ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μελέτη και Ανάπτυξη Μηχανισμών Ανίχνευσης της Διάδοσης Λανθασμένων Πληροφοριών σε Κοινωνικά Δίκτυα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μιχάλης Σ. Γέρου

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια

Αθήνα, Ιούνιος 2013

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Μελέτη και Ανάπτυξη Μηχανισμών Ανίχνευσης της Διάδοσης Λανθασμένων Πληροφοριών σε Κοινωνικά Δίκτυα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μιχάλης Σ. Γέρου

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 12η Ιουνίου 2013.

.....	.....	.....
Θεοδώρα Βαρβαρίγου	Συμεών Παπαβασιλείου	Βασίλειος Λούμος
Καθηγήτρια	Αν. Καθηγητής	Καθηγητής

Αθήνα, Ιούνιος 2013

.....

Μιχάλης Σ. Γέρου

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περιεχόμενα

Περιεχόμενα Εικόνων

Ευχαριστίες

Με την ευκαιρία της ολοκλήρωσης αυτής της διπλωματικής εργασίας, θα ήθελα να ευχαριστήσω την Καθηγήτρια Ε.Μ.Π. κ. Θεοδώρα Βαρβαρίγου για τη δυνατότητα που μου έδωσε να δουλέψω κάτω από την επίβλεψη και την πολύτιμη καθοδήγησή της.

Επίσης, θα ήθελα να ευχαριστήσω ιδιαίτερα την Δρ. Βασιλική Ανδρόνικου και τον Δρ. Κωνσταντίνο Τσερπε για την συνεχή καθοδήγηση και ενθάρρυνση που μου έδωσαν κατά την εκπόνηση της εργασίας αυτής καθώς και για τον χρόνο που μου αφιέρωσαν.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς και τους φίλους μου για την αγάπη και στήριξη που μου δίνουν αδιάκοπα όλα αυτά τα χρόνια.

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας ήταν η ανάπτυξη μιας μεθοδολογίας και η υλοποίηση μιας σειράς από μηχανισμούς για την ανίχνευση λανθασμένων πληροφοριών που διαδίδονται σε κοινωνικά δίκτυα. Δεδομένης της ταχύτατης εξάπλωσης των κοινωνικών δικτύων και της – μεταξύ άλλων – ανάδειξής τους σε ανερχόμενη πηγή πληροφόρησης για χρήστες τους, η ανίχνευση της διάδοσης λανθασμένης πληροφορίας μέσα σε αυτά αποτελεί ένα ερευνητικό αντικείμενο που αποκτά ολοένα και μεγαλύτερη σπουδαιότητα.

Για τη αποκόμιση των κατάλληλων γνώσεων, αρχικά μελετήθηκε τι είναι λανθασμένη πληροφορία και πιο λεπτομερώς τι σημαίνει ακούσια και εκούσια παραπληροφόρηση (disinformation / misinformation). Έπειτα, μελετήσαμε τη δουλειά και την προσέγγιση άλλων ερευνητών στο θέμα της αυτόματης ανίχνευσης λανθασμένων πληροφοριών σε κοινωνικά δίκτυα.

Ακολούθως, βασισμένοι στην δουλειά και προηγούμενων ερευνητών που ασχολήθηκαν με το θέμα, διατυπώσαμε την μεθοδολογία που προτείνουμε ως τρόπο για αυτόματη ανίχνευση λανθασμένης πληροφορίας σε κοινωνικά δίκτυα και πιο συγκεκριμένα για το κοινωνικό δίκτυο Twitter. Βασιστήκαμε στην υπόθεση πως μια πληροφορία που είναι αληθής συνήθως έχει πολλές και ανεξάρτητες πηγές, ενώ μια ψευδής πληροφορία συνήθως ξεκινά με ένα μικρό πλήθος – συχνά εξαρτώμενων μεταξύ τους - πηγών. Χρησιμοποιώντας σύγχρονες τεχνολογίες, δημιουργήσαμε μια υλοποίηση της μεθοδολογίας μας, την οποία και δοκιμάσαμε σε ένα σύνολο δεδομένων για να επαληθεύσουμε πως μπορεί να εφαρμοστεί έστω και σε μια απλοποιημένη μορφή.

Λέξεις κλειδιά

ανίχνευση λανθασμένων πληροφοριών, ακούσια και εκούσια παραπληροφόρηση, φήμες, κοινωνικά δίκτυα, web 2.0, twitter, cloud computing, java

Abstract

The aim of this thesis was to create and implement a methodology for the detection of false information spread in social networks. Given the rapid spreading of social networks and their emerging role as a source of information for many of its users, detecting the propagation of false information across them comprises a research field gaining increasing importance.

To reap the appropriate knowledge, initially we studied what is false information and what unintentional and deliberate misinformation (disinformation / misinformation) means. Then, we studied the work and approach of other researchers on the issue of automatic detection of false information on social networks.

Finally, based on previous work and research that dealt with the topic, we propose a methodology as a method for automatic detection of false information in social networks and more specifically on the Twitter social network. We were based on the assumption that a piece of information that is true usually has many independent sources, while false information usually is initiated from a small number of dependent sources. Using modern technologies, we created an implementation of our methodology, which we tested on a dataset to verify that it can be implemented even in a simplified form.

Keywords

detection of false information, unintentional and deliberate misinformation, disinformation, misinformation, rumor, social networks, web 2.0, twitter, cloud computing, java

1. Εισαγωγή

1.1 Γενικό πλαίσιο

Το διαδίκτυο είναι πλέον ένα εξελιγμένο εργαλείο που αποτελεί μέρος της καθημερινής ζωής πολλών ανθρώπων και σιγά σιγά επηρεάζει κάθε ανθρώπινη δραστηριότητα. Με ένα κλικ μπορείς να μεταφερθείς σε ένα οποιοιδήποτε σημείο του κόσμου, να μάθεις ό,τι σε ενδιαφέρει σε πολύ μικρό χρονικό διάστημα και να μιλήσεις με τους φίλους σου όσο μακριά και αν είναι.

Με την έλευση της νέας γενιάς του διαδικτύου, του Web 2.0, όλο και περισσότερο περιεχόμενο δημιουργείται από απλούς χρήστες με παραλήπτες άλλους απλούς χρήστες. Η τάση αυτή είναι αναμφίβολα ένα πολύ ευχάριστο στοιχείο, αφού το διαδίκτυο γίνεται ολοένα και πιο δημοκρατικό με το να δίνει φωνή σε οποιονδήποτε έχει πρόσβαση σε αυτό.

Δυστυχώς, μαζί με τα ισχυρά πλεονέκτημα του, η νέα γενιά του διαδικτύου έχει εξίσου ισχυρά μειονεκτήματα. Ένας απλός χρήστης, ακούσια ή εκούσια, μπορεί να δημιουργήσει μια λανθασμένη πληροφορία χωρίς κανένα έλεγχο και αυτή να διαδοθεί ραγδαία με απρόβλεπτες συνέπειες. Η παραπληροφόρηση μπορεί να αποτελεί ένα απλό ψέμα για την προσωπική ζωή κάποιου γνωστού προσώπου, μέχρι φήμες που αποσκοπούν να πλήξουν τις χρηματοπιστωτικές αγορές.

Όλα τα προαναφερθέντα, μπορούν να εκτελεστούν με σχετική ευκολία με τα εργαλεία που προσφέρει το σύγχρονο διαδίκτυο. Παρόλη την ραγδαία ανάπτυξη του διαδικτύου και των υπηρεσιών που προσφέρει, δεν παρέχει κανένα τρόπο να προειδοποιήσει τον χρήστη για την πιθανή διάδοση λανθασμένης πληροφορίας για το περιεχόμενο του.

1.2 Στόχος Διπλωματικής

Στην παρούσα εργασία εστιάσαμε στα κοινωνικά δίκτυα και πιο συγκεκριμένα στο Twitter. Στόχος μας, να αναπτύξουμε και να υλοποιήσουμε μια μεθοδολογία για την ανίχνευση λανθασμένων πληροφοριών που διαδίδονται σε κοινωνικά δίκτυα.

1.3 Διάρθρωση κειμένου

Στο δεύτερο κεφάλαιο της παρούσας εργασίας παρουσιάζεται μια επισκόπηση της νέας γενιάς του διαδικτύου, του Web 2.0, του περιβάλλοντος στο οποίο αναπτύχθηκαν τα κοινωνικά δίκτυα. Έπειτα, εστιάζουμε στο κοινωνικό δίκτυο Twitter αναφέροντας μερικά χρήσιμα στατιστικά και χαρακτηριστικά του. Τέλος, εξετάζουμε τις διάφορες μορφές τις οποίες μπορεί να πάρει η λανθασμένη πληροφορία και ποια προβλήματα εμφανίζονται όταν η διάδοση λανθασμένης πληροφορίας συναντήσει τα κοινωνικά δίκτυα.

Συνεχίζοντας, στο τρίτο κεφάλαιο αναφέρουμε συνοπτικά την έρευνα που έχει γίνει σχετικά με τον εντοπισμό των πηγών ή και των δεδομένων παραπληροφόρησης καθώς επίσης και τα ανοικτά ζητήματα που θεωρούμε πως υπάρχουν στο συγκεκριμένο θέμα.

Στο τέταρτο κεφάλαιο παρουσιάζεται η θεωρητική προσέγγιση της μεθοδολογίας μας, μοντελοποιώντας το πρόβλημα και παρουσιάζοντας βήμα προς βήμα την θεωρητική εκτέλεση της μεθοδολογίας μας.

Στην μεγαλύτερη ενότητα αυτής της εργασίας, στο πέμπτο κεφάλαιο, κάνουμε μια επισκόπηση των τεχνολογιών και υπηρεσιών που χρησιμοποιήσαμε για να υλοποιήσουμε την προτεινόμενη μεθοδολογία. Έπειτα, παρουσιάζουμε το βοηθητικό εργαλείο που αναπτύξαμε για να αναλύσουμε το σύνολο δεδομένων, εξηγούμε πως εφαρμόσαμε πρακτικά τα βήματα της μεθοδολογίας που προτείναμε και παρουσιάζουμε κάποιες εφαρμογές των μηχανισμών.

Τέλος, στο έκτο κεφάλαιο αναλύονται τα πλεονεκτήματα και μειονεκτήματα της αναπτυχθείσας μεθοδολογίας. Ακολούθως, παρουσιάζεται η σύνοψη της διπλωματικής εργασίας και τα επόμενα βήματα για τη βελτίωση της απόδοσης και της ακρίβειας της μεθοδολογίας και του εργαλείου που φτιάξαμε.

2. Το πρόβλημα: Φήμες και Κοινωνικά Δίκτυα

Σε αυτήν την ενότητα παρατίθεται μια γενική επισκόπηση του σύγχρονου περιβάλλοντος στο οποίο αναπτύχθηκαν τα κοινωνικά δίκτυα (Web 2.0) και του κοινωνικού δικτύου Twitter στο οποίο θα εστιάσουμε. Ακολούθως, εξετάζουμε τις διάφορες μορφές τις οποίες μπορεί να πάρει η λανθασμένη πληροφορία και ποια προβλήματα εμφανίζονται όταν η διάδοση λανθασμένης πληροφορίας συναντήσει τα κοινωνικά δίκτυα.

2.1 Web 2.0 & Κοινωνικά Δίκτυα

Web 2.0

Το Web 2.0 αποτελεί έναν όρο που επινοήθηκε για να δείξει μια μεγάλη αλλαγή στον χώρο του Παγκόσμιου Ιστού. Συνοπτικά, ο όρος Web 2.0 πρωτοαναφέρθηκε το 1999 για να περιγράψει ιστοσελίδες που χρησιμοποιούν τεχνολογίες για να δημιουργήσουν δυναμικές σελίδες, σε αντίθεση με τις στατικές σελίδες της προηγούμενης 'γενιάς' του παγκόσμιου ιστού.

Οι δυναμικές σελίδες του Web 2.0, επιτρέπουν στους χρήστες να κάνουν περισσότερα από το να ανακτούν μόνο πληροφορίες, όλα μέσα από το παράθυρο του web-browser τους και ανεξαρτήτως από το λειτουργικό τους σύστημα. Οι χρήστες συνήθως παραχωρούν και ασκούν έλεγχο στα δεδομένα που διαθέτουν σε ιδιοκτήτες σελίδων Web 2.0 για να τα αποθηκεύσουν. Αυτού του είδους οι ιστοσελίδες έχουν συνήθως μια νοοτροπία που ενθαρρύνει τους χρήστες να μοιράζονται δεδομένα τους, καθώς όσο περισσότερα χρησιμοποιήσουν την ιστοσελίδα τόσο πιο χρήσιμη γίνεται.

Γενικά, η πιο σημαντική αλλαγή που προσφέρει το Web 2.0 είναι μια ριζική αλλαγή στον τρόπο που οι άνθρωποι επικοινωνούν και ανταλλάζουν πληροφορίες μέσω του διαδικτύου. Ο παγκόσμιος ιστός αποτελείται από μια σειρά από online εργαλεία και πλατφόρμες όπου οι άνθρωποι μοιράζονται τις απόψεις, τις σκέψεις και τις εμπειρίες τους. Οι Web 2.0 εφαρμογές τείνουν να αλληλεπιδρούν πολύ περισσότερο με τον τελικό χρήστη παρά οι εφαρμογές-στοσελίδες του παραδοσιακού ιστού. Λόγω αυτού, ο τελικός χρήστης δεν είναι μόνο ένας χρήστης της εφαρμογής αλλά επίσης και ένας συμμετέχων. Χαρακτηριστικά παραδείγματα περιλαμβάνουν:

- *Podcasting*: Ηχογραφημένες εκπομπές δημιουργημένες από απλούς χρήστες που είναι διαθέσιμες στους "ακροατές" όποτε αυτοί θελήσουν να τις ακούσουν. Μπορούν να δημιουργηθούν από οποιοδήποτε απλό χρήστη διαθέτει μικρόφωνο, λογισμικό ηχογράφησης, και σύνδεση στο internet. Μπορούν να αναπαραχθούν από φορητές συσκευές και οι ακροατές τους μπορούν να σχολιάσουν και να βαθμολογήσουν την κάθε εκπομπή. (1)

- *Blogging*: Ξεκίνησαν σαν ημερήσια προσωπικά ημερολόγια από απλούς χρήστες. Συνήθως ένα blog δημιουργείται από κάθε άτομο ξεχωριστά και περιέχει σχόλια, περιγραφές από προσωπικές εμπειρίες και γεγονότα. Ένα blog μπορεί να περιέχει κείμενο, φωτογραφίες ακόμα και βίντεο σχετικά με το θέμα που αναφέρουν. Επίσης υπάρχει η δυνατότητα από τους αναγνώστες να γράφουν σχόλια και να συμμετέχουν έτσι ενεργά σε ένα θέμα. (2)

- *Tagging*: Τα tags είναι λέξεις-κλειδιά που συνδέονται ή αποδίδονται σε κομμάτια πληροφορίας (όπως εικόνες, άρθρα, αρχεία βίντεο κλπ) και έγιναν ιδιαίτερα δημοφιλής τεχνική κατηγοριοποίησης σε εφαρμογές του Web 2.0. Με τον τρόπο αυτό περιγράφουν το αντικείμενο και επιτρέπουν ταξινόμηση και αναζήτηση της πληροφορίας με βάση λέξεις-κλειδιά. Για παράδειγμα, στα blogs τα tags χρησιμοποιούνται ευρέως καθώς είναι πιο δυναμικά και ευέλικτα από την κλασική μέθοδο ταξινόμησης με κατηγορίες. Μπορούν να εξηγήσουν ευκολότερα τις θεματικές ενότητες τις οποίες καλύπτει ένα άρθρο, καθώς καλύπτουν με μεγαλύτερη ακρίβεια το ποικίλο υλικό το οποίο συνήθως αναγράφεται σε ένα blog. (3)

- *Social bookmarking*: Παρέχει την δυνατότητα να αποθηκευτούν οι σελιδοδείκτες ενός χρήστη σε μια ιστοσελίδα και να τους κατηγοριοποιήσει εκεί. Επίσης, επιτρέπει την πρόσβαση στους σελιδοδείκτες από οποιοδήποτε μέρος του κόσμου, αφού δεν βρίσκονται αποθηκευμένοι στον προσωπικό υπολογιστή του χρήστη. Τέλος, επιτρέπει τον διαμοιρασμό των σελιδοδεικτών με φίλους, μέλη της οικογένειας ή ακόμα και με τελείως άγνωστους. (4)

- *Social networking*: Πρόκειται για ιστοσελίδες που αποτελούν εικονικές κοινότητες, όπου οι χρήστες του Διαδικτύου έχουν τη δυνατότητα να δημιουργήσουν τα εικονικά τους προφίλ και να αναπτύξουν ένα δίκτυο επαφών, με τις οποίες μπορούν να επικοινωνούν μέσω της ιστοσελίδας. Θα αναλυθούν με περισσότερη λεπτομέρεια στην επόμενη ενότητα.

- *Web content voting*: Πρόκειται για τη δυνατότητα των χρηστών να βαθμολογήσουν το περιεχόμενο που διαβάζουν / βλέπουν / ακούν στο διαδίκτυο. Έγινε ιδιαιτέρα δημοφιλής τακτική για τις εφαρμογές του Web 2.0 αφού το περιεχόμενο που δημιουργείται από απλούς χρήστες αυξήθηκε και εμφανίστηκε η ανάγκη να ξεχωρίσει το πιο ποιοτικό περιεχόμενο. Χαρακτηριστικά παραδείγματα που βασίζονται στο web content voting είναι το Reddit και το Digg - πρόκειται για ιστοσελίδες που επιτρέπουν σε οποιοδήποτε χρήστη να προσθέσει περιεχόμενο, αλλά για να ανεβεί το περιεχόμενο στην καταταξη είναι απαραίτητη η “θετική” ψήφος από όσο το δυνατό περισσότερους χρήστες.

Μαζί με την αλλαγή στον τρόπο δημιουργίας περιεχομένου στο Web 2.0, υπήρξε μια μεγάλη αλλαγή και στις χρησιμοποιούμενες τεχνολογίες. Πιο συγκεκριμένα, από την πλευρά του πελάτη και του web-browser του χρήστη άρχισαν να χρησιμοποιούνται εκτενώς τεχνολογίες όπως Ajax και JavaScript. Αυτές οι τεχνολογίες έχουν την δυνατότητα να ανεβάζουν και να κατεβάζουν νέα δεδομένα από τον web-server χωρίς να απαιτείται η πλήρης επαναφόρτωση της σελίδας. Αυτό επιτρέπει στους χρήστες να μην περιμένουν την φόρτωση της σελίδας πριν να αλληλεπιδράσουν με την σελίδα, αλλά να μπορούν να έχουν πρόσβαση σε όλες τις πληροφορίες που προσφέρει μια σελίδα εκτός από το κομμάτι που φορτώνει.

Από την πλευρά του web-server, οι ιστοσελίδες Web 2.0 χρησιμοποιούν πολλές από τις ίδιες τεχνολογίες που χρησιμοποιούν και οι παραδοσιακές ιστοσελίδες. Γλώσσες όπως η PHP, η Ruby, η Perl, η Python καθώς και η JSP και η ASP.NET είναι συχνά χρησιμοποιούμενες από τους προγραμματιστές για την δημιουργία δυναμικού περιεχομένου σε ιστοσελίδες χρησιμοποιώντας πληροφορίες από αρχεία και βάσεις δεδομένων.

Αυτό που έχει επίσης αλλάξει στο Web 2.0 και απαίτησε τη δημιουργία νέων τεχνολογιών είναι ο τρόπος που οι πληροφορίες των ιστοσελίδων είναι διαμορφωμένες. Στις πρώτες μέρες του Διαδικτύου, υπήρχε μικρή ανάγκη για τις διαφορετικές ιστοσελίδες να επικοινωνούν μεταξύ τους και να μοιράζονται δεδομένα. Στο νέο Web 2.0 όμως, που δίνεται έμφαση στην συλλογικότητα και στο περιεχόμενο που δημιουργούν οι χρήστες, η ανταλλαγή δεδομένων μεταξύ των ιστοσελίδων έχει γίνει μια σημαντική υπηρεσία που πρέπει να προσφέρει μια ιστοσελίδα. Για να μοιραστεί τα δεδομένα της ένας δικτυακός τόπος με άλλους, πρέπει να είναι σε θέση να παράγει έξοδο σε μορφές όπως XML (Atom, RSS, κ.λπ.) και JSON. Όταν τα δεδομένα ενός δικτυακού τόπου είναι διαθέσιμα σε μία από αυτές τις μορφές, μια άλλη ιστοσελίδα μπορεί να τα χρησιμοποιήσει για να ενσωματώσει ένα μέρος της λειτουργικότητας αυτής της ιστοσελίδας στον εαυτό της.

Κοινωνικά Δίκτυα

Μια υπηρεσία κοινωνικής δικτύωσης είναι μια πλατφόρμα για τη δημιουργία κοινωνικών δικτύων ή κοινωνικών σχέσεων μεταξύ ανθρώπων οι οποίοι για παράδειγμα έχουν κοινά ενδιαφέροντα, δραστηριότητες, κτλ. Μια τέτοια υπηρεσία αποτελείται από μια αναπαράσταση του κάθε χρήστη (συχνά ένα προφίλ) που περιλαμβάνει τους κοινωνικούς δεσμούς του χρήστη, καθώς και μια ποικιλία επιπρόσθετων υπηρεσιών. Οι περισσότερες κοινωνικές υπηρεσίες δικτύου είναι web-based, δηλαδή παρέχουν τρόπους για τους χρήστες τους ώστε να αλληλεπιδρούν μέσω του Διαδικτύου. Οι υπηρεσίες κοινωνικής δικτύωσης επιτρέπουν στους χρήστες να μοιράζονται ιδέες, φωτογραφίες, δημοσιεύσεις, δραστηριότητες, εκδηλώσεις και τα ενδιαφέροντά τους με άλλους ανθρώπους στο δίκτυό τους. (5)

Τα κοινωνικά δίκτυα θεωρούνται εξαιρετικά επιτυχημένα αυτή την στιγμή και είναι ίσως τα πιο πετυχημένα παραδείγματα εφαρμογών της Web 2.0 εποχής. Παρακάτω παραθέτουμε μερικά επιτυχημένα κοινωνικά δίκτυα και μερικά στατιστικά για αυτά:

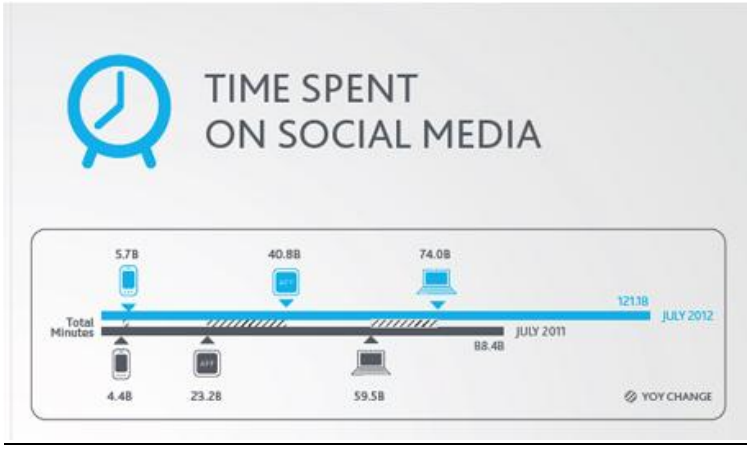
- Το *Facebook* είναι το πιο δημοφιλές κοινωνικό δίκτυο στον κόσμο. Οι χρήστες πρέπει να εγγραφούν πριν από τη χρήση του κοινωνικού δικτύου, και εν συνεχεία μπορούν να δημιουργήσουν ένα προσωπικό προφίλ, να προσθέσουν άλλους χρήστες ως φίλους, να ανταλλάσσουν μηνύματα και να ειδοποιούνται όταν οι φίλοι τους ανανεώνουν τις προσωπικές πληροφορίες τους. Όλοι έχουν ελεύθερη πρόσβαση στο να συμμετάσχουν σε δίκτυα που σχετίζονται με το πανεπιστήμιο τους, την εργασία τους, τη γεωγραφική περιοχή τους, κτλ. Αυτή τη στιγμή, σύμφωνα με το κοινωνικό δίκτυο, υπάρχουν 1.11 δισεκατομμύρια ενεργοί χρήστες. (6)

- Το *YouTube* είναι το δεύτερο πιο δημοφιλές κοινωνικό δίκτυο στον κόσμο. Επιτρέπει στους χρήστες του την αποθήκευση, αναζήτηση και αναπαραγωγή ψηφιακών ταινιών. Όλοι μπορούν να βλέπουν τις αποθηκευμένες ψηφιακές ταινίες, ενώ τα εγγεγραμμένα μέλη μπορούν να αποθηκεύουν απεριόριστο αριθμό ταινιών. Μαζί με τις ταινίες φαίνεται και ο αριθμός των μελών που τις έχουν δει, επιτρέποντας έτσι μια γενική εικόνα της δημοτικότητάς τους. Επίσης ένας χρήστης μπορεί να δηλώσει αν του αρέσει ένα βίντεο ή όχι, να αφήσει σχόλια στο κάθε βίντεο και να βαθμολογήσει τα σχόλια άλλων χρηστών. Το δίκτυο ανακοίνωσε πως έχει 1 δισεκατομύριο μοναδικούς *επισκεπτες* κάθε μήνα. (7)

- Το *Twitter*, το τρίτο πιο δημοφιλές κοινωνικό δίκτυο, επιτρέπει στους χρήστες του να στέλνουν και να διαβάζουν σύντομα μηνύματα κειμένου μέχρι 140 χαρακτήρες. Το Twitter ανακοίνωσε πως έχει 500 εκατομμύρια ενεργούς χρήστες τον Ιούλιο του 2012. (8) Το συγκεκριμένο δίκτυο θα αναλυθεί με λεπτομέρεια σε επόμενη ενότητα.

