πίνακας (array) είναι μια συλλογή από τιμές σε σειρά. Ένας πίνακας (array) ξεκινάει με [(αριστερή αγγύλη) και τελειώνει με] (δεξιά αγγύλη). Οι τιμές χωρίζονται με , (κόμμα).



Εικόνα 2

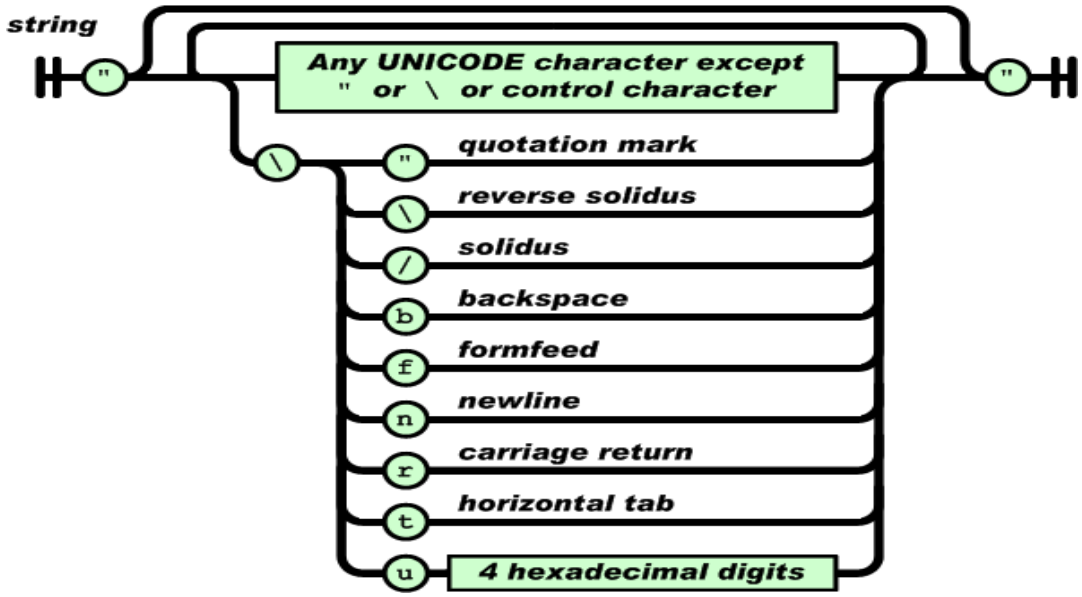
Μία τιμή μπορεί να είναι string μέσα σε διπλά quotes, ή αριθμός (number), ή true ή false ή null, ή αντικείμενο (object) ή πίνακας (array). Αυτές οι τιμές μπορεί να είναι και ανακατεμμένες.



Εικόνα 3

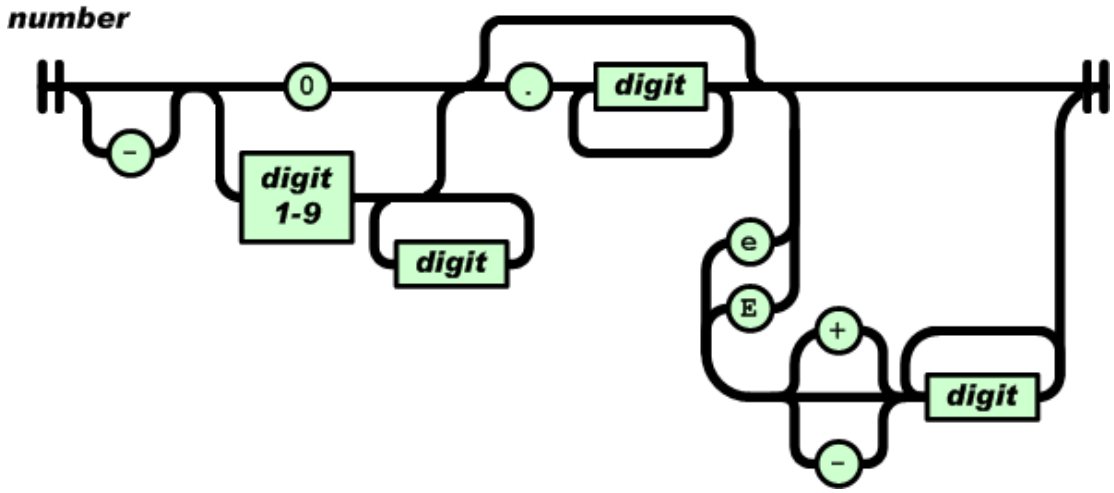
```
string
        ""
        " chars "
chars
        char
        char chars
char
        any-Unicode-character-
          except-"-or-\-or-
          control-character
        \"
        \\
        \/
        \b
        \f
        \n
        \r
        \t
        \u four-hex-digits
number
        int
        int frac
        int exp
        int frac exp
int
        digit
        digit1-9 digits
        - digit
        - digit1-9 digits
frac
        . digits
exp
        e digits
digits
        digit
        digit digits
e
        e
        e+
        e-
        E
        E+
        E-
```