- Το *Foursquare* είναι ένα κοινωνικό δίκτυο βασισμένο στην τοποθεσία του χρήστη που εστιάζει σε φορητές συσκευές, όπως smartphones. Οι χρήστες του κάνουν "check in" σε χώρους (όπως εστιατόρια, καταστήματα, κτλ) χρησιμοποιώντας μια εφαρμογή για κινητά ή μέσω γραπτού μηνύματος επιλέγοντας από μία λίστα από χώρους που βρίσκει η εφαρμογή σε κοντινή απόσταση. Η τοποθεσία του χρήστη ανακτάτε από το GPS που συνήθως υπάρχει στα κινητά τηλέφωνα. Κάθε "check-in" που κάνει ένας χρήστης "βραβεύεται" με εικονικούς πόντους που του επιφέρουν σε βάθος χρόνου κάποια εικονικά "βραβεία". Το συγκεκριμένο δίκτυο ανακοίνωσε πως έχει 20.000.000 εγγεγραμμένους χρήστες μέσα σε μόλις 3 χρόνια. (9)

Ενδεικτικά, αναφέρουμε πως έρευνα της Nielsen Company για το τρίτο τετράμηνο του 2011 έδειξε πως οι Αμερικανοί πολίτες αφιερώνουν στα κοινωνικά δίκτυα το 22.5% του χρόνου που είναι συνδεμένοι στο διαδίκτυο. Αυτό το ποσοστό είναι πολύ περισσότερο από το διπλάσιο της δεύτερης κατά σειρά ασχολίας τους, που είναι τα διαδικτυακά παιχνίδια με ποσοστό 9.8%. (10)



Εικόνα 1: Ο συνολικός χρόνος που οι χρήστες ξόδεψαν σε κοινωνικά δίκτυα τον Ιούλιο 2012 και τον Ιούλιο 2011.

Όπως φαίνεται και στην παραπάνω εικόνα, έρευνα από την ίδια εταιρεία δείχνει πως ο συνολικός χρόνος που οι χρήστες ξόδεψαν σε κοινωνικά δίκτυα αυξήθηκε αισθητά τον Ιούλιο του 2012 σε σχέση με τον Ιούλιο του 2011. Τα παραπάνω δείχνουν ξεκάθαρα πως οι τάσεις που επικρατούν για τη χρήση κοινωνικών δικτύων είναι διαρκώς αυξανόμενες. (11)

2.2 Twitter

Το Twitter είναι μια διαδικτυακή υπηρεσία κοινωνικής δικτύωσης που επιτρέπει στους χρήστες του να στέλνουν και να διαβάσουν μηνύματα κειμένου μέχρι 140 χαρακτήρες, που είναι γνωστά ως 'tweets'.

Εικόνα 2: Το λογότυπο του Twitter.

Το Twitter δημιουργήθηκε το Μάρτιο 2006 από τον Jack Dorsey και τον Ιούλιο 2006 το κοινωνικό δίκτυο ξεκίνησε κανονικά τη δημόσια λειτουργία του. Έχει έδρα στο Σαν Φρανσίσκο με επιπλέον servers και γραφεία στη Νέα Υόρκη, Βοστώνη και Σαν Αντόνιο.

Η υπηρεσία κέρδισε πολύ γρήγορα δημοτικότητα παγκοσμίως, με πάνω από 500 εκατομμύρια εγγεγραμμένους χρήστες μέχρι το 2012, οι οποίοι αναρτούσαν πάνω από 340 εκατομμύρια tweets καθημερινά και πάνω από 1,6 δισ. ερωτήματα αναζήτησης ανά ημέρα. Από την έναρξή του, το Twitter έχει γίνει μία από τις δέκα πιο δημοφιλείς ιστοσελίδες στο διαδίκτυο και έχει περιγραφεί ως 'το SMS του Διαδικτύου'.

Μηνύματα Tweet

Οι μη εγγεγραμμένοι χρήστες μπορούν να διαβάσουν tweets, ενώ οι εγγεγραμμένοι χρήστες μπορούν να δημοσιεύουν tweets μέσω της ιστοσελίδας, μέσω SMS σε ορισμένες χώρες ή μέσω εφαρμογών για κινητές συσκευές.

Τα Tweets είναι δημόσια από προεπιλογή, αλλά οι συγγραφείς τους μπορούν να περιορίσουν την εμφάνιση των μηνυμάτων τους μόνο σε οπαδούς (follower) τους. Ως κοινωνικό δίκτυο, το Twitter περιστρέφεται γύρω από την αρχή των οπαδών. Όταν επιλέξετε να ακολουθήσετε ένα άλλο χρήστη του Twitter, tweets του χρήστη εμφανίζονται σε αντίστροφη χρονολογική σειρά στην κύρια σελίδα σας στο Twitter. Αν ακολουθήσετε 20 άτομα, θα δείτε ένα μίγμα των tweets στην σελίδα σας από τα 20 άτομα που ακολουθάτε.

Re-tweet

Ένα από τα πιο ενδιαφέροντα χαρακτηριστικά του κοινωνικού δικτύου είναι η δυνατότητα του re-tweet. Το re-tweet είναι όταν αναδημοσιεύει κάποιος το tweet κάποιου άλλου. Αυτό συμβαίνει διότι το αρχικό tweet είναι ενδιαφέρον ή/και σημαντικό και κάποιος θέλει να το προωθήσει στους ανθρώπους που τον ακολουθούν.

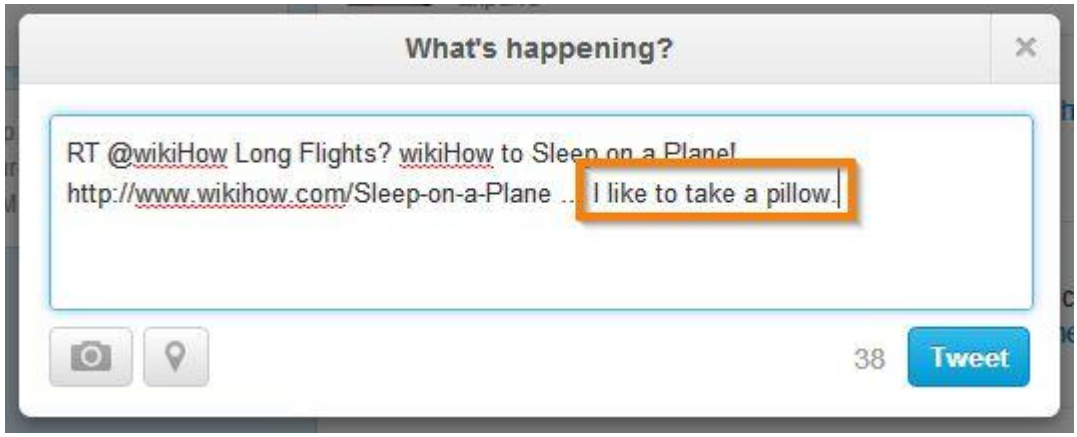
Το Twitter δίνει την δυνατότητα στους χρήστες του να κάνουν re-tweet με δύο τρόπους, τον αυτόματο και τον χειροκίνητο. Ο καθένας εκτελείται με διαφορετικό τρόπο και έχουν κάποιες μικροδιαφορές στο πώς εμφανίζονται στους χρήστες.

Ο χειροκίνητος τρόπος για να κάνεις re-tweet είναι ο πιο παλιός και αυτός που δίνει στους χρήστες την ελευθερία να εκφράσουν την αναδημοσίευση του μηνύματός τους όπως αυτοί επιθυμούν. Πιο συγκεκριμένα, δεν υπάρχει αυστηρή μορφή για να κάνεις re-tweet, απλά κάποια μορφή που επικράτησε λόγω του ότι ήταν η πιο συχνά χρησιμούποιημενη. Παρόλα

αυτά, λιγότερο δημοφιλείς μορφές για re-tweet εξακολουθούν να χρησιμοποιούνται από κάποιους χρήστες.

Παρακάτω παραθέτουμε τα βήματα για να κάνει κάποιος χειροκίνητα re-tweet χρησιμοποιώντας την πιο δημοφιλή μορφή: (12)

1. Ξεκινάς το tweet με το πρόθεμα 'RT'.
2. Συνεχίζεις με το να προσθέσεις τον χρήστη που δημοσίευσε το αρχικό tweet. Για παράδειγμα, αν το αρχικό tweet ανήκει στον χρήστη 'User' στο tweet προσθέτεις '@User'.
3. Αντιγράφεις το αρχικό μήνυμα που θέλεις να αναδημοσιεύσεις, έτσι ώστε να το δουν οι χρήστες που σε ακολουθούν.
4. Προσθέτεις κάποιο σχόλιο αν υπάρχουν διαθέσιμοι χαρακτήρες από το όριο των 140 που θέτει το Twitter.



Εικόνα 3: Η διεπαφή για την δημιουργία ενός χειροκίνητου re-tweet μέσα από την ιστοσελίδα του Twitter.

Παραπάνω φαίνεται η διεπαφή μέσα από την σελίδα του Twitter για να δημοσιεύσει κάποιος ένα νέο re-tweet. Το τετραγωνισμένο μέρος του κειμένου είναι το σχόλιο του χρήστη για το αρχικό tweet. Αυτό το κομμάτι μπορεί να διαχωριστεί από το αρχικό μήνυμα με κάποιους συγκεκριμένους χαρακτήρες όπως '|', '<<', '..' ή κάποιους χαρακτήρες της επιλογής του χρήστη. Για να είναι πιο ξεκάθαρο το σχόλιο, το Twitter προτείνει στους χρήστες του να τοποθετούν τα σχόλια στην αρχή του tweet. (13)

Επιπλέον, το Twitter επιτρέπει την αυτόματη αναδημοσίευση των tweets μέσω της διεπαφής τους όπως φαίνεται παρακάτω. Το μειονέκτημα του αυτόματου re-tweet είναι πως δε δίνει την δυνατότητα στον χρήστη να προσθέσει ένα δικό του σχόλιο στο μήνυμα που αναδημοσιεύει.

Εικόνα 4: Η διεπαφή για την δημιουργία ενός αυτόματου re-tweet μέσα από την ιστοσελίδα του Twitter.

Reply

Κατά αναλογία με την δυνατότητα για re-tweet, υπάρχει και η δυνατότητα για απάντηση σε κάποιο μήνυμα tweet. Για να απαντήσει κάποιος σε ένα μήνυμα tweet, μέσω της διεπαφής του Twitter, πρέπει να βρει το μήνυμα και να επιλέξει το κουμπί 'Reply' όπως φαίνεται παρακάτω. (14)

Εικόνα 5: Η διεπαφή για την δημιουργία μιας απάντησης σε ένα tweet μέσα από την ιστοσελίδα του Twitter.

Hashtag

Οι χρήστες μπορούν να ομαδοποιήσουν τις θέσεις τους, το θέμα τους ή το είδος του μηνύματος με τη χρήση των hashtags (ετικέτες), δηλαδή λέξεις ή φράσεις με πρόθεμα ένα χαρακτήρα '#'.

Μια λέξη, φράση ή το θέμα που έχει 'κολληθεί' σε μεγαλύτερο ποσοστό από ό, τι άλλες ετικέτες λέγεται ότι είναι ένα θέμα που αποτελεί τάση (trend). Τέτοια θέματα γίνονται δημοφιλή, είτε μέσα από μια συντονισμένη προσπάθεια από τους χρήστες, είτε λόγω ενός γεγονότος που ωθεί τους ανθρώπους να μιλήσουν για ένα συγκεκριμένο θέμα. Τα θέματα αυτά βοηθούν το Twitter και τους χρήστες του να καταλάβουν τι συμβαίνει στον κόσμο.

Επιβεβαιωμένοι λογαριασμοί

Τον Ιούνιο του 2008, το Twitter ξεκίνησε ένα πρόγραμμα επαλήθευσης των λογαριασμών του, επιτρέποντας σε διασημότητες να επιβεβαιώσουν τους λογαριασμούς τους. Με την αύξηση των χρηστών του κοινωνικού δικτύου όλο και περισσότερες διασημότητες άρχισαν να χρησιμοποιούν το κοινωνικό δίκτυο. Αυτό είχε σαν αποτέλεσμα την εμφάνιση και πολλών ψεύτικων λογαριασμών που παρουσιάζονταν σαν κάποιο γνωστό πρόσωπο ενώ δεν ήταν. Το βήμα αυτό προοριζόταν για να βοηθήσει τους χρήστες να ελέγξουν ποιοι λογαριασμοί διασημοτήτων δημιουργήθηκαν από τις προσωπικότητες τις ίδιες, δηλαδή δεν ήταν πλαστοί.

Δημογραφικά

Το 2009, το Twitter χρησιμοποιήθηκε κυρίως από τους ενήλικες μεγαλύτερης ηλικίας οι οποίοι μπορεί να μην είχαν χρησιμοποιήσει άλλους δικτυακούς τόπους κοινωνικής προηγουμένως, βρήκε μια μελέτη των κοινωνικών μέσων μαζικής ενημέρωσης.

Επιπλέον, σύμφωνα με μια μελέτη από την Sysomos τον Ιούνιο του 2009, οι γυναίκες αποτελούν ένα ελαφρώς μεγαλύτερο ποσοστό των χρηστών του Twitter από τους άνδρες με 53% σε σχέση με το 47%. Επίσης, η συγκεκριμένη έρευνα ανάφερε πως το 5% των

χρηστών αντιπροσώπευαν το 75% του συνόλου των δραστηριοτήτων και ότι η Νέα Υόρκη έχει περισσότερους χρήστες του Twitter από άλλες πόλεις.

Σύμφωνα με έρευνα της Quancast μέχρι την 3η Σεπτεμβρίου του 2009, 27 εκατομμύρια άνθρωποι στις ΗΠΑ χρησιμοποιούν το Twitter. Ενδιαφέρον παρουσιάζει πως το 63% των χρηστών του Twitter είναι κάτω από 35 χρονών. Ακόμη, 60% των χρηστών είναι Καυκάσιοι, αλλά υπάρχει υψηλότερος από το μέσο όρο σε σύγκριση με άλλες ιστοσελίδες συμμετοχή Αφρικοαμερικάνων στο 16% και των ισπανόφωνων στο 11%.

Στις 7 Σεπτεμβρίου 2011, το Twitter ανακοίνωσε ότι έχει 100 εκατομμύρια ενεργούς χρήστες που μπαίνουν στο κοινωνικό δίκτυο τουλάχιστον μία φορά το μήνα και 50 εκατομμύρια ενεργούς χρήστες κάθε μέρα.

2.3 Φήμες (Ακούσια / Εκούσια Παραπληροφόρηση)

Η έλλειψη σαφών ορισμών αναφορικά με την παραπληροφόρηση μπορεί να δημιουργήσει σύγχυση στους μελετητές οι οποίοι χρησιμοποιούν κάποιους όρους καθώς και ερευνητές που είναι νέοι σε ένα πεδίο.

Η ακούσια και εκούσια παραπληροφόρηση είναι εξαιρετικά παραδείγματα αυτών των περιπτώσεων: χρησιμοποιούνται σε ένα μεγάλο αριθμό διαφορετικών πεδίων, όπως η ψυχολογία, η φιλοσοφία, η επιστήμη των υπολογιστών, με αποτέλεσμα διακυμάνσεις στην κατανόηση και τη χρήση των όρων αυτών. Μερικές φορές οι όροι χρησιμοποιούνται αδιακρίτως και αντιμετωπίζονται ως εάν δεν υπήρχε διαφορά.

Πληροφορία

Πριν προχωρήσουμε παρακάτω, θα προσπαθήσουμε να ορίσουμε τι σημαίνει πληροφορία για να είναι πιο ξεκάθαροι οι ορισμοί που θα αναλυθούν στην συνέχεια.

Μετά από μια μακρά εξέταση του τι δεν είναι πληροφορία, ο Fox το 1983 ανέφερε ότι:
'Οι πληροφορίες που μεταφέρονται από μια φράση, ή ένα σύνολο από φράσεις, προσδιορίζονται με το νόημα τους, και ως εκ τούτου καθορίζονται σε σχέση με την έννοια τους. […] Χωρίς έναν επαρκή ορισμό του τι σημαίνει νόημα, δεν διαθέτουμε επαρκή μέσα για να προσδιορίσουμε με ακρίβεια ποιες πληροφορίες μεταφέρονται από μια συγκεκριμένη πρόταση ή σύνολο προτάσεων.'

Εδώ, ο Fox υπογράμμισε το ρόλο της έννοιας για τον καθορισμό του τι είναι πληροφορία. Αυτό είναι σημαντικό επειδή οι δηλώσεις χωρίς νόημα μπορεί να μην είναι πληροφορίες. Για παράδειγμα, εάν η Jill μίλησε με τον Frank σε μια γλώσσα που δεν καταλαβαίνουν, οι δηλώσεις της Jill δεν αποτελούν πληροφορία γιατί η γλώσσα δεν έχει κανένα νόημα για τον Frank. Εάν η δήλωση *δεν παρέχει νέες ή άγνωστες πληροφορίες* για τον παραλήπτη ή αν η δήλωση *δεν έχει νόημα* για τον δέκτη, η δήλωση δεν αποτελεί πληροφορία. (15)

Ακούσια παραπληροφόρηση

Ο Fox περιγράφει την *ακούσια παραπληροφόρηση* ως εξής:
'Δεδομένου ότι οι πληροφορίες μπορεί να είναι ψευδείς, βλέπουμε ότι η ακούσια παραπληροφόρηση είναι ένα είδος πληροφορίας, όπως ακριβώς η ακούσια παραπληροφόρηση είναι ένα είδος ενημέρωσης […] Η ενημέρωση δεν απαιτεί την αλήθεια και οι πληροφορίες δεν χρειάζεται να είναι αλήθεια. Αλλά η ακούσια παραπληροφόρηση απαιτεί ψεύδος, και η παραπληροφόρηση πρέπει να είναι ψευδής.'

Όταν δηλώνει ότι 'η παραπληροφόρηση είναι ένα είδος πληροφορίας' όρισε την σχέση με σαφήνεια: παραπληροφόρηση, αν και ψευδής, εξακολουθεί να είναι πληροφορία και ως εκ τούτου, μπορεί ακόμα και να είναι χρήσιμη.

Πώς γίνεται να ενημερωθούμε από ψευδείς δηλώσεις; Ένα πιθανό σενάριο είναι ο ομιλητής να αποκαλύψει κάποιες πληροφορίες (ίσως λανθασμένες) ή μπορεί να υπονοεί κάποιες πληροφορίες σχετικά με την κατάσταση.

Επιπλέον, σε ένα άρθρο σχετικά με τη φύση των πληροφοριών, ο Losee το 1997 παρουσίασε τις τρεις μορφές της ακούσιας παραπληροφόρησης:

1. Η παραπληροφόρηση μπορεί να είναι απλώς πληροφορίες που είναι *ελλιπείς, απόκρυφες, αμφίσημες, παραμορφωμένες* ή και *αβέβαιες*.
2. Πληροφορίες που δε δικαιολογούνται, διότι κάποιος βασίζεται σε 'λανθασμένους λόγους' και έτσι θεωρείται παραπληροφορημένος.
3. Η παραπληροφόρηση είναι πληροφορίες που είναι *μερικώς* ή *ολικώς ψευδείς*.

Εκούσια παραπληροφόρηση

Το Oxford English Dictionary ορίζει την εκούσια παραπληροφόρηση ως:
'τη διάδοση *εσκεμμένων* ψευδών πληροφοριών, π.χ. όταν παρέχονται από μια κυβέρνηση ή τον πράκτορά της σε μια ξένη δύναμη ή τα μέσα μαζικής ενημέρωσης, με την πρόθεση να επηρεάσουν τις πολιτικές απόψεις ή τις απόψεις εκείνων που τις λαμβάνουν.'

Με πιο απλά λόγια, η διαφορά της ακούσιας από την εκούσια παραπληροφόρηση βρίσκεται αποκλειστικά στην *πρόθεση* του ομιλητή. Η πρόθεση του ομιλητή, ωστόσο, είναι συχνά άγνωστη σε άλλα άτομα. Δεν μπορούμε να αρνηθούμε την παρουσία προθέσεων στην επικοινωνία, αλλά να εξαρτάται ο ορισμός κάποιων όρων μόνο από την, συχνά-άγνωστη, πρόθεση φαίνεται κάπως περιοριστικό.

Συνεπώς, η κατανόηση χαρακτηριστικών στοιχείων όταν υπάρχει πρόθεση για εξαπάτηση μπορεί να μας βοηθήσει να αναγνωρίσουμε και να ορίσουμε την εκούσια παραπληροφόρηση, συμφώνα με τους Karlova et al. Πιο συγκεκριμένα προτείνουν τις ακόλουθες κατηγορίες χαρακτηριστικών:

1. **Φυσικά χαρακτηριστικά**
2. **Λεκτικές νύξεις**
3. **Γραπτά χαρακτηριστικά**

Φήμες

Ως φήμες μπορούμε να ορίσουμε τόσο την εκούσια όσο και την ακούσια παραπληροφόρηση. Παρόλο που πολλοί ερευνητές θεωρούν ως φήμη μόνο την εκούσια παραπληροφόρηση, στην συνέχεια αυτής της εργασίας θεωρούμε ως φήμη και ως λανθασμένη πληροφορία οποιοδήποτε είδος παραπληροφόρησης.

2.4 Το πρόβλημα

Τα Μέσα Μαζικής Ενημέρωσης ανέκαθεν είχαν πρόβλημα με το θέμα της ακούσιας και εκούσιας παραπληροφόρησης.

Το πιο κλασσικό παράδειγμα παραπληροφόρησης από τα παραδοσιακά μέσα, ήταν όταν την Κυριακή 30 Οκτωβρίου 1938, εκατομμύρια ακροατές του ραδιοφώνου έμειναν έκπληκτοι όταν το ραδιόφωνο του CBS ανακοίνωσε τα 'νέα' για την εισβολή των Αριανών στη Γη. Η ραδιοφωνική προσαρμογή του μυθιστορήματος 'Ο Πόλεμος των Κόσμων' του H.G. Wells προκάλεσε πανικό στην Αμερική. Χιλιάδες άνθρωποι κάλεσαν την αστυνομία για την προσγείωση Αριανών στο κεντρικό New Jersey. Ορισμένοι κάτοικοι φόρτωσαν μέχρι και τα αυτοκίνητα τους και εγκατέλειψαν τα σπίτια τους, καθώς το ραδιόφωνο μετέδιδε δήλωση από έναν ηθοποιό ο οποίος ακουγόταν έντονα σαν τον πρόεδρο Franklin D. Roosevelt. (16)

Παρόλο που σήμερα, στην εποχή του Διαδικτύου, κάτι τέτοιο θα έλεγε κάποιος πως είναι δύσκολο να συμβεί, κάθε μέρα κυκλοφορούν στο διαδίκτυο πληροφορίες χωρίς να επαληθεύονται. Χαρακτηριστικό παράδειγμα ήταν όταν στις 8 Μαΐου 2012, ο blogger Nate St. Pierre έγραψε ένα διασκεδαστικό άρθρο-φάρσα ότι ο πρόεδρος των ΗΠΑ Αβραάμ Λίνκολν είχε κατάθεσει δίπλωμα ευρεσιτεχνίας για το Facebook το 1845. Η ιστορία περιλάμβανε σκόπιμα μερικά στοιχεία που έδειχναν πως δεν πρόκειται για αληθινή ιστορία όπως την παραπομπή στον φαρσέρ P.T. Barnum όπως και μια κακής ποιότητας φωτογραφία της εφημερίδας Springfield Gazette της 24ης Δεκεμβρίου 1845. Για να προωθήσει το άρθρο του, έκανε μια δημοσίευση στο Facebook και μια στο Twitter. Μέσα σε 36 ώρες, πήρε 16.000 Facebook Like και 104.463 μοναδικές προβολές του άρθρου του στο blog του. Επίσης, του είχαν ήδη πάρει συνέντευξη δημοσιογράφοι από το CNN, την The Atlantic και την Washington Post. (17)


SPRINGFIELD GAZETTE.

Εικόνα 6: Η κακής ποιότητας φωτογραφία της εφημερίδας Springfield Gazette.

Το Internet δυστυχώς, με την αποτελεσματικότητα και την αμεσότητα του, είναι ευάλωτο στους κινδύνους της ακούσιας και εκούσιας παραπληροφόρησης. Η νέα γενιά του Παγκόσμιου Ιστού, το Web 2.0, και ειδικά οι νέες εφαρμογές του όπως τα κοινωνικά δίκτυα, έκαναν το πρόβλημα της εξάπλωσης της παραπληροφόρησης πολύ πιο έντονο. Πολλοί έχουν χαρακτηρίσει την εποχή του Web 2.0 ως την εποχή του Συλλογικού Ιστού αφού βασίζεται σε μεγάλο μέρος στο περιεχόμενο που δημιουργούν οι χρήστες. Δυστυχώς, το κοινό πολλές φορές ξεχνά πως η κάθε πληροφορία που βλέπει στο Facebook, στο

Twitter, και στο YouTube είναι πολλές φορές αναξιόπιστη με αποτέλεσμα η διάδοση λανθασμένων πληροφοριών να έχει γίνει συχνό φαινόμενο στα σύγχρονα κοινωνικά δίκτυα. Παρακάτω παραθέτουμε μερικά σημαντικά παραδείγματα διάδοσης λανθασμένων πληροφοριών που δείχνουν την έκταση του προβλήματος που υπάρχει στα κοινωνικά δίκτυα: (18)

- Το 2006, η χρήστης 'lonelygirl15' εμφανίστηκε ως ένα 16-χρονο συγχυσμένο παιδί που δεν πήγαινε σχολείο αλλά εκπαιδευόταν στο σπίτι. Το όνομά της ήταν 'Bree' και τα βίντεο της στο YouTube ήταν εξαιρετικά δημοφιλή με πάνω από ένα εκατομμύριο αναπαραγωγές. Για τέσσερις μήνες, εκείνη ξεγελούσε τους θεατές του καναλιού της με το να πιστέψουν πως είχε πραγματικά προβλήματα με τους αποξενωμένους γονείς της και την δυσλειτουργική οικογένεια της. Μέχρι που ο δημοσιογράφος των Los Angeles Times, Richard Rushfield, αποκάλυψε ότι η 'Bree' ήταν η 19-χρονη ηθοποιός Jessica Lee Rose. Ο Rushfield ανέφερε πως κατάφερε να εντοπίσει την αληθινή ταυτότητα της διότι τα e-mail που αποστέλλονταν από τον λογαριασμό της 'lonelygirl15' έρχονταν μέσα από τα γραφεία του πρακτορείου ταλέντων Creative Artists Agency στο Beverly Hills.

- Το 2009, μια ομάδα μαθητών του Millburn High School στο New Jersey δημιούργησε ένα ψεύτικο λογαριασμό στο Facebook για ένα φανταστικό νέο φοιτητή με το όνομα 'Lauren' στο σχολείο τους. Σε σύντομο χρονικό διάστημα περίπου 120 μαθητές και 55 άλλοι την είχαν προσθέσει ως φίλο στο κοινωνικό δίκτυο.

- Το 2010, ο καθηγητής του Πανεπιστημίου της Indiana, Filippo Menczer και άλλοι ερευνητές ξεκίνησαν το Project Truthy για τον εντοπισμό λανθασμένων πληροφοριών στο Διαδίκτυο σχετικές με πολιτικά κινήματα και επιχειρήσεις. Βρήκαν ενδείξεις ότι οι πολιτικές καμπάνιες και ομάδες ειδικών συμφερόντων χρησιμοποιούν ψεύτικους λογαριασμούς στο Twitter για να δημιουργήσουν την εσφαλμένη εντύπωση της έντονης κινητικότητας στα κοινωνικά δίκτυα. Επαναλαμβανόμενα μηνύματα από ένα αριθμό ψεύτικων χρηστών παρουσιάστηκαν ως 'trending' θέματα στο Twitter και επηρέασαν τελικά τα αποτελέσματα αναζήτησης του Google.

- Το 2011, ένας φοιτητής στο γυμνάσιο Rancho Bernardo δημιούργησε ένα λογαριασμό στο Facebook χρησιμοποιώντας το όνομα ενός άλλου εφήβου στο ίδιο σχολείο και δημοσίευσε απειλές μαζικών δολοφονιών στο σχολείο. Η αστυνομία συνέλαβε τον φοιτητή για την πραγματοποίηση τρομοκρατικών απειλών και για πλαστογράφηση ταυτότητας μέσω του Διαδικτύου.

- Το 2012, ο μεγιστάνας των μέσων ενημέρωσης Ρούπερτ Μέρντοχ εγγράφηκε ως χρήστης του Twitter και άρχισε να ακολουθεί τέσσερα άτομα στο κοινωνικό δίκτυο όπως τον συνιδρητή και CEO της Google, Larry Page. Εν αγνοία του ο Μέρντοχ, ακολούθησε ένα ψευτικο προφίλ του Larry Page. Ήταν ένας λογαριασμός-παρωδία που δημιουργήθηκε από φοιτητές του Virginia Tech για ένα πανεπιστημιακό πρόγραμμα.

Σοβαρές συνέπειες της παραπληροφόρησης

Η εκούσια και ακούσια παραπληροφόρηση μπορεί να έχει μια πιο σοβαρή συνέπεια από απλά ένα καλό γέλιο ή μια αστεία είδηση.

- Τον Νοέμβριο του 2010, η Νικαράγουα για να δικαιολογήσει μια επιδρομή σε μια αμφισβητούμενη περιοχή με την Κόστα Ρίκα αναφέρθηκε στα σύνορά της στον χάρτη της Google.

- Τον Μάιο του 2012, το Ιράν απείλησε να μηνύσει τη Google για την μη επισήμανση του Περσικού Κόλπου και των γειτονικών υδάτινων μαζών, συμπεριλαμβανομένου του Κόλπου του Ομάν, την Αραβική Θάλασσα, τον Κόλπο του Άντεν και την Ερυθρά Θάλασσα.

- Το FBI έχει ξεκινήσει έρευνα για το πώς χάκερ πήραν τον έλεγχο του λογαριασμού του Associated Press στο Twitter και κατάφεραν να στείλουν ψεύτικα μηνύματα με απρόβλεπτες συνέπειες στις χρηματοπιστωτικές αγορές.

  Ένα ψεύτικο tweet στο λογαριασμό του Associated Press ανέφερε ότι υπήρξαν δύο εκρήξεις στο Λευκό Οίκο και πως ο πρόεδρος των Η.Π.Α. Μπαράκ Ομπάμα είναι μέσα στους τραυματίες.

  Παρά το γεγονός ότι ο Λευκός Οίκος χρειάστηκε μόνο τρία λεπτά για να εκδώσει μια διάψευση, σχεδόν όλες οι χρηματοπιστωτικές αγορές των ΗΠΑ έπεσαν απότομα καθώς ένας χρηματιστής περιέγραψε την συγκεκριμένη στιγμή δημοσίευσης του μηνύματος ως ‘καθαρό χάος’. Όπως φαίνεται και στο μήνυμα που δημοσιεύτηκε χρειάστηκαν μόλις 3 λεπτά για να γίνει το συγκεκριμένο μήνυμα re-tweet 3000 φορές. (19)

Εικόνα 7: Το ψεύτικο tweet που δημοσιεύτηκε στο λογαριασμό του Associated Press.

Συμπέρασμα

Είναι ξεκάθαρο, μετά από τα προαναφερθέντα παραδείγματα, πως το πρόβλημα της διάδοσης ψεύτικων πληροφοριών, κυρίως μέσα από τα κοινωνικά δίκτυα, είναι ένα πρόβλημα που λαμβάνει όλο και μεγαλύτερες διαστάσεις. Παρόλο που στις πρώτες μέρες του Διαδικτύου η πληροφόρηση μέσω αυτού δεν ήταν ο κυρίαρχος τρόπος για ενημέρωση των ανθρώπων, πλέον έχει γίνει. Αυτό συνεπάγεται πως οποιαδήποτε διάδοση λανθασμένης πληροφορίας, είτε εκούσιας είτε ακούσιας, μέσω του Διαδικτύου και πιο συνηθισμένα μέσω των κοινωνικών δικτύων, μπορεί να έχει απρόβλεπτες συνέπειες.

3. Εντοπισμός δεδομένων και πηγών παραπληροφόρησης στα κοινωνικά δίκτυα

Η έρευνα σε κοινωνικά δίκτυα σχετικά με τον εντοπισμό των πηγών ή και των δεδομένων παραπληροφόρησης είναι ένας σχετικά νέος τομέας στην έρευνα που άρχισε να αναπτύσσεται μετά την πρόσφατη επιτυχία των κοινωνικών δικτύων. Παρακάτω παραθέτουμε μερικές χαρακτηριστικές δουλειές άλλων ερευνητών σχετικές με το θέμα της παρούσας εργασίας.

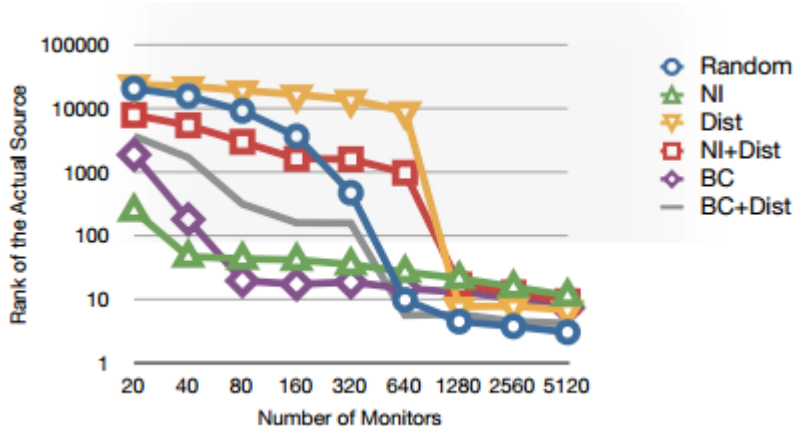
3.1 Επισκόπηση τρέχουσας τεχνολογικής κατάστασης

Οι Seo et al. (20) προσπάθησαν να λύσουν το πρόβλημα της εύρεσης φημών και των πηγών τους σε κοινωνικά δίκτυα. Για να το πετύχουν αυτό, χρησιμοποιούν κόμβους monitor οι οποίοι ενημερώνουν αν *έφτασε* μια φήμη (possitive monitor) ή όχι (negative monitor) στον συγκεκριμένο κόμβο. Προτείνουν διάφορες μεθοδολογίες για την επιλογή των κόμβων monitor, που έχουν διαφορετική απόδοση ανάλογα με τα δεδομένα εισόδου.

Για να βρουν την πηγή μιας φήμης, για κάθε κόμβο υπολογίζουν τέσσερις μετρικές:

1. Reachability to all positive monitors,
2. Distance to positive monitors,
3. Reachability to negative monitors,
4. Distance to negative monitors

και έπειτα ταξινομούν λεξικογραφικά όλους τους κόμβους βάσει των παραπάνω μετρικών. Στην ιδανική περίπτωση ο κόμβος πηγή της φήμης θα ήταν πρώτος στην κατάταξη αυτή.



Εικόνα 8: Η τελική θέση της αληθινής πηγής μιας φήμης σε σχέση με το πλήθος των Monitor που χρησιμοποιήθηκαν και την τεχνική επιλογής των κόμβων Monitor.

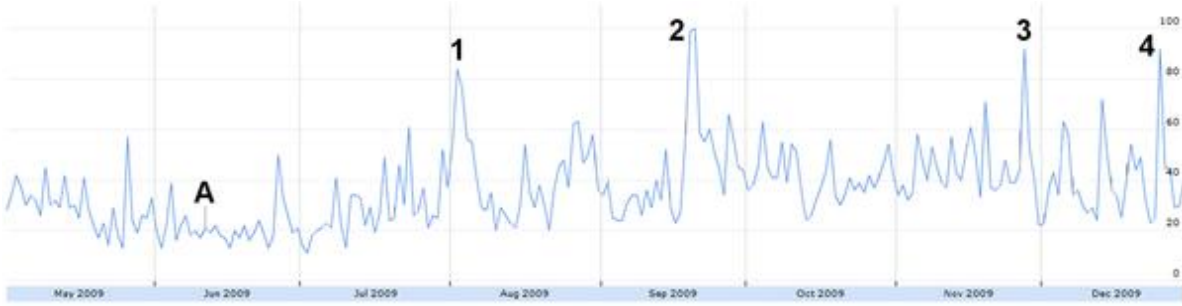
Στο παραπάνω σχήμα παρουσιάζεται η θέση στην τελική κατάταξη της αληθινής πηγής μιας φήμης για διάφορους τρόπους επιλογής των κόμβων monitor. Φαίνεται ξεκάθαρα πως η απόδοση του αλγόριθμου βελτιώνεται με την αύξηση του πλήθους των κόμβων monitor.

Για να αποφανθούν αν ένα μήνυμα είναι φήμη ή όχι, χρησιμοποιούν την *υπόθεση ότι οι φήμες ξεκινούν από ένα μικρό πλήθος πηγών*, αντίθετα με την αληθινή πληροφορία η οποία ξεκινά από ένα μεγάλο πλήθος ανεξάρτητων μεταξύ τους πηγών. Προτείνουν έναν άπληστο αλγόριθμο για τον εντοπισμό του μεγέθους του συνόλου των πηγών. Τα αποτελέσματα τους δείχνουν πως όταν η διαφορά του πλήθος των πηγών μιας φήμης και μιας αληθινής πληροφορίας είναι μεγάλη, τότε ο αλγόριθμός τους πετυχαίνει ικανοποιητικά αποτελέσματα.

Οι Qazvinian et al. (21) προσπάθησαν επίσης να λύσουν το πρόβλημα του εντοπισμού της διάδοσης λανθασμένης πληροφορίας σε ένα κοινωνικό δίκτυο και πιο συγκεκριμένα στο Twitter. Δοκίμασαν να κατηγοριοποιήσουν χειροκίνητα 10,000 tweets που αφορούσαν γνωστές φήμες και έπειτα εφάρμοσαν μια *γραμμική στατιστική συνάρτηση* για να ανιχνεύσουν περεταίρω φήμες. Για να βελτιώσουν την απόδοση της συνάρτησης τους χρησιμοποίησαν ακόμη πληροφορίες από την δομή του κοινωνικού δικτύου και από χαρακτηριστικά όπως τα hashtags και τις διευθύνσεις URL.

Οι Mendoza et al. (22) μελέτησαν tweets σχετικά με τον σεισμό στην Χιλή το 2010 και βρήκαν πως οι φήμες και η αληθινή πληροφορία γίνεται re-tweet με διαφορετικό τρόπο. Πιο συγκεκριμένα, έφτασαν στο συμπέρασμα πως οι χρήστες τείνουν να αμφιβάλουν περισσότερο για τις φήμες, ενώ επιβεβαιώνουν τις πληροφορίες όταν αυτές πρόκειται για αληθινά γεγονότα. Προτείνουν δηλαδή πως μπορείς να ξεχωρίσεις τις φήμες από τα υπόλοιπα tweets *με βάση τα σχόλια των χρηστών*.

Οι Chew et al. (23) έκαναν μια προσπάθεια για ανάλυση κοινωνικών δικτύων για τον ιό H1N1 στον οποίο αναφέρουν διάφορους τρόπους για την ανάλυση των tweets. Στα πλαίσια της προσπάθειας τους αυτής αναγνωρίζεται η ανάγκη ενός λεξικού με βάση το οποίο αναζητούν και κατηγοριοποιούν τα διάφορα μηνύματα του κοινωνικού δικτύου. Για την δημιουργία αυτού του λεξικού βασίζονται στο *περιεχόμενο του tweet*, στον *τρόπο έκφρασης* του και στον *τύπο της διεύθυνσης URL* που το συνοδεύει. Επίσης, χρησιμοποιούν επιπλέον τρόπους στην κατηγοριοποίηση των tweets όπως την *αναγνώριση emoticons* και *χαρακτηριστικών φράσεων* που συνδέονται με κάποια συγκεκριμένη διάθεση.



Εικόνα 9: Το ποσοστό των tweets που εκφράζουν λανθασμένη πληροφορία. Οι κορυφές που είναι σημειωμένες αντιπροσωπεύουν σημαντικές φήμες που διαδίδονταν την συγκεκριμένη χρονική περίοδο.

Στο παραπάνω σχήμα παρουσιάζεται μια γραφική παράσταση του ποσοστού των tweets που εκφράζουν λανθασμένη πληροφορία. Οι κορυφές που είναι σημειωμένες αντιπροσωπεύουν σημαντικές φήμες που διαδίδονταν την συγκεκριμένη χρονική περίοδο. Η ύπαρξη των κορυφών αυτών δείχνει ότι τα κοινωνικά δίκτυα είναι πολύ ευάλωτα σε διαδόσεις φημών, ειδικά σε περιόδους κρίσεων όπως μιας πανδημίας.

Οι M. Krieck et al. (24) προσπαθούν να μελετήσουν κατά πόσο μπορούν να χρησιμοποιήσουν το Twitter σαν εναλλακτική πηγή πληροφόρησης με στόχο την πιο έγκαιρη από τα συνηθισμένα μέσα ενημέρωση για ξεσπάσματα επιδημιών και ασθενειών γενικότερα. Στην προσπάθεια τους αυτή βρίσκουν πως ανάλογα με τις λέξεις κλειδιά που χρησιμοποιούν για την κάθε ασθένεια, η ακρίβεια εντοπισμού σχετικών δημοσιεύσεων μεταβάλεται. Πιο συγκεκριμένα, βρίσκουν πως έχουν καλύτερα αποτελέσματα στον εντοπισμό σχετικών δημοσιεύσεων όταν ψάχνουν για συμπτώματα ή και για τοποθεσίες μαζί με την ασθένεια παρά όταν ψάχνουν μόνο για την ασθένεια.

3.2 Ανοικτά ζητήματα

Παρόλο που η δουλειά που έχει γίνει μέχρι τώρα στο θέμα της αναγνώρισης της διάδοσης λανθασμένης πληροφορίας είναι υποσχόμενη, θεωρούμε πως υπάρχουν σημαντικά ανοικτά ζητήματα:

- Οι Seo et al. (20) παρουσιάζουν ένα αλγόριθμο που αποδίδει καλά αλλά έχει δοκιμαστεί μόνο σε προσομοιωμένες διαδόσεις φημών σε κοινωνικά δίκτυα. Επιπλέον, προνοεί πως οι κόμβοι monitor στο κοινωνικό δίκτυο μπορούν να ξέρουν με βεβαιότητα αν μια φήμη έφτασε σε αυτούς ή όχι, κάτι που δεν μπορεί ακριβώς να εφαρμοστεί σε αληθινές συνθήκες.

- Οι Qazvinian et al. (21) παρόλο που κατηγοριοποίησαν χειροκίνητα 10,000 tweets σχετικά με το αν πρόκειται για φήμες ή όχι, δεν επιτυγχάνουν σε όλα τα πειράματα τους ικανοποιητική ακρίβεια. Επίσης, δεν μπορούν να ξέρουν πως οι νέες φήμες που θα εμφανιστούν στο μέλλον θα ταιριάζουν στο στατιστικό μοντέλο που δημιούργησαν από τα κατηγοριοποιημένα tweets τους.

- Οι Chew et al. (23) στην ανάλυση του κοινωνικού δικτύου και την ανίχνευση των δημοσιεύσεων που αφορούν διάδοση λανθασμένης πληροφορίας, εξαρτώνται αποκλειστικά από την λεξικογραφική ανάλυση των tweets χωρίς να λαμβάνουν υπόψη χαρακτηριστικά του κοινωνικού δικτύου, όπως την δυνατότητα του re-tweet.

Λαμβάνοντας υπόψη τα παραπάνω ανοικτά ζητήματα στην ήδη υπάρχουσα δουλειά στο θέμα της ανίχνευσης λανθασμένων πληροφοριών, προσπαθήσαμε να προτείνουμε μια βελτιωμένη προσέγγιση στο θέμα που να συνδυάζει κάποια από τα δυνατά σημεία των προαναφερθέντων δουλειών καθώς και να αποφεύγει, όσο είναι δυνατόν, τις αδυναμίες τους.

4. Η προτεινόμενη προσέγγιση

4.1 Μοντελοποίηση Προβλήματος

Έστω ότι έχουμε *N* στο πλήθος tweets. Τα tweets αυτά ομαδοποιούνται σε **re-tweet δέντρα**, δηλαδή δέντρα με ρίζα ένα αρχικό tweet και κόμβους όλα τα re-tweet του αρχικού μηνύματος, λαμβάνοντας υπόψιν την πηγή του re-tweet σε κάθε φάση. Αν υποθέσουμε πως r_i είναι το i-οστό re-tweet δέντρο και $s(r_i)$ είναι το πλήθος των κόμβων του i-οστού re-tweet δέντρου, τότε ισχύει πως:

$$\sum_{i=0}^{m-1} s(r_i) + C = N$$

με C να είναι το πλήθος των tweets τα οποία δεν ανήκουν σε κάποιο re-tweet δέντρο, όπως τα tweet απαντήσεις σε κάποιο μήνυμα, και *m* το πλήθος των re-tweet δέντρων.