Ένα string είναι μια συλλογή από μηδέν ή περισσότερους Unicode χαρακτήρες, μέσα σε διπλά quotes, χρησιμοποιώντας αντίστροφους κάθετους \ (backslash) για escapes. Ένας χαρακτήρας αντιπροσωπεύεται ως ένας μονός χαρακτήρας string. Ένα string μοιάζει πολύ σαν ένα C ή Java string.



Εικόνα 4

Ένας αριθμός (number) μοιάζει πάρα πολύ με ένα C ή Java αριθμό (number), με την διαφορά πως τα οκταδικά και δεκαεξαδικά συστήματα δεν χρησιμοποιούνται.



Εικόνα 5

Τα κενά (whitespace) μπορούν να εισαχθούν ανάμεσα σε οποιοδήποτε ζευγάρι tokens. Με εξαίρεση μερικών λεπτομερειών κωδικοποίησης (encoding), αυτό περιγράφει γενικότερα την γλώσσα (προγραμματισμού).

III. Rayzit

Τα βασικά χαρακτηριστικά της τεχνολογίας Rayzit είναι:

- Αποστολή μηνυμάτων "Rayz" (κείμενο, ήχο, βίντεο ή εικόνα) σε πλήθος χρηστών (συνήθως σε μερικές εκατοντάδες ανθρώπους) που είναι πιο κοντά σε εσάς.
- Κάνετε "Re-Rayz" ένα μήνυμα που θα λάβετε για να γνωστοποιήσετε σημαντικά πράγματα.
- Κάνετε "Star" ένα μήνυμα για να παρακολουθήσετε μια συζήτηση όπου κι αν πάτε.
- Δεν απαιτείται εγγραφή (sign-up), δημιουργία προφίλ, ή συλλογή προσωπικών δεδομένων. Μπορείτε να διατηρήσετε την ανωνυμία σας και την ιδιωτική σας ζωή ενώ παράλληλα να αλληλεπιδράτε με τους ανθρώπους γύρω σας.

Για να μπορέσουμε να λάβουμε πληροφορίες για κάποια μηνύματα "Rayzs" πρέπει να μελετήσουμε το API που παρέχει το Rayzit (https://api.rayzit.com/). Παρόλο που το Rayzit API απαιτεί ένα ειδικό κλειδί χρήσης (appId), δεν χρειάζεται στο παρόν στάδιο πιστοποίηση της ταυτότητας (authentication) της εφαρμογής ούτε θέτει περιορισμούς στον όγκο δεδομένων που ανακτώνται εντός συγκεκριμένου χρονικού διαστήματος. Συνεπώς, η πρόσβαση στις πληροφορίες του Rayzit είναι κάπως ευκολότερη από την πρόσβαση στις πληροφορίες του Twitter, το οποίο θα δούμε αργότερα και το οποίο απαιτεί πιστοποίηση τόπου OAuth.

Εάν θέλετε να πειραματιστείτε με το Rayzit μπορείτε να ανακτήστε το απευθείας από το Windows Phone Market. Για Android ή iOS παρακαλώ μιλήστε με τον Γ. Λάρκου (DMSL) για προμήθεια αντίστοιχων beta εκδόσεων.

Στα πλαίσια της εργασίας θα χρησιμοποιήσουμε *(α)* μια ***αίτηση για την ανάκτηση μηνυμάτων Rayzs εντός των τελευταίων X λεπτών (latest messages)*** και (β) μια ***αίτηση για την ανάκτηση μηνυμάτων Rayzs εντός περιοχής γύρω από κάποια δεδομένη θέση (geo messages)***. Οι αιτήσεις που θα χρειαστούμε για τις ανακτήσεις αυτές δίνονται πιο κάτω.