Για κάθε θέμα που διαδίδεται στο κοινωνικό δίκτυο, μπορούν να υπάρχουν ένα ή περισσότερα re-tweet δέντρα που αναφέρονται στο ίδιο θέμα. Με αυτά τα re-tweet δέντρα, δημιουργούμε σύνολα τα οποία μελετούμε ως προς την ανεξαρτησία.

Πιο συγκεκριμένα, έστω ότι το σύνολο A αναφέρεται στο θέμα Θ^A και αποτελείται από *n* re-tweet δέντρα, δηλαδή ισχύει πως:

$$A = \{r_i, r_j, \dots\}, \quad |A| = n$$

Υποθέτουμε ότι υπάρχει περίπτωση διάδοσης λανθασμένης πληροφορίας, όπως μιας φήμης, αν το πλήθος των **ανεξάρτητων** re-tweet δέντρων του συνόλου είναι μικρότερο από ένα κατώφλι το οποίο ορίζεται κατά προτίμηση.

Δύο **re-tweet δέντρα i** και **j** θεωρούνται ανεξάρτητα αν οι ρίζες τους είναι ανεξάρτητες. Σε κάθε re-tweet δέντρο, η ρίζα αποτελεί την **πηγή** των re-tweets τα οποία περιέχει. Ορίζουμε **ως ανεξάρτητες τις ρίζες R_i και R_j δύο re-tweet δέντρων**, αν:

a. ο χρήστης που αντιστοιχεί στην R_i δεν βρίσκεται στο re-tweet δέντρο j με πηγή τον χρήστη στη ρίζα R_j εντός του συνόλου A και το ανάποδο. Σε περίπτωση που βρίσκεται, τότε συγκρίνουμε τη χρονική στιγμή κατά την οποία έκανε re-tweet (t^i_{UserRi}) με αυτήν που έκανε την αρχική δημοσίευση μηνύματος (t^i_{UserRi}) καθώς και τις χρονικές στιγμές δημοσίευσης των δύο αρχικών μηνυμάτων (t^i_{UserRi}, t^j_{UserRj}).

b. ο χρήστης που αντιστοιχεί στην R_i δε βρίσκεται σε οποιοδήποτε re-tweet δέντρο στο οποίο συνυπάρχει και ο χρήστης της ρίζας του R_j (και το ανάποδο) εντός του συνόλου A.

c. στα re-tweet δέντρα στα οποία οι χρήστες που αντιστοιχούν στις R_i και η R_j *συνυπάρχουν*, είτε δεν βρίσκονται στο ίδιο μονοπάτι από την ρίζα είτε η απόστασή τους στο μονοπάτι αυτό είναι μεγαλύτερη από ένα κατώφλι.

Συμπερασματικά, έχουμε πως ένα σύνολο Α πιθανώς να διαδίδει λανθασμένη πληροφορία αν ισχύει πως:

$$I(A) < k$$

όπου *I(x)* η συνάρτηση που επιστρέφει το πλήθος των ανεξάρτητων re-tweet δέντρων και *k* το κατώφλι που θέτουμε ως τον ελάχιστο ικανοποιητικό αριθμό ανεξάρτητων re-tweet δέντρων.

4.2 Προτεινόμενη μεθοδολογία

Η μεθοδολογία που προτείνουμε αποσκοπεί στο να αναλύσει τα διαθέσιμα tweets από το dataset μας, να δημιουργήσει δέντρα από tweets βάση των re-tweets που έχουν γίνει, να ανιχνεύσει ποια από τα δέντρα αυτά αναφέρονται στο ίδιο θέμα και τέλος να εξετάσει κατά πόσο αυτά τα σύνολα από re-tweet δέντρα είναι ανεξάρτητα. Αν τα re-tweet δέντρα που αναφέρονται στο ίδιο θέμα είναι πολλά και ανεξάρτητα μεταξύ τους, τότε είναι πιθανόν η πληροφορία που διαδίδουν να είναι αληθινή. Σε αντίθετη περίπτωση, αν τα re-tweet δέντρα που αναφέρονται στο ίδιο θέμα είναι λίγα και οι χρήστες των ριζών τους έχουν επηρεαστεί μεταξύ τους, τότε είναι πιθανόν η πληροφορία που διαδίδουν να είναι αναξιόπιστη.

Παρακάτω παρουσιάζονται αναλυτικά τα βήματα της προτεινόμενης μεθοδολογίας σε επίπεδο μοντέλου.

Βήμα 1: Δημιουργία των δέντρων re-tweet

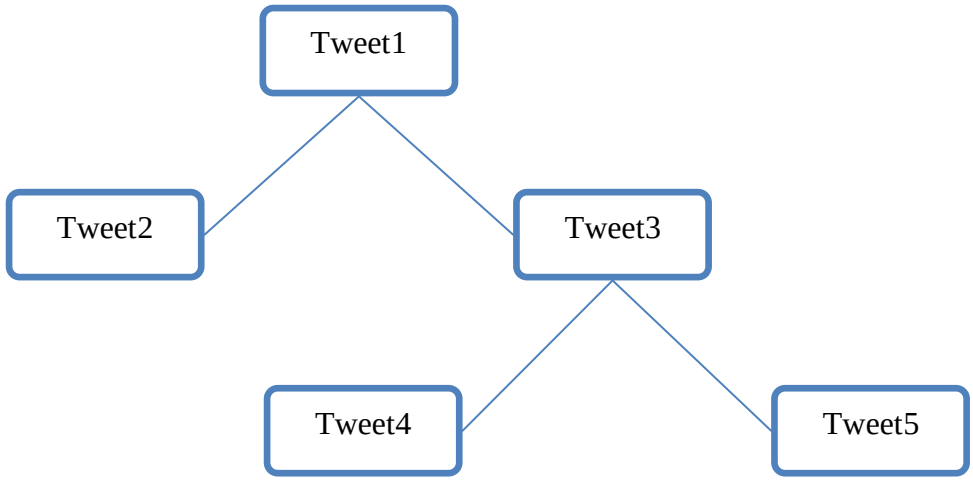
Δημιουργούμε δέντρα τα οποία αποτελούνται από κόμβους που αντιπροσωπεύουν tweets. Το κάθε δέντρο θα έχει τα παρακάτω χαρακτηριστικά:

- ο κόμβος ρίζα (root) είναι ο χρήστης που δημοσίευσε πρώτος το συγκεκριμένο tweet
- κάθε κόμβος έχει παιδιά τους χρήστες που έκαναν re-tweet αυτό το tweet

Για παράδειγμα, για τα παρακάτω tweets θα δημιουργήσουμε το ακόλουθο δέντρο.

Tweet1	User1	Date1	"This is a twitter message."
Tweet2	User2	Date2	"RT @User1 This is a twitter message."
Tweet3	User3	Date3	"RT @User1 This is a twitter message."
Tweet4	User4	Date4	"RT @User3 @User1 This is a twitter message."
Tweet5	User5	Date5	"RT @User3 @User1 This is a twitter message."

Πίνακας 1: Παράδειγμα με tweets και re-tweets για τη δημιουργία δέντρου



Εικόνα 10: Το προκύπτον δέντρο για το παράδειγμα του Πίνακα 1.

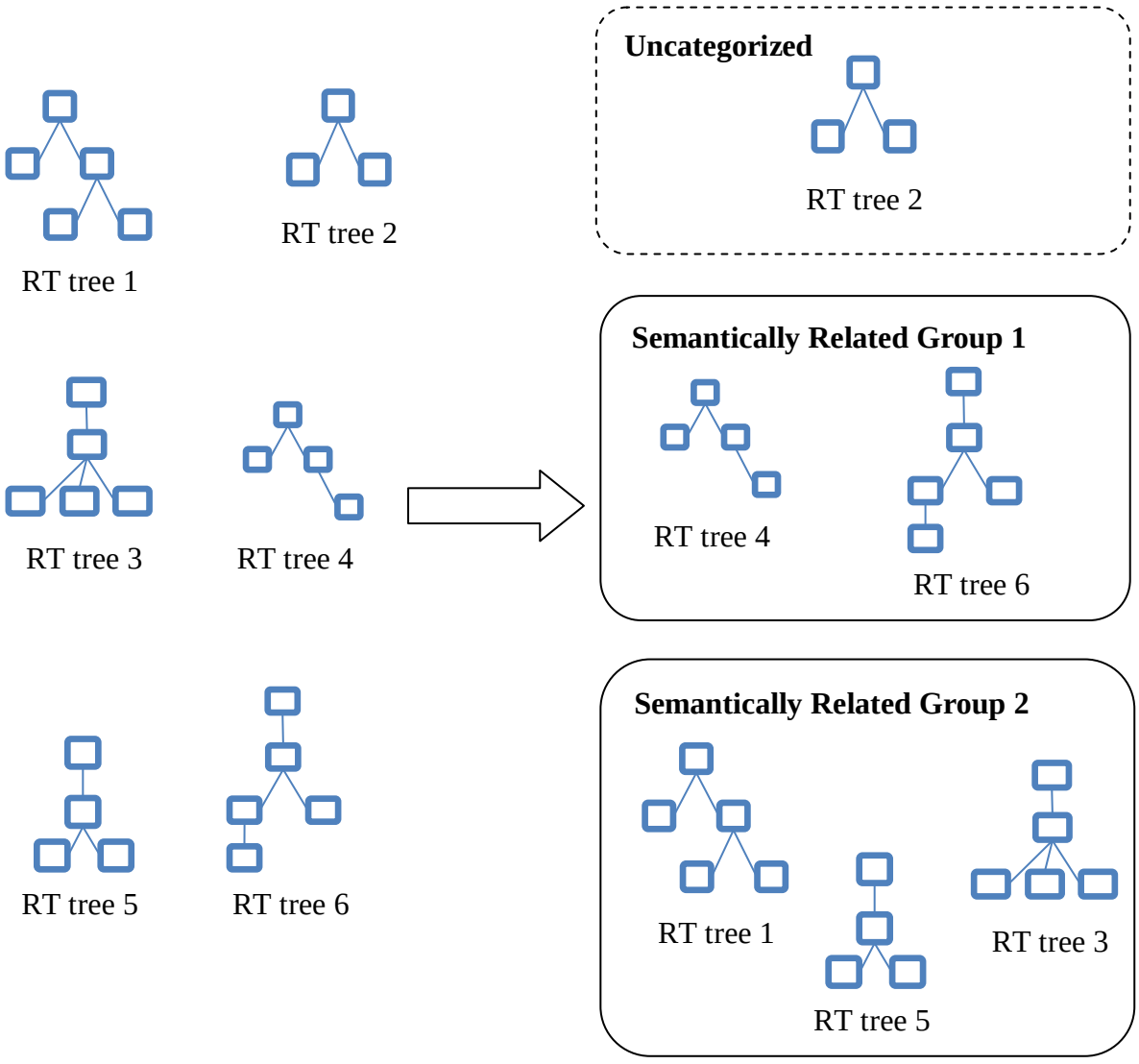
Βήμα 2: Εύρεση των σημασιολογικά κοντινών δέντρων

Βρίσκουμε ποια από τα δέντρα που έχουν δημιουργηθεί στο προηγούμενο βήμα έχουν ρίζες που είναι σημασιολογικά παρόμοιες. Αν βρούμε δέντρα με ρίζες σημασιολογικά παρόμοιες, υποθέτουμε πως τα δέντρα των re-tweets αφορούν στο ίδιο θέμα.

Αφού βρούμε 2 τουλάχιστον σημασιολογικά παρόμοια δέντρα re-tweet, δημιουργούμε ένα νέο σύνολο. Στο τέλος αυτού του βήματος θα πρέπει να έχουμε ένα αριθμό από σύνολα, που το καθένα θα περιέχει μόνο σημασιολογικά παρόμοια δέντρα re-tweet.

Η σημασιολογική ανάλυση μπορεί να υλοποιηθεί με διάφορους τρόπους που θα αναλυθούν σε μεταγενέστερο στάδιο. Να σημειώσουμε πως το συγκεκριμένο βήμα, είναι πολύ κρίσιμο λόγω της δυσκολίας να πετύχει ικανοποιητικά αποτελέσματα σε όλες γενικά τις περιπτώσεις.

Παρακάτω παραθέτουμε μια γραφική απεικόνιση μιας πιθανής εισόδου και μιας πιθανής εξόδου για το βήμα αυτό του αλγορίθμου.



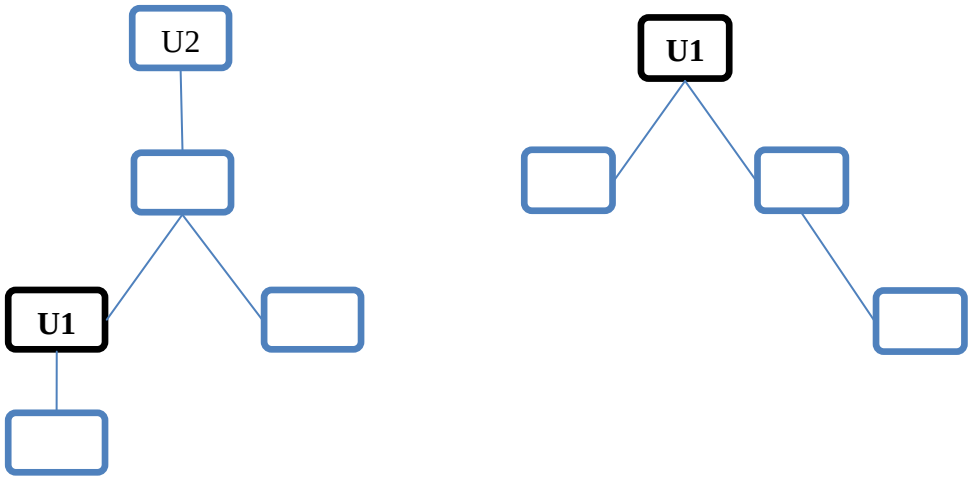
Εικόνα 11: Εύρεση των σημασιολογικά κοντινών δέντρων.

Βήμα 3: Μελέτη ανεξαρτησίας των ριζών

Για κάθε σύνολο που βρήκαμε στο βήμα 2, ερευνούμε τον βαθμό ανεξαρτησίας των ρίζών των σημασιολογικά παρόμοιων δέντρων.

Πιο συγκεκριμένα για να κρίνουμε την ανεξαρτησία 2 σημασιολογικά παρόμοιων ριζών, έστω R1 και R2, έχουμε τους ακόλουθους κανόνες – διατεταγμένους ως προς τη σοβαρότητα της εξάρτησής τους - για τους οποίους και παραθέτουμε μια γραφική απεικόνιση. Θεωρούμε πως ο χρήστης που δημοσίευσε την R1 είναι ο U1 και κατα αντιστοιχία ο χρήστης που δημοσίευσε την R2 είναι ο U2.

1. *Ελέγχουμε αν ο χρήστης που δημοσίευσε το R1 (U1) βρίσκεται στο re-tweet δέντρο της R2.*
   Αν βρίσκεται, τότε συγκρίνουμε τη ***χρονική στιγμή*** *κατά την οποία έκανε re-tweet με αυτήν που έγινε η αρχική δημοσίευση.*

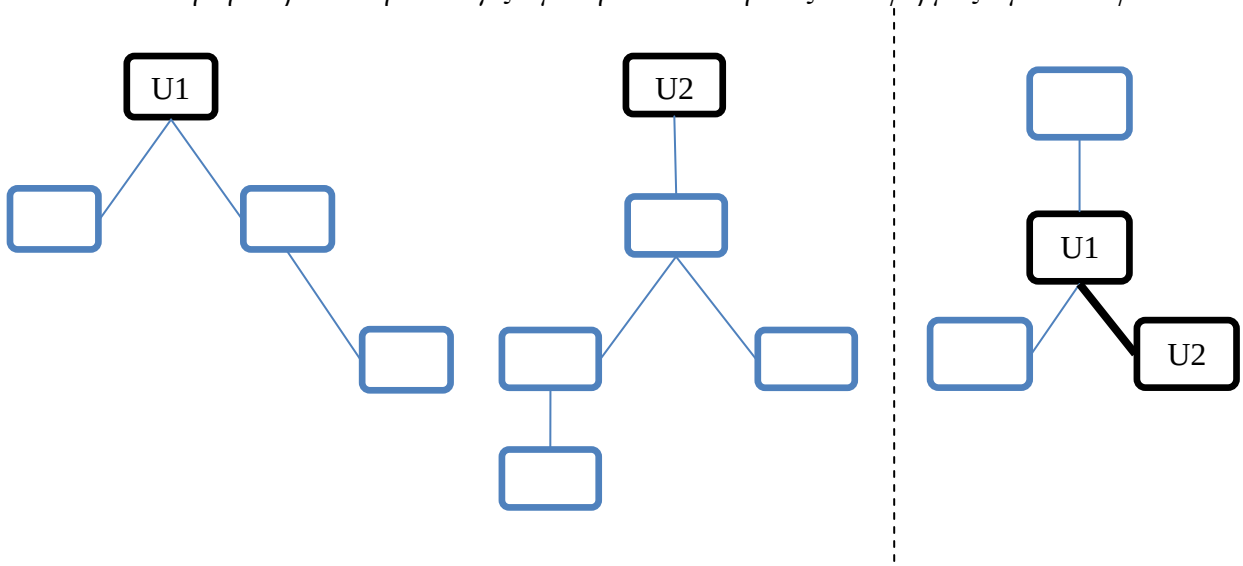


Εικόνα 12: Ο χρήστης που δημοσίευσε το R1 (U1) βρίσκεται στο re-tweet δέντρο της R2.

Αυτού του είδους η εξάρτηση μεταξύ των re-tweet δέντρων είναι η πλέον δηλωτική της εξάρτησης των ριζών των δύο δέντρων. Η ύπαρξη αυτής της εξάρτησης δείχνει πως ο χρήστης που δημοσίευσε το tweet R1, πριν ξεκινήσει το δικό του re-tweet δέντρο, είχε ήδη διαβάσει και αναδημοσιεύσει το tweet R2 το οποίο αφορούσε το ίδιο θέμα.

Εύκολα μπορούμε να συμπεράνουμε πως αυτή η εξάρτηση δείχνει πως τα δύο re-tweet δέντρα δεν είναι ανεξάρτητες πηγές για το συγκεκριμένο θέμα.

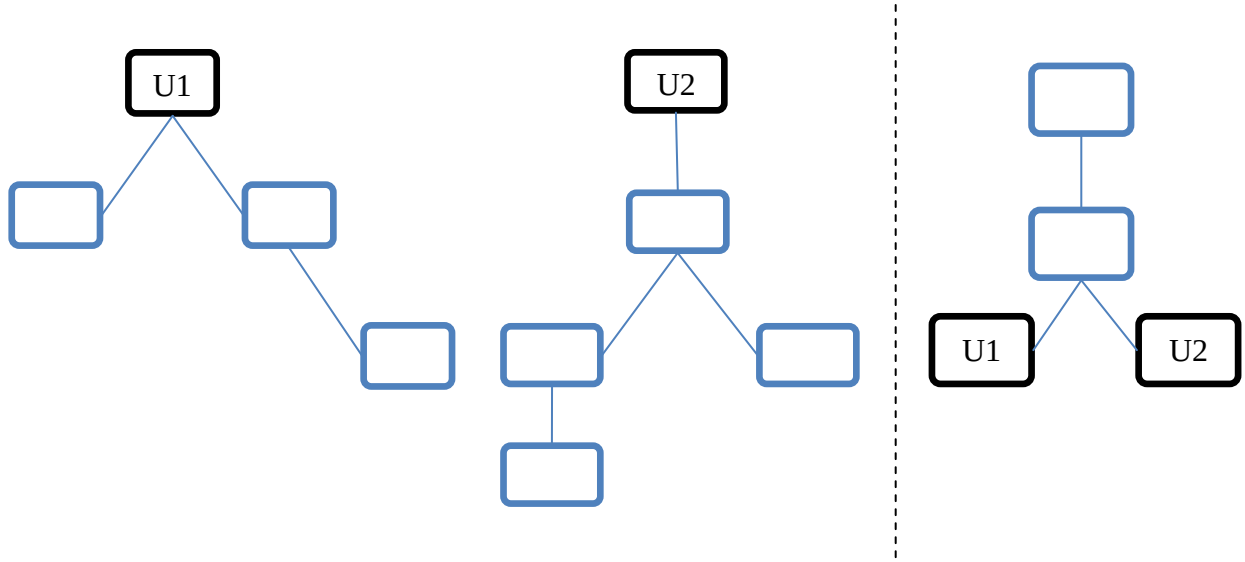
2. *Οι χρήστες που δημοσίευσαν τα R1 και R2 συνυπάρχουν σε κάποιο άλλο δέντρο re-tweet και επιπλέον βρίσκονται στο ίδιο μονοπάτι από την ρίζα.*
   Σε αυτή την περίπτωση υπολογίζουμε την απόστασή τους και εφαρμόζουμε κατώφλι.



Εικόνα 13: Οι χρήστες που δημοσίευσαν τα R1 και R2 συνυπάρχουν σε κάποιο άλλο δέντρο re-tweet επιπλέον βρίσκονται στο ίδιο μονοπάτι από την ρίζα.

Η ύπαρξη αυτής της εξάρτησης δείχνει πως οι χρήστες που δημοσίευσαν τα tweets R1 και R2 έχουν στο παρελθόν κάνει re-tweet ο ένας τον άλλο είτε άμεσα (απόσταση = 1) είτε έμμεσα (απόσταση > 1). Η τιμή του κατωφλίου μπορεί να αλλάξει μέχρι να πετύχουμε το καλύτερο αποτέλεσμα.

3. *Οι χρήστες που δημοσίευσαν τα R1 και R2 συνυπάρχουν σε οποιοδήποτε άλλο δέντρο re-tweet.*



Εικόνα 14: Οι χρήστες που δημοσίευσαν τα R1 και R2 συνυπάρχουν σε κάποιο άλλο δέντρο re-tweet.

Η ύπαρξη αυτής της εξάρτησης δείχνει πως οι χρήστες που δημοσίευσαν τα tweets R1 και R2 ακολουθούν και κάνουν re-tweet κάποιον κοινό τρίτο χρήστη, γεγονός που μειώνει την ανεξαρτησία των δύο χρηστών.

5. Υλοποίηση και Αποτίμηση

5.1 Χρησιμοποιούμενες τεχνολογίες και υπηρεσίες

Στα πλαίσια της υλοποίησης της προτεινόμενης μεθοδολογίας που περιγράφηκε στην προηγούμενη ενότητα, χρησιμοποιήσαμε υπάρχουσες τεχνολογίες που ταίριαζαν στις ανάγκες του λογισμικού που κατασκευάζαμε. Επιπλέον, χρησιμοποιήσαμε υπηρεσίες με σκοπό να έχουμε πρόσβαση σε πληροφορίες και διαδικασίες που χρειαζόμασταν. Σε αυτή την ενότητα περιγράφονται τόσο οι ανάγκες σε τεχνολογίες και πώς αυτές καλύφθηκαν από τις τεχνολογίες που χρησιμοποιήσαμε, όσο και μια επισκόπηση των δυνατοτήτων των υπηρεσιών που χρησιμοποιήσαμε.

5.1.1 Τεχνολογίες

5.1.1.1 Java

Η Java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού γενικής χρήσεως. Άρχισε να αναπτύσσεται από την Sun Microsystems το 1991 και έγινε διαθέσιμη για το ευρύ κοινό το 1995. Από τότε ακολούθησε μια εξαιρετικά επιτυχημένη πορεία με αποτέλεσμα σήμερα να θεωρείται μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού και να μπορεί να τρέξει σχεδόν σε όλα τα σύγχρονα λειτουργικά συστήματα και στις περισσότερες αρχιτεκτονικές υπολογιστών.