A. Ανάκτηση μηνυμάτων Rayzs εντός των τελευταίων X λεπτών (Latest Messages)

Μπορούμε να λάβουμε μηνύματα Rayzs που στάλθηκαν τα τελευταία X λεπτά, όπου το X είναι μεταβλητό και μπορεί να λάβει τιμές από 1 μέχρι 1440. Επιστρέφονται μέχρι 50 μηνύματα. Αν εκτελέσουμε την εντολή:

```
curl http://api.rayzit.com/latest/rayz/300
```

θα λάβουμε (το πολύ 50 μηνύματα) που στάλθηκαν από χρήστες του Rayzit εντός των τελευταίων 300 λεπτών όπως φαίνεται πιο κάτω:

```
{
   "counter":1,
   "status":"success",
   "latest":[
      {
         "rayz_message":"Hi.....free koi nahi hai......sab paid hai",
         "attachments":{
            "images":[
            ],
            "audio":[
            ],
```

```
            "videos":[
            ]
         },
         "timestamp":1391638779855,
         "follow":0,
         "report":0,
         "rerayz":0,
         "maxDistance":0,
         "geometry":{
            "type":"Point",
            "coordinates":[
               77.1,
               28.4
            ]
         }
      }
   ]
}
```

Β. Ανάκτηση μηνυμάτων Rayzs εντός περιοχής γύρω από δεδομένη θέση (Geo Messages)

Μπορούμε να λάβουμε μηνύματα Rayzs που στάλθηκαν γύρω από δεδομένη θέση. Η θέση θα προκαθορίζεται μέσω γεωγραφικού πλάτους (latitude) και γεωγραφικού μήκους (longitude) που θα δίνεται από το χρήστη. Το πιο κάτω ερώτημα:

```
curl http://api.rayzit.com/nearbyrayz/33.221232/35.231231/500000
```

επιστρέφει τα μηνύματα Rayzs που στάλθηκαν από χρήστες που βρίσκονταν γύρω από την περιοχή με latitude=33.221232 και longitude=35.231231 σε ακτίνα από 5001 μέτρα έως 500000 μέτρα (δηλ. 500Km). Εκτός από το γεωγραφικό πλάτος και μήκος, ο χρήστης μπορεί να δώσει το άνω όριο της κυκλικής λωρίδας (δηλ. το 500Km στο πιο πάνω παράδειγμα), ενώ το κάτω όριο θα είναι πάντοτε 5001 m για λόγους ιδιωτικότητα των δεδομένων. Παράδειγμα απάντησης για την πιο πάνω αίτηση δίνεται στο αρχείο **rayz_position.json (as2-supplementary.zip).**

IV. Twitter

Για να μπορέσουμε να εκμεταλλευτούμε τις πληροφορίες του Twitter θα πρέπει οι εφαρμογές μας να αξιοποιήσουν το Web 2.0 *Application programming interface (API) της εν λόγω εφαρμογής*. Σήμερα, αρχές του 2014, η τελευταία (και εν ενεργεία) έκδοση του API είναι η 1.1. Η παλαιότερη έκδοση 1.0 έχει σταματήσει να προσφέρεται πλέον (δείτε ανακοίνωση εδώ https://blog.twitter.com/2013/api-v1-is-retired). Στην έκδοση 1.0 επιτρεπόταν στις εφαρμογές να αποκτήσουν πρόσβαση σε δεδομένα του Twitter χωρίς να απαιτείται πιστοποίηση της ταυτότητας (authentication) δηλαδή χωρίς να γνωρίζει το Twitter ποιος ανακτά τα δεδομένα (αντίστοιχα με το παράδειγμα που είδαμε για την εφαρμογή Rayzit). Στην έκδοση 1.1 αυτό καταργήθηκε, και κάθε αίτηση (request) πρέπει να πιστοποιείται. Έτσι πρέπει να ακολουθείται μια συγκεκριμένη διαδικασία που περιγράφεται πιο κάτω.

Για να δημιουργήσει κάποιος developer ένα twitter application πρέπει κατ' αρχάς να δημιουργήσει λογαριασμό στο Twitter (εκτός κι αν ήδη έχετε). Στη συνέχεια, θα πρέπει ο developer να επισκεφτεί την ιστοσελίδα: https://dev.twitter.com/, πάνω δεξιά να πατήσει "Sign In" και μόλις συνδεθεί, στο πάνω δεξιά drop down menu να επιλέξει "My applications". Στη σελίδα που θα ανοίξει να επιλέξει "Create a new application" και να δημιουργήσει μια εφαρμογή, δηλώνοντας το όνομα, περιγραφή και url (της ιστοσελίδας) της εφαρμογής (για ιστοσελίδα βάλτε τη σελίδα του μαθήματος). Μετά τη δημιουργία της εφαρμογής, ο developer μπορεί από την καρτέλα Settings να ρυθμίζει τα δικαιώματα (read, write) που επιθυμεί να έχει η εφαρμογή. Επίσης πατήστε "Create my access token" για να δημιουργηθούν τα "oauth_token" (ή άλλως Access token που είναι το δημόσιο σας κλειδί) και "oauth_token_secret" (ή άλλως Access Token Secret που είναι το ιδιωτικό σας κλειδί το οποίο δεν θα πρέπει να κοινοποιείται σε άλλους) με τα οποία θα "υπογράφετε" τις αιτήσεις σας (χρησιμοποιούνται στη διαδικασία δημιουργίας υπογραφής πριν την αποστολή κάθε αίτησης, το δημόσιο κλειδί περιλαμβάνεται και σε κάθε αίτηση).