Εικόνα 15: Το λογότυπο της Java.

Παρακάτω παραθέτουμε μια σύντομη αναφορά στα σημαντικότερα για μας χαρακτηριστικά της γλώσσας και πώς αυτά μας φάνηκαν χρήσιμα κατά την υλοποίηση του λογισμικού μας. (25)

Εικονική μηχανή

Κύριο σύνθημα για την προώθηση της γλώσσας από τους δημιουργούς της ήταν το "Write once, run anywhere", δηλαδή ‘Προγραμμάτισε μια φορά, εκτέλεσε παντού’. Αυτό οφείλεται στην ύπαρξη της Εικονικής Μηχανής (Virtual Machine) της Java. Όταν ένα πρόγραμμα Java μεταγλωττιστεί, δεν παράγεται γλώσσα μηχανής όπως σε μια συνηθισμένη γλώσσα, αλλά κώδικας Bytecode που πρόκειται για ένα ενδιάμεσο κώδικα που μπορεί να διαβαστεί από την Εικονική Μηχανή. Η Εικονική Μηχανή είναι υπεύθυνη έπειτα να μετατρέψει τον κώδικα Bytecode σε γλώσσα μηχανής για να εκτελεστεί το λογισμικό. Η διαθεσιμότητα της στην πλειοψηφία των λειτουργικών συστημάτων και αρχιτεκτονικών υπολογιστών, δίνει την δυνατότητα σε ένα πρόγραμμα να γραφτεί σε μια πλατφόρμα και να εκτελεστεί σε μια άλλη.

Παρόλο που στην πραγματικότητα τα πράγματα δεν είναι τόσο ιδανικά όσο τα περίγραφαν οι δημιουργοί της γλώσσας, αφού τις περισσότερες φορές χρειάζονται μικρομετατροπές για να μεταφέρεις ένα λογισμικό από την μια πλατφόρμα σε μια άλλη, το μεγαλύτερο μέρος του κώδικα μπορεί να τρέξει χωρίς αλλαγές. Στην περίπτωσή μας, η πλατφόρμα ανάπτυξης του λογισμικού ήταν διαφορετική από την πλατφόρμα εκτέλεσης γι’αυτό η Java ήταν η ιδανική επιλογή. Μπορούσαμε να μεταγλωττίσουμε το λογισμικό μας στην πλατφόρμα ανάπτυξης και έπειτα να μεταφέρουμε αυτούσιο το εκτελέσιμο στην πλατφόρμα εκτέλεσης για να το τρέξουμε χωρίς αλλαγές.

Αντικειμενοστραφής σχεδίαση

Σαν αντικειμενοστραφή προγραμματισμό ονομάζουμε την μεθοδολογία ανάπτυξης λογισμικού σύμφωνα με την οποία δεδομένα και διαδικασίες που επενεργούν σε αυτά ενοποιούνται σε μια ενιαία οντότητα που ονομάζετε αντικείμενο. Η Java σχεδιάστηκε από την αρχή σαν μια γλώσσα που λάμβανε τις αρχές του αντικειμενοστραφής προγραμματισμού πολύ σοβαρά υπόψη. Χαρακτηριστικά, η αντικειμενοστρέφια είναι τόσο βαθιά χαραγμένη στην Java που δεν επιτρέπει την εκτέλεση εντολών έξω από ορισμένες κλάσεις.

Χρησιμοποιώντας την δυνατότητα να δημιουργούμε και να αλληλεπιδρούμε με αντικείμενα μπορέσαμε να γράψουμε κώδικα που ήταν επαναχρησιμοποιήσιμος, δηλαδή αποφύγαμε οποιαδήποτε αντιγραφή του κώδικα. Επιπλέον, λόγω της αντικειμενοστραφής σχεδίασης της γλώσσας μπορέσαμε να εφαρμόσουμε με ευκολία στον κώδικά μας σχεδιαστικά μοτίβα για να μειώσουμε την πολυπλοκότητά του και να τον κάνουμε πιο αποδοτικό. Τα σχεδιαστικά μοτίβα ορίζονται ως μία αποδεδειγμένα καλή λύση που έχει εφαρμοστεί με επιτυχία στην επίλυση ενός επαναλαμβανόμενου προβλήματος σχεδίασης συστημάτων λογισμικού και με τη χρήση τους μας επιτρέπουν να αντικαταστήσουμε πλήρως μεγάλα τμήματα του κώδικα του με ‘μαύρα κουτιά’.

Συλλέκτης απορριμμάτων

Παρόλο που το συντακτικό της γλώσσας βασίζεται έντονα στην γλώσσα C και C++, η Java δεν επιτρέπει στον προγραμματιστή την διαχείριση της μνήμης. Αυτό οφείλεται στον Συλλέκτη Απορριμμάτων (Garbage Collector) ο οποίος καλείται από την Εικονική Μηχανή όταν η μνήμη του προγράμματος αρχίζει να γεμίζει με σκοπό να διαγράψει από την μνήμη τα μη χρησιμοποιούμενα πλέον αντικείμενα.

Το χαρακτηριστικό αυτό είναι πολύ μεγάλης σημασίας αφού πλέον ο προγραμματιστής μπορεί να εστιάσει στην ανάπτυξη του λογισμικού και όχι στο πότε και αν θα ελευθερώσει ένα συγκεκριμένο τμήμα της μνήμης. Επιπλέον, αποφεύγονται σχεδόν όλα τα σφάλματα δεικτών τα όποια ήταν ένα από τα συνηθέστερα σφάλματα των προγραμματιστών, τα οποία οφείλονταν στον λανθασμένο χειρισμό την μνήμης.

Υψηλή απόδοση

Η Java αρχικά ήταν πιο αργή σε σχέση με άλλες προγραμματιστικές γλώσσες όπως η C και η C++. Αυτό οφειλόταν στην Εικονική Μηχανής της, που παρόλα τα πλεονεκτήματα που πρόσφερε, ήταν ένα ενδιάμεσο στάδιο μεταξύ του κώδικα Bytecode και της γλώσσας μηχανής που στοίχιζε ως αναφορά την απόδοση των εκτελέσιμών της. Ευτυχώς, στις τελευταίες εκδόσεις των Εικονικών Μηχανών καθιερώθηκαν οι μεταγλωττιστές JIT (Just In Time), οι οποίοι μετατρέπουν τον κώδικα Bytecode απευθείας σε γλώσσα μηχανής με αποτέλεσμα η διαφορά ταχύτητας από άλλες γλώσσες προγραμματισμού να είναι πολύ μικρή.

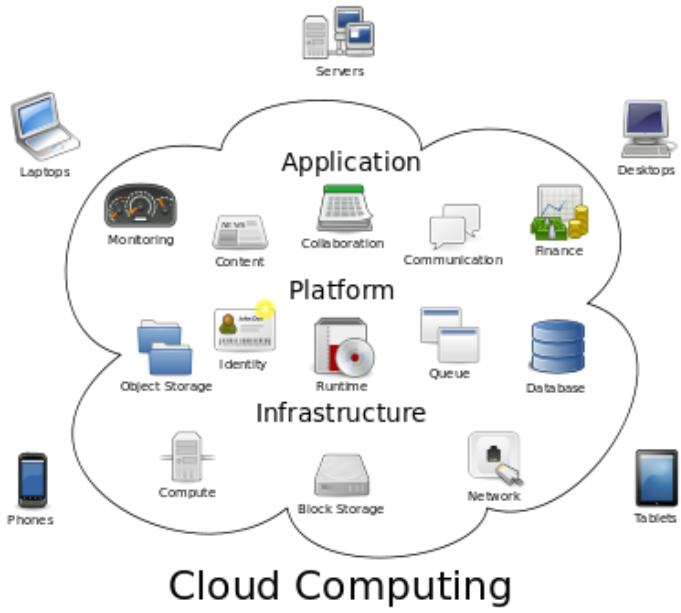
Πλούσιες βιβλιοθήκες και API

Ίσως το πιο χρήσιμο για εμάς χαρακτηριστικό της γλώσσας, είναι οι πλούσιες βιβλιοθήκες που προσφέρει καθώς και η αφθονία πόρων που είναι διαθέσιμες στο διαδίκτυο. Με τις ενσωματωμένες βιβλιοθήκες που παρέχει η γλώσσα, εκτός από το ότι μας δίνει την δυνατότητα να εκτελέσουμε οποιαδήποτε λειτουργία, μας προσφέρει έτοιμες υλοποιήσεις δημοφιλών τύπων δεδομένων που επιταχύνουν αισθητά τον χρόνο ανάπτυξης του λογισμικού.

Επίσης, λόγω του ότι η γλώσσα είναι μια από τις πλέον διαδεδομένες, οι περισσότερες Διεπαφές Προγραμματισμού Εφαρμογών (Application programming interfaces) είναι διαθέσιμες πρώτα για γλώσσα Java. Αυτό μας βοήθησε ιδιαίτερα στην χρήση των υπηρεσιών που θα αναφερθούμε στην συνέχεια, αφού σχεδόν όλες οι υπηρεσίες που χρησιμοποιήσαμε είχαν API συμβατό με Java.

5.1.1.2 Cloud Computing

Ο όρος Cloud Computing αναφέρεται στην χρήση υπολογιστικών πόρων σαν υπηρεσίες διαμέσου ενός δικτύου, συνήθως του διαδικτύου. Το όνομα Cloud (σύννεφο) αναφέρεται για να δείξει το αφαιρετικό επίπεδο που διέπει την συγκεκριμένη τεχνολογία, αφού ο χρήστης που εκμεταλλεύεται την υπηρεσία δεν ασχολείται με την πολυπλοκότητα που απαιτείτε για να εγκαταστήσει και να συντηρήσει το σύστημα.



Εικόνα 16: Λογικό διάγραμμα για το Cloud Computing.

Παραπάνω φαίνεται μια γραφική απεικόνιση της δομής που μπορεί να έχει ένα σύστημα Cloud Computing. Πιο συγκεκριμένα, έξω από το σύννεφο, που πρόκειται για το σύνορο του συστήματος, εμφανίζονται οι διάφοροι πιθανοί πελάτες όπως φορητοί Η.Υ. ή έξυπνα τηλέφωνα.

Στο εσωτερικό του σύννεφου μπορούμε να διακρίνουμε τον διαχωρισμό του Cloud Computing ως προς το είδος των προσφερόμενων υπηρεσιών:

- *SaaS (Software as a Service)*: Πρόκειται για την υπηρεσία παροχής λογισμικού από ένα παροχέα υπηρεσιών αντί της αγοράς του ίδιου του λογισμικού. Η εφαρμογή καθεαυτή και τα δεδομένα της βρίσκονται στο σύννεφο και έτσι ο χρήστης της μπορεί να έχει πρόσβαση στις εφαρμογές και δεδομένα που επιθυμεί από παντού.

- *PaaS (Platform as a Service)*: Πρόκειται για την υπηρεσία όπου οι βιβλιοθήκες και τα εργαλεία ανάπτυξης εφαρμογών βρίσκονται στο σύννεφο. Ο χρήστης δεν χρειάζεται να ασχοληθεί με τη συντήρηση του λειτουργικού και της και της πλατφόρμας ανάπτυξης αφού προσφέρονται και συντηρούνται από το σύννεφο.

- *IaaS (Infrastructure as a service)*: Πρόκειται για την υπηρεσία παροχής υποδομής (υλικού) ανάλογα με τις απαιτήσεις του χρήστης την παρούσα χρονική στιγμή.

Τα πλεονεκτήματα της τεχνολογίας του Cloud Computing είναι πολλά και παρουσιάζονται συνοπτικά παρακάτω:

- Πρόσβαση από οπουδήποτε, οποτεδήποτε αφού υποδομή, εφαρμογές και δεδομένα δεν περιορίζονται σε εσωτερικά δίκτυα.

- Το κόστος και ο κόπος αγοράς υλικού και αναβαθμίσεων μειώνεται αφού αυτά αναλαμβάνονται από την υποδομή του νέφους.

- Μειώνετε το ρίσκο της αγοράς επιπλέον εξοπλισμού και υποχρησιμοποίησης του αφού ο χρήστης πληρώνει μόνο όσους πόρους (υπολογιστική ισχύς, μνήμη, κτλ) χρησιμοποιεί.

- Εύκολη κλιμάκωση αφού όταν οι απαιτήσεις σε πόρους αυξάνονται τότε απλά δεσμεύονται αυτόματα περισσότεροι πόροι στο σύννεφο και το ανάποδο.

Εμείς χρησιμοποιήσαμε την τεχνολογία Cloud Computing σαν πλατφόρμα εκτέλεσης του λογισμικού μας για να εκμεταλλευτούμε τα πλεονεκτήματα που αναφέρθηκαν. Πιο συγκεκριμένα, χρησιμοποιήσαμε τις υπηρεσίες Cloud Computing (IaaS) που προσφέρει δωρεάν για ακαδημαϊκούς σκοπούς το Okeanos από το GRNET. (26)

5.1.2 Υπηρεσίες

5.1.2.1 Big Huge Thesaurus

Κατά την υλοποίηση του λογισμικού, προέκυψε η ανάγκη να βρούμε συνώνυμες λέξεις για κάποιες λέξεις κλειδιά που δεχόμασταν ως είσοδο με στόχο να γενικεύσουμε την αναζήτηση.

Για να το πετύχουμε αυτό, χρησιμοποιήσαμε το API που προσφέρει το Big Huge Thesaurus από τα Big Hude Labs (27). Η συγκεκριμένη υπηρεσία προσφέρει τη δυνατότητα να ρωτάμε για μια συγκεκριμένη λέξη και να μας επιστρέφει πίσω συνώνυμα, αντώνυμα, σχετικές λέξεις και παρόμοιες λέξεις καταχωρημένες από τους χρήστες του. Στην περίπτωσή μας ενδιαφερόμασταν μόνο για τις συνώνυμες λέξεις.

Ως πηγή η συγκεκριμένη υπηρεσία χρησιμοποιεί το WordNet database του Princeton University (28). Το WordNet είναι μια λεξιλογική βάση δεδομένων στην οποία περιέχονται ομαδοποιήσεις αγγλικών λέξεων σε σύνολα συνωνύμων, μικρός ορισμός των λέξεων και καταγραφές των διαφόρων σημασιολογικών σχέσεων μεταξύ αυτών των συνόλων συνωνύμων. Στην τελευταία έκδοση της (έκδοση 3.0, Δεκέμβρης 2006) η βάση δεδομένων περιείχε 155,287 λέξεις χωρισμένες σε 117,659 σύνολα συνωνύμων.

Χρήση της υπηρεσίας

Η δωρεάν πρόσβαση στο API μας επιτρέπει την ερώτηση για 10,000 λέξεις ημερησίως. Η υπηρεσία δεν προσφέρει κάποια έτοιμη βιβλιοθήκη για χρήση από κάποια γλώσσα προγραμματισμού. Προσφέρει όμως ένα RESTful API για να ρωτάμε και να παίρνουμε απαντήσεις με τα σχετικά συνώνυμα, αντίθετα, κτλ. Γι' αυτόν το λόγο για να χρησιμοποιήσουμε την υπηρεσία δημιουργήσαμε μια συνάρτηση, ως μέρος του σχετικού αντικειμένου που είναι υπεύθυνο για την επεξεργασία των keywords, η οποία δέχεται ως είσοδο μια λέξη και επιστρέφει μια συμβολοσειρά με τις συνώνυμες λέξεις χωρισμένες με κόμματα.

Για παράδειγμα η εκτέλεση της παρακάτω εντολής:

```
String syn = getSynonymsList ( "happy" );
```

θα έχει σαν αποτέλεσμα η μεταβλητή syn να περιέχει:

```
felicitous,glad,well-chosen
```

5.1.2.2 OpenCalais

Στην προσπάθεια να βρούμε ποια μηνύματα είναι σημασιολογικά παρόμοια, χρειαστήκαμε μια υπηρεσία που θα μας βοηθούσε να αναλύσουμε το κείμενο των μηνυμάτων.

Για να το πετύχουμε αυτό, χρησιμοποιήσαμε το API που προσφέρει το OpenCalais από την Thomson Reuters (29). Πρόκειται για μια υπηρεσία η οποία προσφέρει την δυνατότητα να της δώσουμε ένα κείμενο και να εξάγει σημασιολογικές πληροφορίες από αυτό, χρησιμοποιώντας τεχνολογίες Επεξεργασίας Φυσικής Γλώσσας (Natural Language Processing - NLP).

Εικόνα 17: Το λογότυπο του OpenCalais.

Πιο συγκεκριμένα, οι δυνατότητες της υπηρεσίας που μας ενδιαφέρουν είναι η ικανότητά του να εξάγει 38 είδη οντοτήτων (Named Entities) όπως:

- Επετείους
- Πόλεις
- Εταιρείες
- Ηπείρους
- Χώρες
- Νομίσματα
- Διευθύνσεις Email
- Λειτουργικά Συστήματα
- Αριθμούς
- Τηλεφώνου
- Διευθύνσεις URL
- Ραδιοσταθμούς

και 82 είδη εκδηλώσεων και γεγονότων (Events and Facts) όπως:

- Αποκτήσεις
- Συγχωνεύσεις
- Πτωχεύσεις
- Συλλήψεις
- Δίκες
- Απολύσεις

Χρήση της υπηρεσίας

Η υπηρεσία προσφέρεται δωρεάν και δεν υπάρχει ημερήσιο όριο για τη χρήση της. Προσφέρουν επίσημα API για SOAP και για REST, αλλά επίσης υπάρχουν από τρίτους έτοιμες βιβλιοθήκες για διάφορες γλώσσες όπως η Java. Χρησιμοποιώντας την βιβλιοθήκη για Java δημιουργήσαμε μια συνάρτηση, η οποία δέχεται για είσοδο το κείμενο ενός μηνύματος tweet και την οντότητα που μας ενδιαφέρει και επιστρέφει ως έξοδο ένα πίνακα με τις τιμές της συγκεκριμένης οντότητας, αν υπάρχουν.

Για παράδειγμα η εκτέλεση της παρακάτω εντολής:

```
Vector<String> res = getCalaisEntity ( “I want to travel to Greece and Cyprus!” , “country” );
```

θα έχει σαν αποτέλεσμα η μεταβλητή res να περιέχει:

```
{ “Greece”, “Cyprus” }
```

5.1.2.3 GeoNames

Κατά την προσπάθεια εντοπισμού σημασιολογικής ομοιότητας των μηνυμάτων, εμφανίστηκε η ανάγκη να μπορούμε να αναγνωρίσουμε σε ποια χώρα ανήκει μια πόλη ή περιοχή, ώστε να ξέρουμε ότι κάποια μηνύματα αναφέρονται στον ίδιο γεωγραφικό χώρο.

Για να το πετύχουμε αυτό, χρησιμοποιήσαμε το API που προσφέρει το GeoNames (30). Το GeoNames είναι μια γεωγραφική βάση δεδομένων, διαθέσιμη και προσβάσιμη μέσα από διάφορες διαδικτυακές υπηρεσίες, σύμφωνα με μια ελεύθερη άδεια Creative Commons Attribution. Η βάση περιλαμβάνει περισσότερα από 10,000,000 γεωγραφικές ονομασίες και εκτός από ονομασίες πόλεων, χωρών και τις σχέσεις μεταξύ τους, που μας ενδιαφέρει άμεσα, περιέχει πληροφορίες όπως γεωγραφικά μήκη και πλάτη για το κάθε μέρος, υψόμετρο, πληθυσμό, ταχυδρομικό κώδικα, κτλ.

Εικόνα 18: Το λογότυπο του GeoNames.

Χρήση της υπηρεσίας

Η δωρεάν πρόσβαση στο API μας επιτρέπει την ερώτηση για 30,000 ονομασίες πόλεων ημερησίως. Η υπηρεσία προσφέρει επίσημα βιβλιοθήκες για να χρησιμοποιηθεί από διάφορες γλώσσες προγραμματισμού, συμπεριλαμβανομένης και της Java. Χρησιμοποιώντας την βιβλιοθήκη για Java δημιουργήσαμε μια συνάρτηση, η οποία δέχεται σαν είσοδο το τοπωνύμιο μιας πόλης ή περιοχής και επιστρέφει σαν έξοδο την χώρα στη οποία ανήκει το συγκεκριμένο τοπωνύμιο. Να σημειώσουμε πως σε περίπτωση που υπάρχει συνωνυμία σε ένα τοπωνύμιο τότε επιλέγεται το πιο δημοφιλές για λόγους απλότητας.

Για παράδειγμα η εκτέλεση της παρακάτω εντολής:

```
String country = getCountry ( “Athens” );
```

θα έχει σαν αποτέλεσμα η μεταβλητή country να περιέχει:

```
”Greece”
```

5.2 Σύνολο Δεδομένων

5.2.1 Χαρακτηριστικά Συνόλου Δεδομένων

Το σύνολο δεδομένων που χρησιμοποιήσαμε περιλαμβάνει 467 εκατομμύρια tweets από 20 εκατομμύρια χρήστες. Η χρονική περίοδος που έγιναν τα tweets ξεκινάει την 1η Ιουνίου 2009 και τελειώνει την 31η Δεκεμβρίου 2009. Υπολογίζεται πως τα tweets στο σύνολο δεδομένων αποτελούν το 20-30% όλων των δημοσίων tweets που έγιναν την συγκεκριμένη χρονική περίοδο. Το πλήθος των tweets είναι χωρισμένο σε 6 αρχεία, ένα για κάθε μήνα της χρονικής περιόδου, με συνολικό μέγεθος 72.4 GB. Το dataset ήταν μέρος του SNAP (Stanford Network Analysis Project) (31).

Εικόνα 19: Το λογότυπο του Stanford Network Analysis Project.

Για κάθε δημόσιο tweet στο dataset είναι διαθέσιμες οι ακόλουθες πληροφορίες:

- Όνομα χρήστη
- Ημερομηνία και ώρα δημοσίευσης
- Περιεχόμενο

Παρακάτω παρατίθενται μερικά στατιστικά του dataset που χρησιμοποιήσαμε:

Πλήθος χρηστών	17,069,982
Πλήθος tweets	476,553,560
Πλήθος URLs	181,611,080
Πλήθος Hashtags	49,293,684
Πλήθος re-tweets	71,835,017

Πίνακας 2: Στατιστικά του συνόλου δεδομένων που χρησιμοποιήσαμε.

Twitter API

Ο λόγος που δεν χρησιμοποιήσαμε απευθείας το Twitter API, και πιο συγκεκριμένα το *Twitter Search API* (32), ήταν λόγω του περιορισμού που θέτει όσον αφορά στο πλήθος των tweets που είναι δυνατό να ανακτηθούν για μια συγκεκριμένη λέξη-κλειδί. Ως εναλλακτική, μπορούσαμε να χρησιμοποιήσουμε το *Twitter Streaming API* (33), το οποίο θα μας επέτρεπε την ανάκτηση των tweets καθώς αυτά δημοσιεύονται.

Και στις δυο περιπτώσεις που αναφέρθηκαν παραπάνω, η ανάκτηση του συνόλου δεδομένων θα αποτελούσε εργασία αρκετής πολυπλοκότητας, αφού θα έπρεπε να αντιμετωπίσουμε θέματα όπως την ανίχνευση των διπλών μηνυμάτων tweets που θα ανακτούσαμε αλλά και υποβολή μεγάλου αριθμού συναφών λέξεων-κλειδιών αποσκοπώντας σε έναν επαρκή όγκο δεδομένων προς ανάλυση. Γι'αυτόν τον λόγο επιλέξαμε να βασιστούμε σε ένα έτοιμο σύνολο δεδομένων για την υλοποίηση και αποτίμηση της μεθοδολογίας μας.

Παρατηρήσεις σχετικά με το σύνολο δεδομένων

Χρησιμοποιώντας το συγκεκριμένο σύνολο δεδομένων, παρατηρήσαμε μερικά χαρακτηριστικά στα οποία θεωρούμε απαραίτητο να αναφερθούμε.

- Τα μηνύματα του συνόλου δεδομένων χρονολογούνται όλα στο 2009. Εκείνη την περίοδο δεν είχε υποστηριχτεί το Re-Tweet ως επίσημο API από το Twitter (34). Αυτό είχε ως αποτέλεσμα η κάθε εφαρμογή πελάτης για το δίκτυο του Twitter και ο κάθε χρήστης να μπορεί να χρησιμοποιεί το δικό του re-tweet format. Ενδεικτικά αναφέρουμε τα παρακάτω μηνύματα, τα οποία κάνουν re-tweet το ίδιο μήνυμα αλλά έχουν εντελώς διαφορετική μορφή:

    - RT **@user**: This is an original tweet! **URL** | This is a comment
    - R/T **@user** This is an original tweet! **URL** << This is a comment
    - This is a comment **URL** (via **@user**)

    Έτσι, ο εντοπισμός των μηνυμάτων που είχαν γίνει re-tweeted κατέστη δυσχερής αφού ο κάθε χρήστης μπορούσε να κάνει re-tweet χρησιμοποιώντας τον δικό του τρόπο. Εμείς υποστηρίξαμε την πλειοψηφία των re-tweet formats και δώσαμε έμφαση στους πιο δημοφιλείς τρόπους re-tweet.

- Για να διευκολυνθούμε όσον αφορά στην επεξεργασία του συνόλου δεδομένων, προτιμήσαμε να διαβάζουμε τα tweets με χρονολογική σειρά. Δυστυχώς, τα tweets στο σύνολο δεδομένων δεν ήταν καταχωρημένα με χρονολογική σειρά και η ταξινόμηση με μια ενσωματωμένη συνάρτηση της Java δεν ήταν εφικτή λόγω του μεγάλου μεγέθους των αρχείων του συνόλου (το μεγαλύτερο αρχείο είχε μέγεθος 20 GB).

    Γι'αυτό βασιζόμενοι σε ένα σχετικό project (35), δημιουργήσαμε ένα βοηθητικό εργαλείο που χρησιμοποιούσε μια μέθοδο External Sorting για να ταξινομήσουμε τα tweets που βρίσκονταν σε κάθε αρχείο του συνόλου δεδομένων. Το External Sorting είναι μια κατηγορία αλγορίθμων ταξινόμησης η οποία χρησιμοποιείται όταν τα δεδομένα προς ταξινόμηση δεν χωρούν στην κύρια μνήμη και γι'αυτό βρίσκονται στην πιο αργή εξωτερική μνήμη, όπως ένα σκληρό δίσκο. Συνήθως χρησιμοποιείται ένα είδος merge-sort, αφού δημιουργούνται κομμάτια δεδομένων αρκετά μικρά ώστε να χωράνε στην κύρια μνήμη τα οποία ταξινομούνται και αντιγράφονται σε ένα προσωρινό αρχείο. Στο στάδιο της συγχώνευσης, τα ταξινομημένα αρχεία συνδυάζονται σε ένα ενιαίο μεγαλύτερο αρχείο.

5.2.2 Η διεπαφή Social Sensing Portal

Στόχος μας ήταν η δημιουργία ενός βοηθητικού εργαλείου προσβάσιμου από το διαδίκτυο που θα μας επέτρεπε να βρούμε ποια tweets έγιναν re-tweet ή είχαν πολλές απαντήσεις (replies) ανάλογα με τις παραμέτρους που θέτουμε.

Σκοπός μας κατά την δημιουργία του συγκεκριμένου εργαλείου ήταν να εξοικειωθούμε με το σύνολο δεδομένων, να προσπαθήσουμε να εντοπίσουμε χειροκίνητα αν υπήρχαν φήμες στο σύνολο δεδομένων μας, αλλά και να δώσουμε μια διεπαφή με τους υποκείμενους μηχανισμούς.

Περιγραφή παραμέτρων

Το Social Sensing Portal είναι ένα εργαλείο που μας επιτρέπει την παραμετροποίηση των αποτελεσμάτων του ανάλογα με τις παρακάτω παραμέτρους που μπορούμε να αλλάξουμε δυναμικά:

- **Social Sensing Mode:** Επιλογή μεταξύ των δύο τρόπων λειτουργίας του εργαλείου:
    1. της ανίχνευσης των re-tweets
    2. της ανίχνευσης των απαντήσεων (replies) των tweets

- **Keywords:** Λίστα με τα keywords που πρέπει να περιέχουν τα re-tweets ή τα αρχικά tweets για να ληφθούν υπόψιν. Παρακάτω παρατίθενται μερικά παραδείγματα που δείχνουν τις δυνατότητες παραμετροποίησης του πεδίου:
    - *key1,key2* – Απαραίτητη η εμφάνιση του key1 ή του key2 στο tweet.
    - *key1,key2|key3,key4* – Απαραίτητη η εμφάνιση του key1 ή του key2 ΚΑΙ η εμφάνιση του key3 ή του key4 στο tweet
    - *!!custom_regex* - Απαραίτητη η ικανοποίηση της regular expression στο tweet

- **Spreading Hours** και **Spreading Number of Tweets**: Για να παρουσιαστεί στα αποτελέσματα, ένα tweet θα πρέπει να έχει γίνει re-tweet περισσότερες φορές από όσες ορίζει η παράμετρος "Spreading Number of Tweets" μέσα σε όσες ώρες ορίζει η παράμετρος "Spreading Hours". Κατα αναλογία, ένα tweet θα πρέπει να έχει περισσότερες απαντήσεις (replies) από όσες ορίζει η παράμετρος "Spreading Number of Tweets" μέσα σε όσες ώρες ορίζει η παράμετρος "Spreading Hours".

- **Filter mode:** Εφαρμόσιμο μόνο στον τρόπο λειτουργίας για την ανίχνευση re-tweets. Επιλογή της σειράς με την οποία θα γίνει το φιλτράρισμα των tweets:
    1. Πρώτα έλεγχος για την ικανοποίηση της συνθήκης του keyword και έπειτα έλεγχος για το αν το tweet ανήκει σε κάποιο re-tweet tree.
    2. Δημιουργία πρώτα των re-tweet trees και έπειτα έλεγχος για την ικανοποίηση της συνθήκης του keyword.

  Σημειώνεται πως η παράμετρος αυτή δεν επηρεάζει τα αποτελέσματα, αλλά την απόδοση του εργαλείου.

- **Extra Options:** Δυνατότητα για επιπλέον παραμέτρους που θα προστεθούν στο μέλλον στο εργαλείο. Την παρούσα στιγμή, το εργαλείο παρέχει την εξής επιπλέον δυνατότητα:

    - **syn:** Αυτόματη εύρεση συνώνυμων για τις λέξεις που έχουν καθοριστεί στο πεδίο keywords από το Princeton University WordNet database και το Carnegie Mellon Pronouncing Dictionary.

      Για να το πετύχουμε αυτό χρησιμοποιήσαμε το API που προσφέρεται από το Big Huge Thesaurus για την εύρεση συνωνύμων. Πιο συγκεκριμένα, το API μας επιτρέπει να ρωτούμε μέχρι και για 10,000 λέξεις ημερησίως και να λαμβάνουμε μια λίστα με όλα τα συνώνυμα από τα προαναφερθέντα λεξικά όπως περιγράφηκαν με λεπτομέρεια στην ενότητα 5.1.2.1.

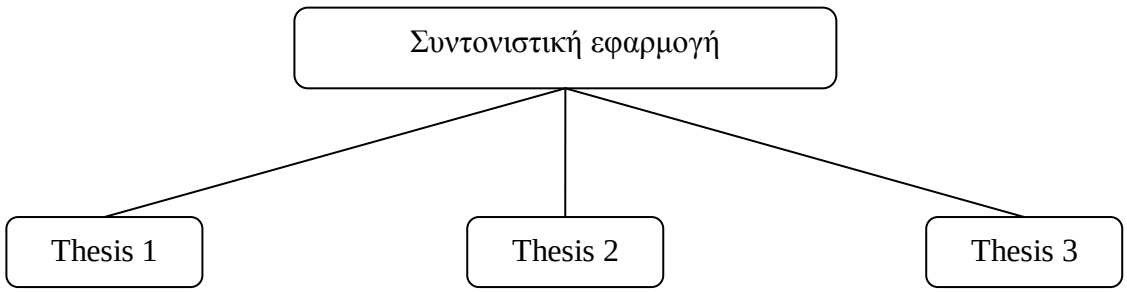
- **Email:** Όταν το εργαλείο έχει έτοιμα τα αποτελέσματα της σάρωσης του dataset, υπάρχει η δυνατότητα να σταλούν με email στην διεύθυνση που ορίζεται στο συγκεκριμένο πεδίο.

Αρχιτεκτονική συστήματος

Το σύστημα αποτελείται από 3 απομακρυσμένους εικονικούς υπολογιστές σε περιβάλλον Cloud που είναι υπεύθυνοι για την επεξεργασία του συνόλου δεδομένων. Το dataset έχει κατανεμηθεί ισοδύναμα στους 3 εικονικούς υπολογιστές. Επιπλέον, υπάρχει μια κεντρική web-εφαρμογή με σκοπό να στέλνει τα αιτήματα στους 3 εικονικούς υπολογιστές όπως επίσης και να παραλαμβάνει τα αποτελέσματα για να τα στείλει στο email του χρήστη που ζήτησε τα αποτελέσματα.

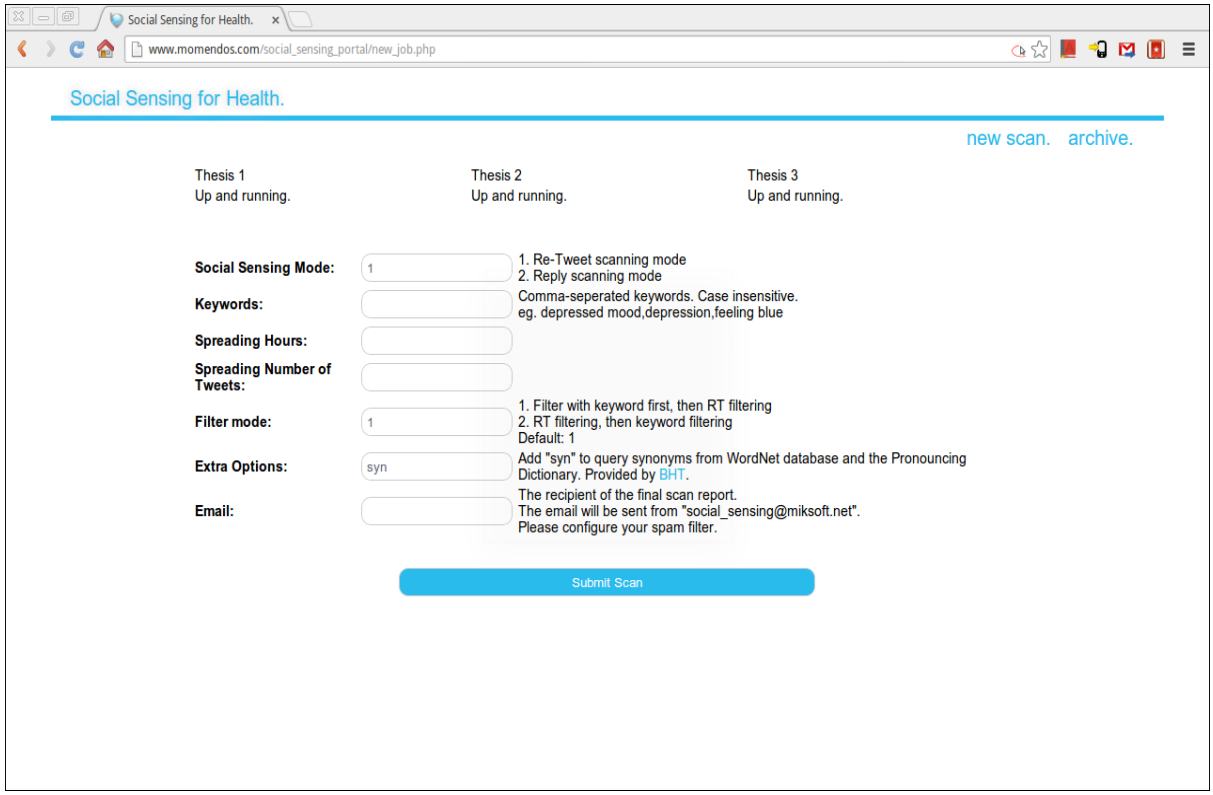
Το λογισμικό επεξεργασίας του συνόλου δεδομένων καθώς και το λογισμικό εξυπηρετητή το οποίο είναι υπεύθυνο για την επικοινωνία με την κεντρική web-εφαρμογή έχουν υλοποιηθεί σε γλώσσα Java. Η κεντρική web-εφαρμογή υπεύθυνη για τον συντονισμό του εργαλείου έχει γραφτεί με PHP και σε κάποια κομμάτια χρησιμοποιήθηκε και Javascript.

Η διάταξη των υπολογιστών που χρησιμοποιεί το εργαλείο φαίνεται παρακάτω:



Εικόνα 20: Η διάταξη των υπολογιστών που χρησιμοποιεί το βοηθητικό εργαλείο.

Παρουσίαση εργαλείου



Εικόνα 21: Η αρχική οθόνη (New Scan) του βοηθητικού εργαλείου.

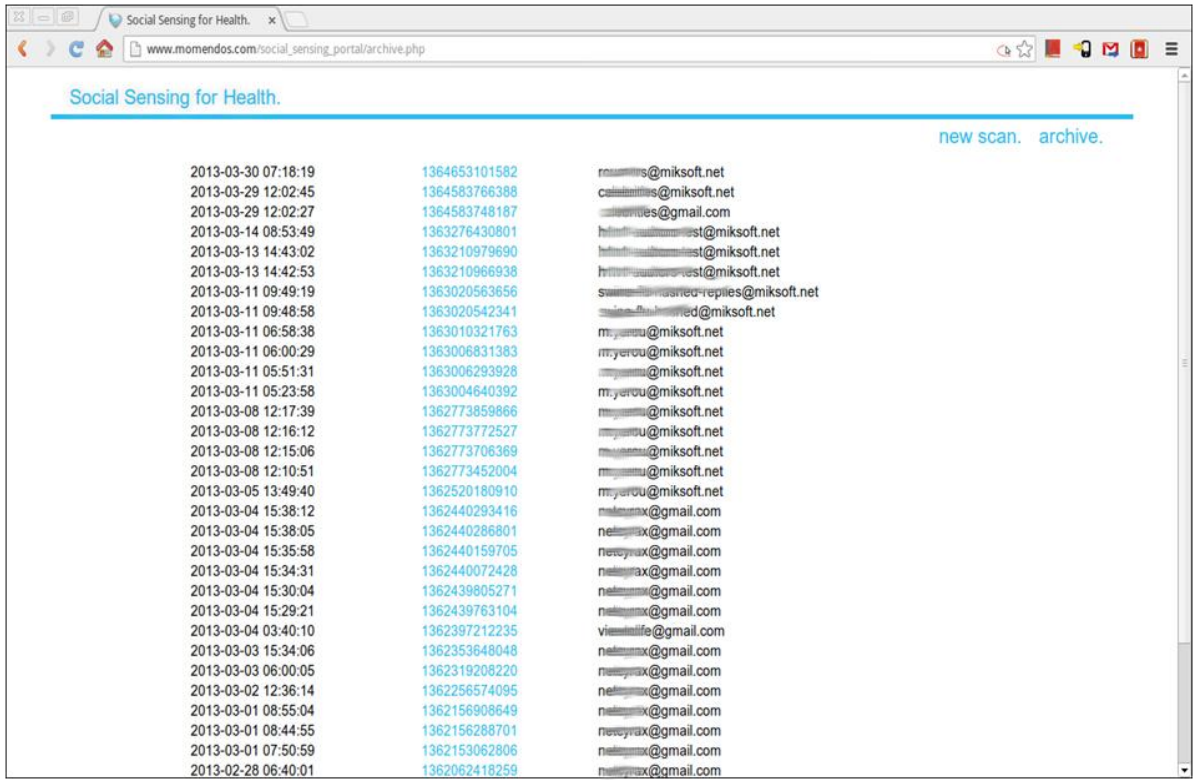
Η αρχική οθόνη (*New Scan*) που βλέπει ο χρήστης μόλις έχει πρόσβαση στο εργαλείο.

Πάνω δεξιά, διακρίνεται το μενού του εργαλείου που επιτρέπει:

1. την υποβολή νέας ‘σάρωσης’ του συνόλου δεδομένων
2. την ανασκόπηση αποτελεσμάτων προηγούμενων σαρώσεων

Κάτω από το μενού μπορούμε να δούμε την κατάσταση των 3 απομακρυσμένων εικονικών υπολογιστών. Αν υπάρχει πρόβλημα επικοινωνίας με κάποιον από τους υπολογιστές, θα αναγράφεται στο συγκεκριμένο σημείο. Να σημειώσουμε πως το εργαλείο δεν επιτρέπει την καταχώρηση νέας σάρωσης αν δεν είναι σε κατάσταση αναμονής όλοι οι απομακρυσμένοι υπολογιστές.

Παρακάτω υπάρχει μια ειδικά διαμορφωμένη φόρμα για να καταχωρηθούν οι παράμετροι της νέας σάρωσης που επιθυμούμε να υποβάλουμε. Δίπλα από το κάθε πεδίο υπάρχει σύντομη επεξήγηση της συγκεκριμένης παραμέτρου με τις διαθέσιμες επιλογές ή και παραδείγματα που μπορούν να καταχωρηθούν.



Εικόνα 22: Η οθόνη ανασκόπησης αποτελεσμάτων προηγούμενων σαρώσεων (Archive) του βοηθητικού εργαλείου.

Η οθόνη που βλέπει ο χρήστης όταν επιλέξει να δει την ανασκόπηση αποτελεσμάτων προηγούμενων σαρώσεων (*Archive*).

Αποτελείται από 3 στήλες:

1. Ημερομηνία και ώρα που καταχωρήθηκε η σάρωση
2. Ο μοναδικός αριθμός JobID που ανατίθεται σε κάθε νέα σάρωση
3. Το email του παραλήπτη που θα σταλεί η αναφορά των αποτελεσμάτων μόλις τελειώσει η σάρωση

Επιπλέον, αν ο χρήστης επιλέξει τον σύνδεσμο που υπάρχει πάνω στον μοναδικό αριθμό JobID, τότε θα μεταφερθεί σε μια νέα οθόνη που θα μπορεί να δει τα αποτελέσματα της συγκεκριμένης σάρωσης. Αν δεν έχουν ολοκληρώσει όλοι οι εικονικοί υπολογιστές την επεξεργασία του κομματιού του συνόλου δεδομένου που τους αντιστοιχεί, τότε στην συγκεκριμένη οθόνη εμφανίζονται μόνο τα αποτελέσματα που έχουν ολοκληρωθεί μέχρι εκείνη την στιγμή.

Ενδεικτικά αποτελέσματα

Παρακάτω παρατίθενται ενδεικτικά μια σάρωση της ανίχνευσης των re-tweets και μια σάρωση της ανίχνευσης των απαντήσεων (replies) των tweets που εκτελέσαμε. Οι παραμέτροι που χρησιμοποιήθηκαν για κάθε σάρωση αναγράφονται στην αρχή της κάθε αναφοράς.

JobID 1363006293928, Complete report. (*Σάρωση ανίχνευσης re-tweet*)

Social Sensing build version 24, thesis1
social-sensing-mode: 1
keywords: h1n1,swine flu,swineflu|vaccine,vaccination,vaccinations,vaccinated,vaccines
extended-keywords:
spreading-hours: 200
spreading-tweets: 10
filter-mode: 1
extra: none

---------- /home/user/tweets2009-06-sorted.txt ----------

---------- /home/user/tweets2009-07.txt ----------
count: 94, date: 2009-07-09 13:38:29: H1N1 vaccine should be ready by mid-October http://bit.ly/14x8Gx
count: 15, date: 2009-07-13 18:24:51: WHO says health workers priority for H1N1 vaccine http://bit.ly/HWcFI Is #swineflu still a worry?
count: 16, date: 2009-07-17 11:41:29: WHO says health workers priority for H1N1 vaccine http://bit.ly/HWcFI Is #swineflu still a worry?

---------- /home/user/tweets2009-08.txt ----------
count: 12, date: 2009-08-04 18:07:14: A closer look at 1 of the toxic ingredients to be used in upcoming #swineflu vaccines: SQUALENE http://bit.ly/77ki6
count: 14, date: 2009-08-06 10:14:44: #H1N1 Safety of pandemic vaccines http://bit.ly/142unH
count: 20, date: 2009-08-11 15:58:28: My latest about #swineflu: Swine Flu is NOT the Problem - It is the Vaccine that May Harm or Kill You http://bit.ly/JzzmM
count: 11, date: 2009-08-18 00:54:17: The Shocking Truth About The Swine Flu Vaccine http://is.gd/2lPuc
count: 11, date: 2009-08-21 20:36:46: New YouTube video: H1N1 Vaccine Questions? ...ask Dr. Anne http://tinyurl.com/ma6ne8
2013.03.11-21:51:22

Social Sensing build version 24, thesis2
social-sensing-mode: 1
keywords: h1n1,swine flu,swineflu|vaccine,vaccination,vaccinations,vaccinated,vaccines
extended-keywords:
spreading-hours: 200
spreading-tweets: 10
filter-mode: 1
extra: none

---------- /home/user/tweets2009-09.txt ----------

---------- /home/user/tweets2009-10.txt ----------
count: 11, date: 2009-10-01 08:48:00: CDC States H1N1 Vaccine May Maim and Kill 30,000 Americans http://is.gd/3MNz0 #tcot
count: 11, date: 2009-10-01 15:40:31: Watch press brief on 2009 #H1N1 flu and vaccine distribution led by Dr. Ann Schuchat. Today @ 12:00 noon EDT on flu.gov/live.
count: 13, date: 2009-10-06 13:55:51: Watch the CDC Director's live press brief on 2009 #H1N1 flu and vaccine distribution. Today @ 1:30 p.m. ET on flu.gov/live.
count: 13, date: 2009-10-16 05:09:53: Swine Flu Shots Revive a Debate About Vaccines http://bit.ly/15QdQ2
count: 15, date: 2009-10-21 16:36:47: Is the swine flu vaccine safe? We sort the myths from reality http://bit.ly/4eAkgO #H1N1
2013.03.11-19:07:15

Social Sensing build version 24, thesis3
social-sensing-mode: 1
keywords: h1n1,swine flu,swineflu|vaccine,vaccination,vaccinations,vaccinated,vaccines
extended-keywords:
spreading-hours: 200
spreading-tweets: 10
filter-mode: 1
extra: none

---------- /home/user/tweets2009-11-sorted.txt ----------
count: 25, date: 2009-11-10 17:20:31: Find locations near you for both the seasonal and H1N1 flu vaccine http://bit.ly/4nOr53
count: 40, date: 2009-11-14 01:47:26: Chinese state media reports that two people have died after receiving the H1N1 swine flu vaccine.
count: 14, date: 2009-11-17 15:40:50: Forget the Vaccines for H1N1, Here's Something Better http://www.top-wellness-products.com/blog/

---------- /home/user/tweets2009-12-sorted.txt ----------
count: 22, date: 2009-12-15 17:05:23: You're in good hands: Govt recalls 800,000 doses of kids' swine flu vaccine - http://is.gd/5oI3T
count: 43, date: 2009-12-15 22:56:50: 800,000 H1N1 vaccine doses for children recalled because they aren't as potent as they should be http://bit.ly/59c8sL
count: 15, date: 2009-12-19 00:42:04: if they created a vaccine for greed, i believe many other vaccines would be of little purpose, and dreaded swine flu, jus ...
2013.03.11-18:06:12

JobID 1362319208220, Complete report. (*Σάρωση ανίχνευσης απαντήσεων*)

Social Sensing build version 22, thesis1
social-sensing-mode: 2
keywords: i,myself,me|feeling blue,depressed,depression,feeling blue,feeling of sadness,low mood,melancholy,mood depression
spreading-hours: 48
spreading-tweets: 30
filter-mode: 2

---------- /home/user/tweets2009-06-sorted.txt ----------

---------- /home/user/tweets2009-07.txt ----------
count: 36, date: 2009-07-27 13:21:26newT: 2009-07-29 13:21:27: t_isfortammy: i dont know what to do. I thought i was over my depression but it looks like i'm anything but

count: 56, date: 2009-07-29 01:21:15newT: 2009-07-31 01:21:16: petrilude: Seriously, its annoying when you aren't bouncing off the walls you get 8340343 comments "are you ok? sad? depressed?" Shutup! I'm RELAXED!

count: 100, date: 2009-07-29 03:14:05newT: 2009-07-31 03:14:06: _harrypotter_: OOC: Sorry for all the random music quotes, guys. I'm melancholy!Potter tonight. O.o

count: 40, date: 2009-07-29 06:39:02newT: 2009-07-31 06:39:03: eyeeye0702: I tittybit depressed I know all the words to "Over and Over" not a lot b/c most of is the same.

count: 81, date: 2009-07-29 10:11:35newT: 2009-07-31 10:11:36: leamaw: I wish they wouldn't do 'changing tracks' on Radio 1 - it always makes me totally depressed and is uncool :-(

count: 42, date: 2009-07-29 13:27:20newT: 2009-07-31 13:27:21: yiannopoulos: 'Without my disability allowance I'm left with just £210 incapacity benefit which I get because of my depression.'

count: 70, date: 2009-07-29 13:58:21newT: 2009-07-31 13:58:22: pdurham: really really tired of these meds messing with me. I'd rather be depressed then sick!

count: 51, date: 2009-07-29 15:34:33newT: 2009-07-31 15:34:34: imustbplumcrazy: really depressed....2 weeks to find a job or i'm back in HELL at my parents house in the country! :-(going for a walk...

count: 34, date: 2009-07-29 21:35:59newT: 2009-07-31 21:36:00: themanwhofell: Whenever I'm depressed I think of Eric Djemba-Djemba and his ridiculous name and I cheer up a little.