Στην παρούσα εργασία θα χρησιμοποιήσουμε αιτήσεις από το API έκδοση 1.1 και πιο συγκεκριμένα το REpresentational State Transfer (REST) API 1.1. Όλες οι αιτήσεις που μπορεί κάποιος να στείλει με βάση το API αυτό βρίσκονται στη σελίδα https://dev.twitter.com/docs/api/1.1. Κάθε εντολή στο REST API 1.1 αποτελείται από το **request type** (GET, POST, DELETE, PUT, HEAD), το **request URI**, το **request query** και το OAuth signature (υπογραφή). Για την διαδικασία δημιουργίας της υπογραφής θα μιλήσουμε πιο κάτω.

Στα πλαίσια της εργασίας θα δημιουργήσουμε (α) μια ***αίτηση ανάκτησης των πρόσφατων μηνυμάτων (tweets) από κάποιο χρήστη του Twitter*** και (β) μια ***αίτηση για την ανάκτηση μηνυμάτων (tweets) που έγιναν σε συγκεκριμένη περιοχή***. Οπότε θα χρειαστούμε αιτήσεις τύπου **GET** για τις ανακτήσεις αυτές.

A. Αίτηση ανάκτησης πρόσφατων μηνυμάτων (tweets) από το timeline ενός χρήστη

Όλες οι αιτήσεις του REST API 1.1 βρίσκονται στην σελίδα https://dev.twitter.com/docs/api/1.1. Αν θέλουμε να χρησιμοποιήσουμε μια εντολή που μας επιστρέφει τα μηνύματα (tweets) στο timeline ενός χρήστη του οποίου γνωρίζουμε το username (στο Twitter, το username ενός χρήστη ονομάζεται screen_name), μπορούμε από τη πιο πάνω σελίδα να δούμε ότι η αίτηση αυτή γίνεται μέσω της εντολής GET statuses/user_timeline. Οπότε επιλέγουμε την εντολή αυτή. Εκεί βλέπουμε το https://dev.twitter.com/docs/api/1.1/get/statuses/user_timeline που είναι το request URI. Πιο κάτω (ανάμεσα στα άλλα) έχουμε τις παραμέτρους screen_name και count τα οποία θα χρησιμοποιήσουμε για να δημιουργήσουμε το request query. Αν θέλουμε να αναζητήσουμε τα τελευταία 3 μηνύματα ενός χρήστη με βάση το screen_name (π.χ., csdeptucy), το request query είναι το **screen_name=csdeptucy&count=3**. Για να μπορέσουμε να εκτελέσουμε την αίτηση μας μένει να δημιουργήσουμε την υπογραφή (OAuth signature). Για να δημιουργηθεί μια τέτοια υπογραφή (με χρήση του αλγορίθμου HMAC-SHA1) πρέπει να ακολουθηθούν τα βήματα που περιγράφονται στο σύνδεσμο: https://dev.twitter.com/docs/auth/creating-signature .

Στο σημείο αυτό αξίζει να σημειωθεί ότι το Twitter περιορίζει τον αριθμό των αιτήσεων (ανά χρήστη και ανά εφαρμογή) έτσι ώστε να μην επιτρέπει μαζικές ανακτήσεις δεδομένων. Για παράδειγμα, η εντολή GET statuses/user_timeline έχει

όριο 180 αιτήσεις/user (δείτε στη δεξιά στήλη της σελίδας GET statuses/user_timeline) μέσα σε συγκεκριμένο χρονικό παράθυρο. Στην έκδοση 1.1 το χρονικό παράθυρο έχει διάρκεια 15 λεπτά (δείτε εδώ https://dev.twitter.com/docs/rate-limiting/1.1). Σημειώστε ότι η έκδοση 1.1 εισάγει και την έννοια του "Application Only Authentication" (450 αιτήσεις/εφαρμογή), το οποίο χρησιμοποιεί το OAuth 2.0 αλλά δεν θα χρειαστεί στην εργασία αυτή. Θα χρειαστεί μόνο το OAuth 1.0a HMAC-SHA1 που αφόρα το "User-based Authentication".