---------- /home/user/tweets2009-08.txt ----------
count: 43, date: 2009-08-01 01:02:39newT: 2009-08-03 01:02:40: geekygirl602: Damn, I meant, depressed about missing the fun on twitter today!! Gosh, mushy brain.

count: 34, date: 2009-08-01 01:55:26newT: 2009-08-03 01:55:27: thepioneerwoman: Old episodes of Jon & Kate make me trip and fall into a deep depression. But then Aaden comes on the screen and makes me want to live again.

count: 53, date: 2009-08-01 02:49:07newT: 2009-08-03 02:49:08: arianna_skye: was kinda

depressed earlier. I hit a bunny on the way home from work

count: 35, date: 2009-08-01 04:25:32newT: 2009-08-03 04:25:33: greenbean55: #question is it reason for concern when your fingertips get depressed and are slow to regain shape? <3 xoxo #noh8 (i think it's "txt thumb")

count: 31, date: 2009-08-01 14:00:48newT: 2009-08-03 14:00:49: morethansher: he makes me so mad till the point i feel depressed. fml?

count: 75, date: 2009-08-01 18:03:36newT: 2009-08-03 18:03:37: tallulahdarling: Just found my print out ticket for Mary Stuart and sank into a slight depression. I can't believe I won't ever see this show again. Le sigh.

count: 47, date: 2009-08-01 20:34:20newT: 2009-08-03 20:34:21: alcoholharmony: Ok. That wedding subconsciously depressed me half to death. I need some Phyllis right about now.

count: 71, date: 2009-08-02 06:50:45newT: 2009-08-04 06:50:46: metal_death: ha we had this huge disscussion tonight. it was all about how me and my brother look all depressed and shit all the time and at first..

count: 35, date: 2009-08-02 16:21:12newT: 2009-08-04 16:21:13: soycamo: I'm still wondering if I should even be drinking. Alcohol is a depressant, I'm depressed, seems like a bad combo

count: 40, date: 2009-08-03 00:33:48newT: 2009-08-05 00:33:49: oherrol: oh man. reading the public responses to an op-ed piece on gay marriage is making me So Incredibly Depressed. anyone got a tissue? -s

count: 38, date: 2009-08-03 01:39:05newT: 2009-08-05 01:39:06: anastasiavanite: I'm seriously done listening to cellcasts for good. It's only making me depressed that i'm not actually there.

count: 95, date: 2009-08-03 02:52:17newT: 2009-08-05 02:52:18: kygirlbritt: OK, So Now I'm Juss Gettin' Depressed. Frickin' Guys.

count: 41, date: 2009-08-03 07:55:58newT: 2009-08-05 07:55:59: beatabish: when i go to sleep/passout in the middle of a twitter chat, i wake up the next day depressed. #twittershy.

count: 36, date: 2009-08-03 09:05:29newT: 2009-08-05 09:05:30: think_and_type: There's an almost poetic beauty in the melancholy of listening a song that seems made for you. Or I became a musical masochist.

count: 33, date: 2009-08-03 09:39:00newT: 2009-08-05 09:39:01: happygigi: okay i am done now stalking bands and being depressed. i am finally going to go to bed.

count: 46, date: 2009-08-03 13:02:43newT: 2009-08-05 13:02:44: danger_skies: Should I make myself depressed and go to the google map street view of DisneyLand again?

count: 56, date: 2009-08-03 14:09:43newT: 2009-08-05 14:09:44: rihannaeyes: My owner is depressed right now. I don't know if she steppin out today. :'(

count: 34, date: 2009-08-03 15:15:44newT: 2009-08-05 15:15:45: missmonnie: someone I used to know just had a baby and now I'm really depressed and going to bed

5.3 Υλοποιηθέντες μηχανισμοί

Σε αυτήν την ενότητα παρουσιάζονται οι μηχανισμοί που υλοποιήθηκαν προκειμένου να πετύχουμε την μεθοδολογία που περιγράψαμε στην ενότητα 4.2. Οποιαδήποτε τεχνικά θέματα και περιορισμοί εμφανίστηκαν στην προσπάθεια να υλοποιήσουμε την προτεινόμενη μεθοδολογία περιγράφονται λεπτομερώς παρακάτω.

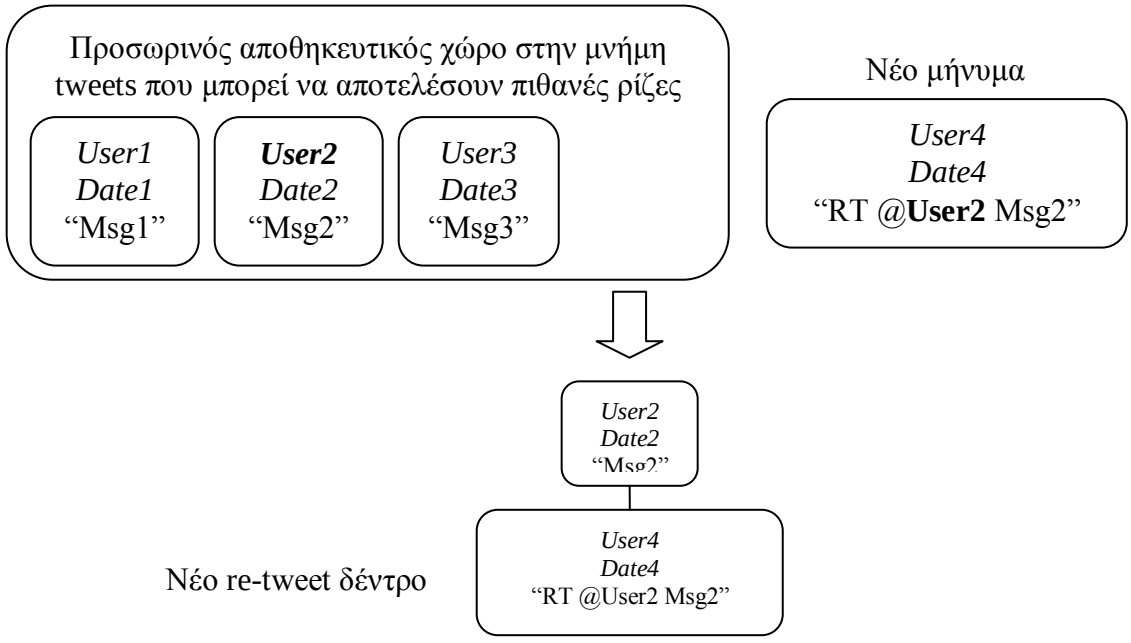
Βήμα 1: Φιλτράρισμα των σχετικών tweets και δημιουργία των re-tweet δέντρων

Στην προσπάθεια να εκτελέσουμε το βήμα της δημιουργίας των re-tweet δέντρων παρατηρήσαμε πως το πλήθος τους ήταν πολύ μεγάλο, τόσο μεγάλο που δεν χωρούσε στην κύρια μνήμη των υπολογιστών που διαθέτουμε. Αυτό οφειλόταν εν μέρη στο γεγονός πως και το σύνολο που διαθέταμε ήταν πολύ μεγάλο (μεγέθη αρχείων 2.7 GB μέχρι και 20.1 GB).

Γι'αυτόν τον λόγο θεωρήσαμε αναγκαίο πριν να επεξεργαζόμαστε ένα μήνυμα tweet να το φιλτράρουμε από ένα σύνολο από λέξεις κλειδιά. Επιλέγαμε ένα δημοφιλές γεγονός που συνέβηκε και θέλαμε να εστιάσουμε, όπως το ξέσπασμα του ιού H1N1, και δημιουργούσαμε ένα σύνολο από σχετικές λέξεις κλειδιά και αν χρειαζόταν και των συνωνύμων τους.

Αφού ένα μήνυμα περάσει επιτυχώς από το φιλτράρισμα των λέξεων κλειδιών, άρα είναι ένα σχετικό μήνυμα, τότε ελέγχεται κατά πόσο είναι ένα μήνυμα re-tweet. Αν είναι, τότε τοποθετείται στην αντίστοιχη θέση στο δέντρο re-tweet που ανήκει, κάτω από το tweet-πατέρα του. Αν δεν είναι μήνυμα re-tweet, τότε τοποθετείται σε ένα προσωρινό αποθηκευτικό χώρο στην μνήμη διότι μπορεί να αποτελέσει πιθανή ρίζα για ένα νέο re-tweet δέντρο. Να σημειώσουμε πως δημιουργούμε ένα νέο δέντρο, μόνο όταν βρούμε περισσότερους από έναν κόμβους που να ανήκουν στο δέντρο.

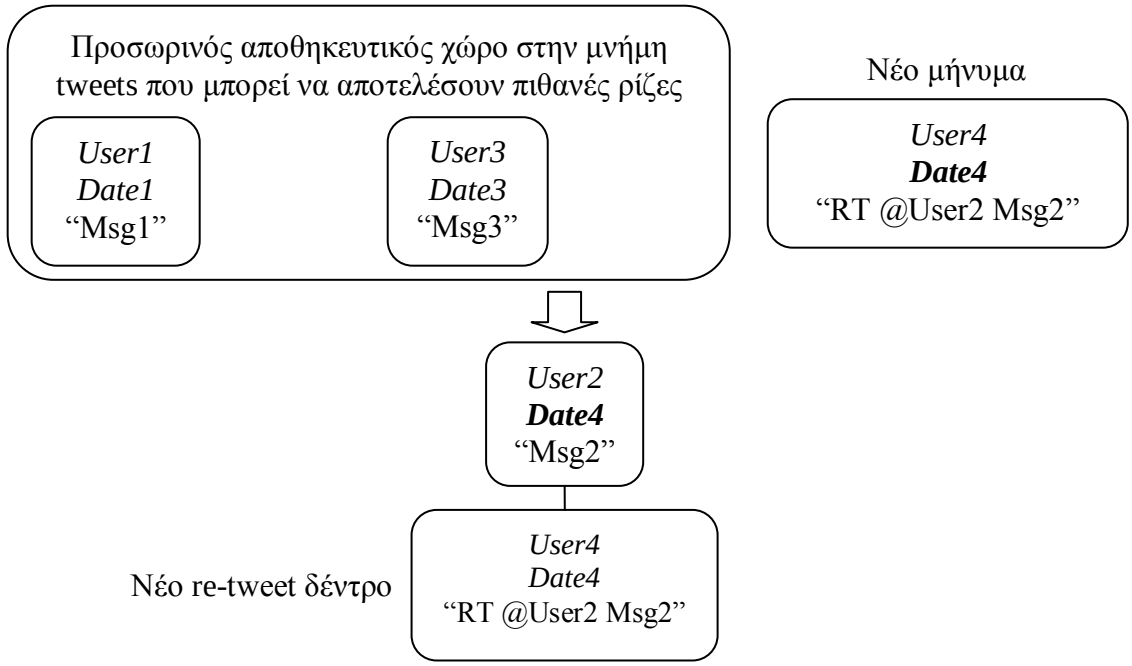
Για παράδειγμα, παρουσιάζουμε παρακάτω γραφικά τι συμβαίνει όταν ανιχνευθεί ένα μήνυμα re-tweet και τη δημιουργία ενός νέου δέντρου.



Εικόνα 23: Δημιουργία ενός re-tweet δέντρου.

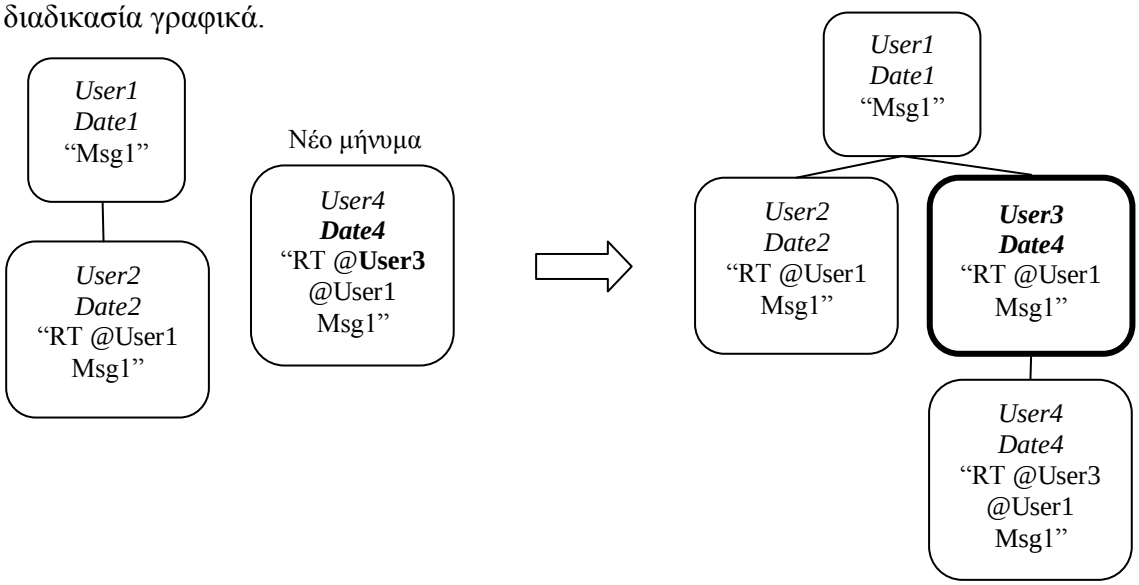
Το πρόβλημα που αντιμετωπίσαμε στη συγκεκριμένη διαδικασία ήταν ότι πολύ συχνά εμφανιζόταν ένα μήνυμα re-tweet για το οποίο δεν υπήρχε το μήνυμα πατέρας του στον προσωρινό αποθηκευτικό χώρο. Αυτό οφείλεται στην διαδικασία που χρησιμοποιήθηκε για την δημιουργία του συνόλου δεδομένων η οποία μπορεί να μην αποθήκευε όλα τα δημόσια tweets. Να αναφερθεί πως το σύνολο δεδομένων περιέχει, όπως προαναφέρθηκε, μόνο το 20-30% των δημοσίων tweets που είχαν γίνει το συγκεκριμένο διάστημα, άρα η απουσία κάποιων μηνυμάτων είναι αναμενόμενη και δικαιολογημένη.

Για να αντιμετωπίσουμε το παραπάνω πρόβλημα, δημιουργούσαμε το νέο δέντρο όπως και πριν αλλά κάναμε την παραδοχή πως ο χρόνος που δημοσιεύτηκε το tweet-ρίζα ήταν η ίδια με το μήνυμα re-tweet, όπως φαίνεται γραφικά παρακάτω.



Εικόνα 24: Δημιουργία ενός re-tweet δέντρου όταν δεν υπάρχει το tweet-ρίζα στο σύνολο δεδομένων.

Κατά αναλογία, όταν βρίσκαμε ένα μήνυμα το οποίο δεν είχε την κατάλληλη ιεραρχία για να τοποθετηθεί στην θέση του, δημιουργούσαμε τους κόμβους που χρειάζονταν με την παραδοχή πως ο χρόνος που δημοσιεύτηκαν ήταν ο ίδιος. Παρακάτω φαίνεται αυτή η διαδικασία γραφικά.



Εικόνα 25: Δημιουργία ενός re-tweet δέντρου όταν δεν υπάρχει το tweet-πατέρας στο σύνολο δεδομένων.

Βήμα 2: Εύρεση των σημασιολογικά κοντινών δέντρων

Αφού τελειώσει το προηγούμενο βήμα, καταλήγουμε με έναν αριθμό από re-tweet δέντρα. Σκοπός μας σε αυτό το βήμα είναι ο εντοπισμός των δέντρων που αναφέρονται στο ίδιο θέμα συζήτησης. Στο τέλος του βήματος αυτού πρέπει να έχουμε έναν αριθμό από σύνολα τα οποία θα περιέχουν μόνο σημασιολογικά παρόμοια re-tweet δέντρα. Για να το πετύχουμε αυτό προσπαθήσαμε να εντοπίσουμε ποια δέντρα έχουν σημασιολογικά παρόμοιες ρίζες.

Η σημασιολογική ομοιότητα μεταξύ κειμένων για να διαπιστωθεί αν δύο αλφαριθμητικά αναφέρονται στο ίδιο θέμα είναι μια σχετικά δύσκολη διαδικασία που δε μπορεί να εφαρμοστεί με μεγάλα ποσοστά επιτυχίας σε γενική μορφή. Υπάρχει σοβαρή έρευνα στο θέμα της σημασιολογικής ομοιότητας μεταξύ δύο κειμένων καθώς και βιβλιοθήκες και εργαλεία που θα μπορούσαν να χρησιμοποιηθούν για να πετύχουμε ανίχνευση μιας πιο γενικής σημασιολογικής ανίχνευσης.

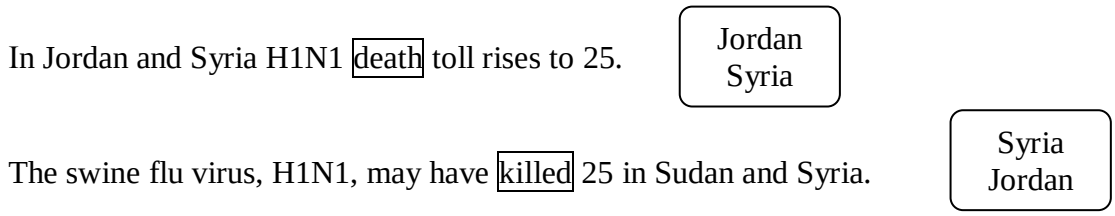
Για λόγους απλότητας επιλέξαμε να υλοποιήσουμε την ανίχνευση της σημασιολογικής ομοιότητας με μια συγκεκριμένη μορφή μόνο. Επιλέξαμε να εστιάσουμε στον τομέα της υγείας δεδομένης της σημαντικής επίδρασης που μπορεί να έχει μια λανθασμένη πληροφορία σε επίπεδο ατομικό, κοινωνικό αλλά και οικονομικό. Για παράδειγμα, σε περιπτώσεις εξάπλωσης μιας επιδημίας, η διάδοση λανθασμένων πληροφοριών όπως μιας φήμης μπορεί εύκολα να προκαλέσει πανικό. Έτσι, αναληθή μηνύματα για εμφάνιση του ιού σε μια περιοχή συνοδευόμενα από τιμές για το πλήθος των θυμάτων του μπορεί να έχει σημαντικές επιπτώσεις στη ζωή σε εκείνη την περιοχή και να οδηγήσει μερίδα του πληθυσμού σε βεβιασμένες κινήσεις.

Για να κρίνουμε αν δύο re-tweet δέντρα είναι σημασιολογικά παρόμοια σύμφωνα με τον ορισμό μας, μελετούμε τα tweet-ρίζες τους. Για τα μηνύματα τα οποία παρουσιάζουν υψηλή σημασιολογική ομοιότητα, απομονώνουμε τους όρους οι οποίοι αφορούν σε τοποθεσίες. Δημιουργούμε έναν πίνακα για κάθε μήνυμα στον οποίο αποθηκεύουμε όλες τις χώρες που εντοπίζονται στο κάθε μήνυμα υπό εξέταση. Αν ο πίνακας με τις χώρες είναι ο ίδιος και στα δύο μηνύματα, τότε θεωρούμε πως τα δύο μηνύματα είναι σημασιολογικά παρόμοια και αναφέρονται στο ίδιο θέμα.

Τότε δημιουργούμε ένα νέο σύνολο τοποθετώντας τα δύο αυτά δέντρα. Αν στην πορεία εντοπιστεί κάποιο άλλο δέντρο με τον ίδιο πίνακα χωρών και την συνύπαρξη των απαιτούμενων λέξεων κλειδιών, τότε προστίθεται και αυτό στο σύνολο.

Παρακάτω παρουσιάζεται γραφικά ένα παράδειγμα για το οποίο τηρούνται τα κριτήρια σημασιολογικής ομοιότητας όπως τα έχουμε ορίσει. Πιο συγκεκριμένα, σε μικρά ορθογώνια φαίνονται οι απαραίτητες λέξεις κλειδιά και σε μεγαλύτερα ορθογώνια φαίνεται το σύνολο των χωρών που εντοπίστηκαν στο συγκεκριμένο μήνυμα. Να σημειώσουμε πως η σειρά εμφάνισης των χωρών δεν έχει σημασία.

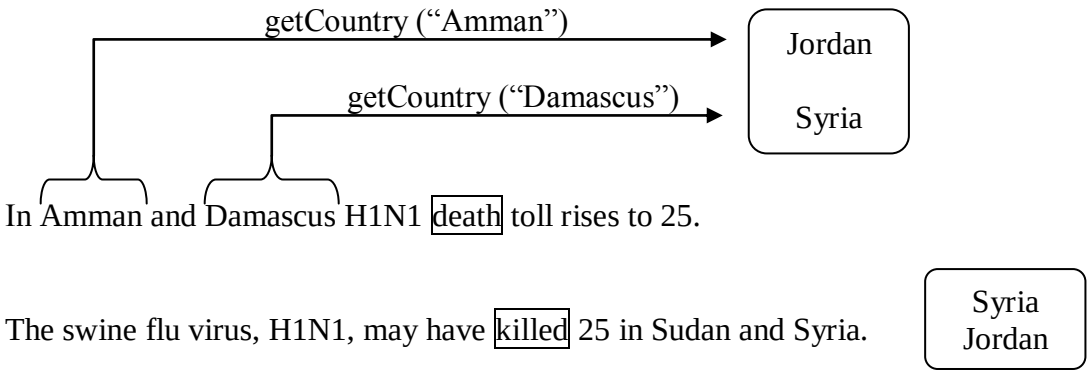
Σημασιολογικά Συσχετισμένες λέξεις κλειδιά: {died,death,dead,killed}



Για να κάνουμε τον εντοπισμό των σημασιολογικά παρόμοιων μηνυμάτων πιο γενικό, όταν εντοπιστεί μια πόλη ή μια περιοχή σε ένα μήνυμα τότε, μέσω της κατάλληλης υπηρεσίας που περιγράφηκε σε προηγούμενη ενότητα, βρίσκουμε την χώρα στην οποία ανήκει το συγκεκριμένο τοπωνύμιο και έπειτα συγκρίνουμε τους δύο πίνακες χωρών αν είναι οι ίδιοι.