Μπορούμε να δούμε την αίτηση (μαζί με την υπογραφή την οποία παράγει ένα online εργαλείο του Twitter) ακολουθώντας την ένδειξη OAuth tool στα δεξιά της σελίδας που βρισκόμαστε. Κάτω από την ένδειξη OAuth tool **επιλέγουμε την εφαρμογή μας** και στη συνέχεια πατούμε "Generate OAuth signature". Στη σελίδα που θα ανοίξει, στο request query βάζουμε screen_name=paul_antoniou&count=20 και στο κάτω μέρος πατούμε "See OAuth signature for this request". Το cURL command που θα δημιουργηθεί αυτόματα μπορεί να χρησιμοποιηθεί μέσα στο bash script **(για δοκιμή και μόνον)** αφού περιέχει ολόκληρη την αίτηση. Πιο κάτω βλέπουμε το παραχθέν cURL command:

---

curl --get 'https://api.twitter.com/1.1/statuses/user_timeline.json' --data 'count=3&screen_name=csdeptucy' --header 'Authorization: OAuth oauth_consumer_key="HWGnEoQwVbw1n3QokFTug", oauth_nonce="ca16a47b5de361f58c603f6dfa3f917b", **oauth_signature="ZSGuNQZgNfjDaUvcDXEIl3AMlP8%3D"**, oauth_signature_method="HMAC-SHA1", oauth_timestamp="1392032010", oauth_token="239719727-sRl4aRHw3oDaJk7KlYCXMq6aLmD5BA2iQmfgtBeA", oauth_version="1.0"' --verbose

---

Στο δικό σας πρόγραμμα θα πρέπει να παράγετε το cURL command το οποίο στην ουσία θα αποστέλλει μια αίτηση GET, όπως φαίνεται και πιο πάνω. Αρχικά χρειάζεστε:

- το request URI : θα είναι δεδομένο στο πρόγραμμά σας https://dev.twitter.com/docs/api/1.1/get/statuses/user_timeline
- το request query : Το screen_name θα δίνεται σαν είσοδος από το χρήστη του προγράμματός σας

Έπειτα, το βασικό θέμα είναι να «κτιστεί» το Authorization header στην επικεφαλίδα του μηνύματος GET. Το header αυτό ξεκινά με τη συμβολοσειρά «Authorization: OAuth» και στη συνέχεια, όπως περιγράφεται αναλυτικά https://dev.twitter.com/docs/auth/authorizing-request, το authorization header αποτελείται από 7 παραμέτρους:

- το oauth_consumer_key : δίδεται από το Twitter κατά τη δημιουργία της εφαρμογής
- το oauth_nonce : είναι τυχαία συμβολοακολουθία δικής σας επιλογής
- το oauth_signature : δημιουργείται ακολουθώντας τα βήματα δημιουργίας της υπογραφής
- το oauth_signature_method : είναι το HMAC-SHA1
- το oauth_timestamp : δείχνει τη χρονική στιγμή δημιουργίας της αίτησης (στη μορφή unix epoch time) και θα πρέπει να λαμβάνεται από το σύστημα
- το oauth_token : δίδεται από το Twitter κατά τη δημιουργία της εφαρμογής
- το oauth_version : που είναι το 1.0

Στο σημείο αυτό, αξίζει να αναφερθούμε στην εντολή cURL. Η εντολή αυτή μας δίνει τη δυνατότητα να επεξεργαστούμε ένα αίτημα HTTP, να ορίσουμε κεφαλίδες (headers), να στείλουμε την αίτηση σε ένα συγκεκριμένο URL, και να εξετάσουμε την απάντηση. Η εντολή cURL είναι χρήσιμη γιατί μας επιτρέπει να δημιουργήσουμε μια αίτηση με χειροκίνητο (manual)

τρόπο. Δέχεται ορισμένες παραμέτρους. Η παράμετρος --get δείχνει ότι το μήνυμα που θα στείλουμε θα είναι HTTP GET (αν δεν δοθεί αυτό η curl θα στείλει μήνυμα HTTP POST) και ακολουθείται από το url στο οποίο θα στείλουμε την αίτηση. Η παράμετρος --data ακολουθείται από κάποια δεδομένα που μεταφέρονται από το μήνυμα HTTP GET. Η παράμετρος --header ακολουθείται από μια σύμβολο-ακολουθία που θα προστεθεί στην κεφαλίδα (header) του προς αποστολή μηνύματος. Η παράμετρος --verbose στο τέλος της πιο πάνω αίτησης χρησιμοποιείται για να εκτυπωθούν στην οθόνη τα μηνύματα που στέλνονται και λαμβάνονται. Οι γραμμές που αρχίζουν με το γράμμα > δείχνουν τα μηνύματα που στέλνονται (outbound data) και οι γραμμές με το γράμμα < δείχνουν τα μηνύματα που λαμβάνονται (inbound data). Περισσότερα στοιχεία για την εντολή cURL μπορείτε να βρείτε στο man curl.

Όταν τρέξουμε την πιο πάνω εντολή, ο εξυπηρετητής του Twitter θα μας απαντήσει με ένα μήνυμα HTTP 200 OK (με Content-type: application/json;charset=utf-8 όπως θα δείτε στην οθόνη) και στο σώμα του θα περιέχει ένα json string. Μπορείτε να δείτε το json string στο αρχείο **twitter_user_csdeptucy_timeline.json (που βρίσκεται στο as2-supplementary.zip)**

Σημείωση: Στο πεδίο text μπορείτε να δείτε χαρακτήρες που ξεκινούν με \u. Το \u είναι ο unicode escape character και προηγείται μιας unicode escape sequence (π.χ., \u0398) που αντιπροσωπεύει ένα Unicode χαρακτήρα (π.χ., \u0398 είναι το γράμμα Θ).

Β. Αίτηση ανάκτησης μηνυμάτων (tweets) που δημιουργήθηκαν σε καθορισμένη απόσταση από μια θέση

Αν θέλουμε να βρούμε τα μηνύματα που δημιουργήθηκαν σε μια καθορισμένη περιοχή π.χ., σε απόσταση 50km από μια συγκεκριμένη θέση (η θέση ορίζεται μέσω των συντεταγμένων latitude (γεωγραφικό πλάτος), longitude (γεωγραφικό μήκος)) πρέπει να χρησιμοποιήσουμε την εντολή GET search/tweets. Η περιοχή (σημείο και απόσταση γύρω από το σημείο δίνονται μέσα στο request query μέσω της παραμέτρου geocode. Η εντολή GET search/tweets απαιτεί μέσα στο request query να περιλαμβάνεται ένα UTF8 URL-encoded search query το οποίο να προσδιορίζει τι ψάχνουμε μέσα στα προς αναζήτηση μηνύματα. Στη δική μας περίπτωση μπορείτε να βάλετε ότι θέλετε στο search query. Η χρήση της εντολής GET search/tweets περιγράφεται και πιο αναλυτικά στο https://dev.twitter.com/docs/using-search.

Αν θέλουμε να αναζητήσουμε τα μηνύματα που περιέχουν τη λέξη ela και που δημιουργήθηκαν 50km γύρω από το σημείο με latitude=35.144345 και longitude=33.411364 (τοποθεσία Τμήματος Πληροφορικής), το request query είναι το **query=ela&geocode=35.144345,33.411364,50km**. Τα βήματα δημιουργίας του cURL για το πιο πάνω ερώτημα περιγράφονται πιο κάτω:

a) πάμε στη σελίδα https://dev.twitter.com/docs/api/1.1

b) επιλέγουμε την εντολή GET search/tweets [Βλέπουμε ότι το request URI είναι το https://api.twitter.com/1.1/search/tweets.json το οποίο στο δικό σας πρόγραμμα θα είναι δεδομένο]

c) στα δεξιά της σελίδας κάτω από το OAuth tool επιλέγουμε την εφαρμογή μας και στη συνέχεια "Generate OAuth signature"

d) στο Request query βάζουμε q=ela&geocode=35.144345,33.411364,50km και αφήνουμε τα υπόλοιπα στοιχεία του query ως έχουν. [Ολόκληρο το request query θα είναι δεδομένο μέσα στο πρόγραμμά σας]

e) στο κάτω μέρος της σελίδας πατούμε "See OAuth signature for this request"

f) παράγεται το cURL (*) για λόγους δοκιμής. Στο δικό σας πρόγραμμα και αυτή η εντολή πρέπει να παράγεται από το πρόγραμμά σας.

V. Ζητούμενα άσκησης

Στην παρούσα άσκηση μας ενδιαφέρει να ανακτήσουμε δεδομένα και από τα 2 προαναφερθέντα συστήματα κοινωνικής δικτύωσης (twitter & rayzit) με bash, να συνδυάσουμε τα αποτελέσματα (mash up) και να τα παρουσιάσουμε στο χρήστη.

Στην άσκηση αυτή καλείστε να υλοποιήσετε τρεις εντολές, που συνοψίζονται ως ακολούθως:

Η πρώτη εντολή θα «κατεβάζει» (με χρήση 2 cURL commands) 50 μηνύματα από κάθε σύστημα κοινωνικής δικτύωσης (twitter & rayzit) που στάλθηκαν σε συγκεκριμένη περιοχή και θα αποθηκεύει τα δεδομένα τοπικά σε ένα κατάλογο (directory). Πιο συγκεκριμένα, θα ανακτήσουμε από το Twitter τα τελευταία 50 μηνύματα (tweets) που έγιναν στην περιοχή αυτή σε ακτίνα 500Km. Θα ανακτήσουμε επίσης και τα τελευταία 50 μηνύματα (rayzs) που έγιναν στην περιοχή 5001-500000m (γύρω από τη προσδιορισμένη θέση). Τα δεδομένα που μας ενδιαφέρουν από τα μηνύματα rayzs και τα οποία φαίνονται μέσα στο json string είναι: rayz_message, timestamp και rerayz. Τα δεδομένα που μας ενδιαφέρουν από τα μηνύματα tweets και τα οποία φαίνονται μέσα στο json string, είναι: created_at, text και retweet_count. Όλες οι ανακτήσεις θα γίνουν με κατάλληλα cURL commands.

Η δεύτερη εντολή θα «κατεβάζει» (με χρήση 2 cURL commands) πρόσφατα μηνύματα από κάθε σύστημα κοινωνικής δικτύωσης (twitter & rayzit) και θα αποθηκεύει τα δεδομένα τοπικά σε άλλο κατάλογο (directory). Πιο συγκεκριμένα, θα ανακτήσουμε τα τελευταία 20 μηνύματα (tweets) από το timeline κάποιου χρήστη με screen_name που θα είναι μεταβλητό (θα δίνεται από το χρήστη). Επίσης θα ανακτηθούν από το Rayit όλα τα μηνύματα που στάλθηκαν τα τελευταία 1440 λεπτά από όλους τους χρήστες. Τα δεδομένα που μας ενδιαφέρουν από τα μηνύματα tweets και rayzs είναι τα ίδια με τα πιο πάνω.

Με την τρίτη εντολή θα γίνονται επερωτήσεις πάνω στα δεδομένα που είναι αποθηκευμένα τοπικά. Λεπτομέρειες δίνονται πιο κάτω.

<u>Α) Ανάκτηση δεδομένων περιοχής</u>

Πρότυπο εντολής: **./geo_msgs.sh <lat> <long> <radius>**

Το πρόγραμμα που θα γράψετε σε bash θα καλεί 2 cURL commands για να ζητά (α) από το twitter τα μηνύματα γύρω από την προσδιορισμένη θέση (latitude, longitude) μέσα σε ακτίνα radius και (β) από το rayzit τα μηνύματα γύρω από την προκαθορισμένη θέση μέσα στην κυκλική λωρίδα όπου το άνω όριο της θα δίνεται από την ακτίνα radius.

Τα μηνύματα tweets που θα λαμβάνονται από το Twitter και Rayzit (σε json μορφή) θα αναλύονται με σκοπό κάποια δεδομένα του χρήστη και να αποθηκευτούν μέσα σε ένα κατάλογο με το όνομα twitter_msgs και rayzit_msgs αντίστοιχα. **Οι 2 αυτοί κατάλογοι θα βρίσκονται μέσα σε ένα κατάλογο με το όνομα geo_msgs**.

Πιο συγκεκριμένα, για κάθε μήνυμα θα δημιουργείται:

- Αρχείο (text file) με όνομα τον αύξων αριθμό του κάθε μηνύματος (αύξων αριθμός είναι η σειρά με την οποία υπάρχει το κάθε μήνυμα μέσα στο αρχείο json). Το αρχείο θα είναι στον αντίστοιχο κατάλογο αναλόγως του αν πρόκειται για μήνυμα tweet ή μήνυμα rayz.
- Μέσα στο κάθε θα υπάρχουν τα πιο κάτω στοιχεία:
    - Text:
    - Time:
    - Retweets: (αν πρόκειται για tweets) ή Rerayzs: (αν πρόκειται για rayzs)

Β) Ανάκτηση πρόσφατων δεδομένων

Πρότυπο εντολής: **./latest_msgs.sh [<screen_name> | <rayzit_time>]**

Το πρόγραμμα που θα γράψετε σε bash θα καλεί 2 cURL commands για να ζητά (α) από το Twitter τα τελευταία 20 tweets του χρήστη με το screen_name και (β) από το Rayzit τα μηνύματα rayzs που στάλθηκαν τα τελευταία **rayzit_time** λεπτά.

Τα μηνύματα tweets που θα λαμβάνονται από το Twitter και Rayzit (σε json μορφή) θα αναλύονται με σκοπό κάποια δεδομένα του χρήστη και να αποθηκευτούν μέσα σε ένα κατάλογο με το όνομα twitter_msgs και rayzit_msgs αντίστοιχα. **Οι 2 αυτοί κατάλογοι θα βρίσκονται μέσα σε ένα κατάλογο με το όνομα latest_msgs**.

Πιο συγκεκριμένα, για κάθε μήνυμα θα δημιουργείται:

- Αρχείο (text file) με όνομα τον αύξων αριθμό του κάθε μηνύματος (αύξων αριθμός είναι η σειρά με την οποία υπάρχει το κάθε μήνυμα μέσα στο αρχείο json). Το αρχείο θα είναι στον αντίστοιχο κατάλογο αναλόγως του αν πρόκειται για μήνυμα tweet ή μήνυμα rayz.
- Μέσα στο κάθε θα υπάρχουν τα πιο κάτω στοιχεία:
    - Text:
    - Time:
    - Retweets: (αν πρόκειται για tweets) ή Rerayzs: (αν πρόκειται για rayzs)

Γ) Ανάλυση Δεδομένων - Επερωτήσεις (Mash up)

Η τρίτη εντολή που θα δημιουργήσετε θα εκτελεί κάποιες λειτουργίες πάνω στα δεδομένα που βρίσκονται τοπικά. Οι διαφορετικές λειτουργίες της εντολής θα ορίζονται από τις διαφορετικές παραμέτρους όπως περιγράφεται πιο κάτω.

Πρότυπο εντολής: **./mashUp.sh [options]**

Options:

(α) geo-sort

Η επιλογή αυτή θα παρουσιάζει στην οθόνη τα μηνύματα (tweets & rayzs) που λήφθηκαν από την προσδιορισμένη θέση και ακτίνα (πρώτη εντολή), ταξινομημένα ως προς το χρόνο δημιουργίας τους από το πιο νέο ως το πιο παλιό, από το **geo_msgs**.

(β) latest-sort

Η επιλογή αυτή θα παρουσιάζει στην οθόνη τα μηνύματα (tweets & rayzs) ταξινομημένα se αύξουσα διάταξη χρονισμών από το **latest_msgs** (δεύτερη εντολή)**.**

(γ) geo-max-retransmitted

Η επιλογή αυτή θα παρουσιάζει στην οθόνη το μήνυμα (tweet ή rayz) που είχε επανασταλεί σε περισσότερους χρήστες, δηλαδή έγινε rerayz ή retweeted σε μεγαλύτερο κοινό από το **geo_msgs**. Εκτός από το μήνυμα θα πρέπει να τυπώνεται σε πόσους κοινοποιήθηκε και το πότε έγινε η κοινοποίηση αυτή στην μορφή που τυπώνει η εντολή date (π.χ Mon Feb 10 12:18:53 EET 2014). Σημειώστε ότι οι ημερομηνίες των rayzs και των tweets μπορεί να ακολουθούν άλλο πρότυπο άρα θα πρέπει να ευθυγραμμιστούν.

(δ) geo-before-date <year>

Η επιλογή αυτή θα παρουσιάζει στην οθόνη τα μηνύματα (twitter & rayzs) που στάλθηκαν πριν την χρονολογία (έτος) που δίδεται στο string year από το **geo_msgs.**

(ε) geo-oldest

Η επιλογή αυτή θα παρουσιάζει στην οθόνη το μήνυμα (tweet ή rayz) που δημιουργήθηκε πιο παλιά από το **geo_msgs.**

(στ) latest-oldest

Η επιλογή αυτή θα παρουσιάζει στην οθόνη το μήνυμα (tweet ή rayz) που δημιουργήθηκε πιο παλιά από το **latest_msgs.**

(στ) all-build-lexicon

Η επιλογή αυτή θα κτίζει ένα λεξικό (αρχείο με το όνομα lexicon.txt) με όλες τις μοναδικές λέξεις που υπάρχουν σε όλα τα μηνύματα της πρώτης & δεύτερης εντολής (**geo_msgs *και* latest_msgs)**. Σε κάθε γραμμή του αρχείου θα υπάρχει μια λέξη και δίπλα η συχνότητα εμφάνισης της κάθε λέξης (2 στήλες). Τα δεδομένα θα είναι ταξινομημένα ως προς την συχνότητα εμφάνισης των λέξεων (2η στήλη) από τη πιο μεγάλη συχνότητα στην πιο μικρή.

Παραδοτέα

Πρέπει να παραδώσετε **όλα** τα πηγαία αρχεία σας μέσω του Moodle. Καθυστερημένες υποβολές δεν θα γίνονται αποδεκτές.

V. Γενικοί Κανόνες

1. Το σύστημα δεν αφήνει ποτέ άχρηστα και μεταβατικά αρχεία στον δίσκο, ανεξάρτητα εάν διακοπεί η λειτουργία του προγράμματος από το κλείσιμο του κελύφους.
2. Το σύστημα πρέπει να χρησιμοποιεί τεχνικές δομημένου προγραμματισμού με την χρήση συναρτήσεων.
3. Το σύστημα πρέπει να ελαχιστοποιεί την χρήση πόρων του συστήματος (αρχεία, μνήμης, κτλ).
4. Το σύστημα πρέπει να μειώνει όσο το δυνατό περισσότερο τον χρόνο διεκπεραίωσης της ανάκτησης και επεξεργασίας των δεδομένων.

Καλή Επιτυχία!