Παρακάτω παρουσιάζουμε γραφικά το προηγούμενο παράδειγμα για το οποίο τηρούνται τα κριτήρια σημασιολογικής ομοιότητας, αλλά μερικά τοπωνύμια δεν είναι χώρες αλλά πόλεις. Σε αυτή την περίπτωση, μόλις εντοπιστεί ότι ένα τοπωνύμιο πρόκειται για πόλη γίνεται η σχετική ερώτηση στην κατάλληλη υπηρεσία και στον τελικό πίνακα τοποθετείται η χώρα στην οποία ανήκει.

Σημασιολογικά Συσχετισμένες λέξεις κλειδιά: {died,death,dead,killed}



Βήμα 3: Μελέτη ανεξαρτησίας των ριζών

Με την ολοκλήρωση του προηγούμενου βήματος προκύπτει ένας αριθμός από σύνολα τα οποία θα περιέχουν σημασιολογικά παρόμοια re-tweet δέντρα. Το κάθε σύνολο περιέχει δέντρα τα οποία αναφέρονται σε ένα συγκεκριμένο θέμα. Σκοπός μας σε αυτό το βήμα είναι να μελετήσουμε κατά πόσο τα δέντρα σε κάθε σύνολο είναι ανεξάρτητα ή είναι επηρεασμένα το ένα από το άλλο.

Για να ερευνήσουμε την ανεξαρτησία των δέντρων, για κάθε σύνολο ελέγχουμε ανά δύο τα re-tweet δέντρα για την ύπαρξη εξαρτήσεων μεταξύ τους. Πιο συγκεκριμένα, θεωρούμε πως δύο δέντρα με ρίζες τους κόμβους R1 και R2 δεν είναι ανεξάρτητες πηγές για το συγκεκριμένο θέμα αν έστω και ένας από τους παρακάτω κανόνες, που αναλυθήκαν λεπτομερώς στην ενότητα 4.2, ισχύει:

1. *Ελέγχουμε αν ο χρήστης που δημοσίευσε το R1 βρίσκεται στο re-tweet δέντρο της R2. Αν βρίσκεται, τότε συγκρίνουμε τη χρονική στιγμή κατά την οποία έκανε re-tweet με αυτήν που έγινε η αρχική δημοσίευση.*

2. *Οι χρήστες που δημοσίευσαν τα R1 και R2 συνυπάρχουν σε κάποιο άλλο δέντρο re-tweet και επιπλέον βρίσκονται στο ίδιο μονοπάτι από την ρίζα.*
   Σε αυτή την περίπτωση υπολογίζουμε την απόστασή τους και εφαρμόζουμε κατώφλι.

3. *Οι χρήστες που δημοσίευσαν τα R1 και R2 συνυπάρχουν σε οποιοδήποτε άλλο δέντρο re-tweet.*

Σε περίπτωση που ανιχνευθεί μια εξάρτηση, τότε καταγράφεται στο αρχείο εξόδου ώστε να μπορεί να ελεχθεί από τον χρήστη η εγκυρότητά της.

5.4 Εφαρμογή Μηχανισμών

Παρακάτω παρουσιάζεται η έξοδος του λογισμικού που υλοποιεί την προτεινόμενη μεθοδολογία για ένα αρχείο εισόδου του συνόλου δεδομένων το οποίο περιέχει μηνύματα που αναγνωρίζονται για σημασιολογική ομοιότητα, όπως την ορίσαμε. Για κάθε βήμα, υπάρχει αναλυτική έξοδος των αποτελεσμάτων του.
Το παράδειγμα αυτό αναζητά αναληθείς πληροφορίες (φήμες) σχετικές με την επιδημία του ιού H1N1.

```
Social Sensing Rumors, netcyrax-vaio, 2013.03.30-21:20:18, build version 1
social-sensing-rumors-mode: 1
keywords: swine flu,pig influenza,Swine influenza,hog flu,pig flu,h1n1,swineflu,swine-flu
extra: none

--------- PROCESSING /dos/Dataset/test2.txt ---------

-----------------------------------------------------------------
Step 1+2: Filter the tweets and create RT-trees
-----------------------------------------------------------------
- Tree 0: Many died and have been found in Nicosia and Atlanta from Swine influenza.
userc: 2009-10-16 18:24:51
        userd: 2009-10-16 18:24:51
        userb: 2009-10-16 18:24:52
                usere: 2009-10-16 18:24:52
                        userf: 2009-10-16 18:24:52

- Tree 1: More than 100 have died in Manila, h1n1 to blame.
techchat: 2009-07-01 03:31:51
        amous: 2009-10-13 18:24:51
        imagpa: 2009-10-13 18:24:52
                jamokies: 2009-10-13 18:24:52
                        ishowsd: 2009-10-13 18:24:52
                        dashchang: 2009-10-13 18:24:54
                                twishtj: 2009-10-13 18:24:54
                                jamokie: 2009-10-13 18:24:55
        soulshow: 2009-10-13 18:24:53
                barbaran61: 2009-10-13 18:24:53
                        fishshark: 2009-10-13 18:24:53

- Tree 2: Breaking News: 100 died in Philippines from H1N1!
inquirerdotnet: 2009-07-01 03:36:53
        user1: 2009-07-01 03:36:53
        userc: 2009-07-20 14:53:44
                usera: 2009-07-20 14:53:44
        user3: 2009-07-27 20:46:22
                user6: 2009-10-02 22:08:12
        alistairisrael: 2009-07-30 05:49:56
                franz_see: 2009-08-02 22:08:12
```

user5: 2009-09-04 20:33:34
bla: 2009-10-02 22:08:13
blu: 2009-10-02 22:08:13
user7: 2009-10-02 22:08:13

- Tree 3: It is said that h1n1 has cuased people died in Cyprus and Malta.
techchat: 2009-10-14 18:24:51
amous: 2009-10-14 18:24:51

- Tree 4: A lot of people died from swine flu, reported from Nicosia and United States!
usera: 2009-10-15 18:24:50
userb: 2009-10-15 18:24:51
userc: 2009-10-15 18:24:52

- Tree 5: Some people died from H1N1 in Cyprus and New York.
imagpa: 2009-10-14 18:24:52
ishowsd: 2009-10-14 18:24:52
techchat: 2009-10-14 18:24:53
soulshow: 2009-10-14 18:24:53
barbaran61: 2009-10-14 18:24:53
fishshark: 2009-10-14 18:24:53

Step 3: Find semantically related tweets

Semantically Related Groups:

Group 0:
- userc: Many died and have been found in Nicosia and Atlanta from Swine influenza.
- usera: A lot of people died from swine flu, reported from Nicosia and United States!
- imagpa: Some people died from H1N1 in Cyprus and New York.

Group 1:
- techchat: More than 100 have died in Manila, h1n1 to blame.
- inquirerdotnet: Breaking News: 100 died in Philippines from H1N1!

Step 4: Find out and print any relations between tweet-roots

Now processing Group 0:
- userc: Many died and have been found in Nicosia and Atlanta from Swine influenza.
- usera: A lot of people died from swine flu, reported from Nicosia and United States!
- imagpa: Some people died from H1N1 in Cyprus and New York.

Possible dependency 4.1 (η R1 βρίσκεται στο δέντρο διάδοσης της R2 για το σημασιολογικά κοντινό tweet που εντοπίστηκε στο βήμα 3):
R1: 2009-10-15 18:24:50: usera: A lot of people died from swine flu, reported from Nicosia and United States!
R2: 2009-10-16 18:24:51: userc: Many died and have been found in Nicosia and Atlanta from Swine influenza.
Found R2: 2009-10-15 18:24:52: userc: RT @userB RT @userA A lot of people died from swine flu, reported from Nicosia and United States!

Possible dependency 4.3 (η R1 βρίσκεται στο δέντρο διάδοσης της R2 σε οποιοδήποτε άλλο δέντρο, βρίσκονται στο ίδιο μονοπάτι από την ρίζα και υπολογίζουμε την απόστασή τους):
in Tree 2: Breaking News: 100 died in Philippines from H1N1!
R1: 2009-07-20 14:53:44: userc: Breaking News: 100 died in Philippines from H1N1!
R2: 2009-07-20 14:53:44: usera: RT @userC: Breaking News: 100 died in Philippines from H1N1!
Distance: 1

Possible dependency 4.2 (η R1 και η R2 συνυπάρχουν σε οποιοδήποτε δέντρο διάδοσης):
in Tree 2: Breaking News: 100 died in Philippines from H1N1!
R1: 2009-07-20 14:53:44: usera: RT @userC: Breaking News: 100 died in Philippines from H1N1!
R2: 2009-07-20 14:53:44: userc: Breaking News: 100 died in Philippines from H1N1!

Now processing Group 1:
- techchat: More than 100 have died in Manila, h1n1 to blame.
- inquirerdotnet: Breaking News: 100 died in Philippines from H1N1!

2013.03.30-21:20:30

6. Συμπεράσματα και Μελλοντική έρευνα

Σε αυτή την ενότητα, παρουσιάζεται μια γενική αποτίμηση των υλοποιηθέντων μηχανισμών δουλειά. Επιπλέον, κάνουμε μια επισκόπηση της διπλωματικής εργασίας και τέλος συζητάμε κάποιες ιδέες για να βελτιώσουμε την υπάρχουσα δουλειά.

6.1 Μειονεκτήματα και Πλεονεκτήματα

Αφού δοκιμάσαμε τα εργαλεία που αναπτύξαμε με διαφορετικές εισόδους παρατηρήσαμε τα παρακάτω πλεονεκτήματα και μειονεκτήματα της προσέγγισης και του εργαλείου που φτιάξαμε.

Πλεονεκτήματα

- *Για την αναγνώριση των λανθασμένων πληροφοριών / φημών δεν βασιζόμαστε σε κάποιο λεξικό που αναπτύξαμε.*

  Πολλοί ερευνητές στην προσπάθεια τους να αναγνωρίσουν φήμες αναγνώρισαν την ανάγκη ενός λεξικού βάση του οποίου θα αναζητούν τις πιθανές φήμες. Για να το πετύχουν αυτό, ανάλυσαν ένα μεγάλο μέρος από ήδη αναγνωρισμένα μηνύματα παραπληροφόρησης και έφτιαχναν ένα πιθανοτικό μοντέλο βάση του οποίου αποφάσιζαν αν το υπό εξέταση μήνυμα είναι φήμη.

  Αντίθετα, εμείς προσπαθήσαμε να μείνουμε όσο γίνεται πιο ανεξάρτητοι από ένα λεξικό βάση του οποίου θα κρίνουμε αν είναι ένα μήνυμα λανθασμένη πληροφορία ή όχι. Βασιστήκαμε όσο περισσότερο μπορούσαμε στην δομή μεταξύ των μηνυμάτων και στο πώς η διάδοση ενός μηνύματος σε ένα κοινωνικό δίκτυο μπορεί να υποδείξει αν πρόκειται για αληθινή ή ψευδή πληροφορία.

  Αυτό είχε σαν αποτέλεσμα, το μοντέλο που αναπτύξαμε να μπορεί να εφαρμοστεί σε ένα μεγάλο πλήθος περιπτώσεων και όχι μόνο στις περιπτώσεις στις οποίες το λεξικό που δημιουργήσαμε μπορεί να εφαρμοστεί. Επιπλέον, αφού βασιστήκαμε αποκλειστικά στον τρόπο διάδοσης του μηνύματος για να ανιχνεύσουμε την παραπληροφόρηση, έχουμε έναν επαναχρησιμοποιήσιμο τρόπο ανίχνευσης που μπορεί να εφαρμοστεί και σε άλλα περιβάλλοντα πέραν από τα κοινωνικά δίκτυα.

- *Η μεθοδολογία μας μπορεί να ανιχνεύσει προσπάθειες εκούσιας και ακούσιας παραπληροφόρησης που ξεκινούν από τα κοινωνικά δίκτυα.*

  Η μεθοδολογία που αναπτύξαμε μπορεί να ανιχνεύσει λανθασμένες πληροφορίες που διαδίδονται τόσο κακόβουλα όσο και κατά λάθος, *φτάνει οι προσπάθειες παραπληροφόρησης αυτές να ξεκινούν από τα κοινωνικά δίκτυα*. Αυτό επιτυγχάνεται με την εύρεση των πηγών μιας πληροφορίας που διαδίδεται και την μελέτη ανεξαρτησίας μεταξύ των πηγών, όπως αναλύθηκε με λεπτομέρεια στην ενότητα 4.2.

- *Η ταχύτητα σάρωσης του εργαλείου που φτιάξαμε είναι ιδιαίτερα ικανοποιητική.*

  Φτιάξαμε ένα εργαλείο που μπορεί να σαρώσει 476 εκατομμύρια tweets και να εκτελέσει την μεθοδολογία μας μέσα σε 3-5 ώρες, ανάλογα με τις λέξεις κλειδιά που εισάγουμε στην κάθε σάρωση.

  Αυτό το πετύχαμε απλά με το να χωρίσουμε το σύνολο δεδομένων σε τρία διαφορετικά κομμάτια και να το επεξεργαζόμαστε ταυτόχρονα. Να σημειώσουμε πως αυτή η ταχύτητα που πετυχαίνουμε με το εργαλείο μας, είναι χωρίς επιπλέον βελτιώσεις για πολυεπεξεργασία στον κώδικά μας.

- *Το εργαλείο που έχουμε είναι πλήρως παραμετροποιήσημο και προσφέρει σημαντικούς αυτοματισμούς.*

  Όπως δείξαμε και στην ενότητα 5.2.2, το εργαλείο μπορεί να δεχθεί κάθε φορά διαφορετικές εισόδους χωρίς να απαιτεί επαναμεταγλώτιση του πηγαίου κώδικα. Κάθε δυνατή παράμετρος που μπορεί να αλλάξει ορίζεται είτε από εξωτερικό αρχείο, είτε απευθείας από τη γραμμή εντολών. Με τον τρόπο αυτόν, οι μηχανισμοί είναι ανεξάρτητοι και από το πεδίο εφαρμογής τους (υγεία, αθλητισμός, οικονομία, δικαιοσύνη, κοινωνικές συνθήκες, κλπ)

  Αυτό μας επέτρεψε να δημιουργήσουμε σημαντικούς αυτοματισμούς και ευκολίες στην χρήση του εργαλείου. Το πιο σημαντικό είναι πως η παραμετροποίηση μας επέτρεψε να δημιουργήσουμε ένα σύστημα σειράς αναμονής (queue) για τις διάφορες εισόδους που θέλουμε να εκτελέσουμε, ώστε να εκτελούνται διαδοχικά χωρίς να χάνουμε χρόνο. Επιπλέον, δημιουργήσαμε αυτό το σύστημα έτσι ώστε να είναι προσβάσιμο μέσω web-browser από παντού. Τέλος, προσθέσαμε τη δυνατότητα να μας στέλνει τα αποτελέσματα της κάθε σάρωσης με e-mail.

Μειονεκτήματα

- *Η μεθοδολογία μας δεν μπορεί να ανιχνεύσει όλες περιπτώσεις παραπληροφόρησης, ειδικά όταν η λανθασμένη πληροφορία έχει ήδη διαδοθεί εκτός του κοινωνικού δικτύου.*

  Παρατηρήσαμε ότι μερικές περιπτώσεις διάδοσης λανθασμένων πληροφοριών σε κοινωνικά δίκτυα η μεθοδολογία μας αδυνατεί να τις εντοπίσει. Βασιστήκαμε στην υπόθεση πως μια πληροφορία για να είναι αληθής πρέπει να έχει ένα μεγάλο πλήθος από ανεξάρτητες πηγές που αναφέρουν όλες την ίδια πληροφορία. Παρόλα αυτά, εντοπίσαμε τις ακόλουθες τρεις περιπτώσεις στις οποίες η υπόθεσή μας δεν ισχύει:

  1. Σε περίπτωση που η παραπληροφόρηση προέρχεται από ένα παραδοσιακό ΜΜΕ, όπως για παράδειγμα η τηλεόραση. Σε αυτήν την περίπτωση πολλοί χρήστες του κοινωνικού δικτύου θα δημοσιεύσουν μηνύματα μέσω των οποίων θα αναπαράγουν την ψευδή είδηση που άκουσαν στην τηλεόραση. Η μεθοδολογία μας θα αποτύχει να ανιχνεύσει την διάδοση λανθασμένων πληροφοριών αφού θα ανιχνεύσει ένα μεγάλο πλήθος από ανεξάρτητες πηγές που θα αναφέρουν την ίδια είδηση.

  2. Σε περίπτωση που ένας μοναδικός χρήστης αναφέρει μια είδηση που έγινε μάρτυρας και ακόμη δεν έχει διαδοθεί. Χαρακτηριστικό παράδειγμα είναι ο θάνατος του γνωστού τρομοκράτη Bin Laden ο οποίος έγινε γνωστός πρώτα μέσω του Twitter όταν κάποιος γείτονάς του δημοσίευσε ένα μήνυμα στο κοινωνικό δίκτυο πριν ακόμη επιβεβαιωθεί από τις επίσημες αρχές. (36) Σε αυτή την περίπτωση η μεθοδολογία μας θα θεωρήσει πως η πληροφορία που διαδίδεται είναι λανθασμένη αφού πρόκειται για μια είδηση με μια και μοναδική πηγή.

  3. Σε περίπτωση εκούσιας παραπληροφόρησης κατά την οποία χρησιμοποιούνται ψεύτικοι λογαριασμοί για να διαδώσουν μια λανθασμένη πληροφορία. Το πρόβλημα με τους όλο και αυξανόμενους ψεύτικους λογαριασμούς στα κοινωνικά δίκτυα, και ειδικά στο Twitter, είναι ένα θέμα που έχει πάρει μεγάλες διαστάσεις το τελευταίο διάστημα. Μέχρι στιγμής, οι ψεύτικοι λογαριασμοί χρησιμοποιούνται κυρίως για να αυξάνουν το πλήθος των χρηστών που ακολουθούν κάποια δημόσια πρόσωπα (37). Αν όμως χρησιμοποιηθούν για να ξεκινήσουν και να υποστηρίξουν μια φήμη μέσω ενός κοινωνικού δικτύου, τότε η μεθοδολογία μας θα αποτύχει να ανιχνεύσει τη διάδοση λανθασμένων πληροφοριών αφού θα εντοπιστεί ένα μεγάλο πλήθος από ανεξάρτητες πηγές που θα αναφέρονται στο ίδιο θέμα.

- *Πιθανή αποτυχία του δεύτερου βήματος της μεθοδολογίας μας, δηλαδή της ανίχνευσης σημασιολογικά παρόμοιων ριζών για τα re-tweet δέντρα, επηρεάζει αρνητικά όλα τα υπόλοιπα βήματα.*

  Αναφέραμε μόνο το δεύτερο βήμα στη μεθοδολογία μας διότι η ανίχνευση σημασιολογικά παρόμοιων κειμένων είναι μια διαδικασία που δύσκολα μπορεί να αποδώσει ικανοποιητικά αποτελέσματα σε όλες γενικά τις περιπτώσεις. Θεωρούμε πως αυτό το βήμα στην μεθοδολογία μας είναι το πιο ευάλωτο στο να αποτύχει και να μην πάρουμε ικανοποιητικά αποτελέσματα.

  Στην υλοποίηση του εργαλείου μας, εφαρμόσαμε μια απλοποιημένη σημασιολογική ανίχνευση αφού ανιχνεύαμε σημασιολογικά παρόμοια κείμενα μιας συγκεκριμένης μορφής μόνο. Για να εφαρμόσουμε μια πιο γενική σημασιολογική ανίχνευση είναι ένα δύσκολο τεχνικό πρόβλημα που χρειάζεται αποκλειστικά πολύ χρόνο για να αποδώσει ικανοποιητικά αποτελέσματα. Σε επόμενη ενότητα που αφορά την μελλοντική δουλειά που μπορεί να γίνει, προτείνουμε κάποιους τρόπους για να βελτιώσουμε το συγκεκριμένο σημείο.

- *Οι πολλές και διαφορετικές μορφές για re-tweet είχαν ως αποτέλεσμα το εργαλείο μας να μην μπορεί να ανιχνεύσει όλα ανεξαιρέτως τα μηνύματα re-tweet.*

  Όπως αναφέραμε σε προηγούμενη ενότητα υποστηρίξαμε τις πιο συνηθισμένες μορφές από τις μορφές re-tweet που χρησιμοποιούνται. Παρόλα αυτά οι μορφές που δεν ανιχνεύαμε είχαν σαν αποτέλεσμα να χάνουμε πληροφορίες για το πρώτο βήμα της μεθοδολογίας μας, δηλαδή την δημιουργία των re-tweet δέντρων. Θεωρούμε πως το ποσοστό των μηνυμάτων που δεν ανιχνεύαμε δεν ήταν τόσο σημαντικό ώστε να μη παίρνουμε ικανοποιητικά αποτελέσματα, αλλά θα ήταν προτιμότερο να μπορούσαμε να ανιχνεύσουμε όλα τα μηνύματα re-tweet για να δημιουργήσουμε τα re-tweet δέντρα στο πρώτο βήμα.

6.2 Σύνοψη Διπλωματικής Εργασίας

Σε αυτή την διπλωματική εργασία προσπαθήσαμε να δημιουργήσουμε και να υλοποιήσουμε μια μεθοδολογία για την ανίχνευση λανθασμένων πληροφοριών που διαδίδονται σε κοινωνικά δίκτυα.

Ξεκινήσαμε την έρευνα μας με το να κατανοήσουμε τι σημαίνει ακούσια και εκούσια παραπληροφόρηση (disinformation / misinformation). Έπειτα, μελετήσαμε ερευνητικές εργασίες οι οποίες αφορούσαν στην αυτόματη ανίχνευση φημών σε κοινωνικά δίκτυα και την αυτόματη ανίχνευση επιδημιών μέσω παρακολούθησης των κοινωνικών δικτύων (Epidemic Intelligence).

Η προταθείσα μεθοδολογία βασίστηκε στην υπόθεση που διατύπωσαν και προηγούμενοι ερευνητές (20), πως μια πληροφορία που είναι αληθής συνήθως έχει πολλές και ανεξάρτητες πηγές, ενώ μια ψευδής πληροφορία συνήθως ξεκινά με ένα μικρό πλήθος – συχνά εξαρτώμενων μεταξύ τους - πηγών. Εστιάσαμε στο κοινωνικό δίκτυο Twitter για να υλοποιήσουμε την μεθοδολογία μας και καταλήξαμε στα παρακάτω, συνοπτικά, βήματα που αναλύθηκαν λεπτομερώς σε προηγούμενες ενότητες:

1. Δημιουργία re-tweet δέντρων.
2. Εντοπισμός των re-tweet δέντρων τα οποία αναφέρονται στο ίδιο θέμα.
3. Για τα re-tweet δέντρα που αναφέρονται στο ίδιο θέμα, εκτέλεση μιας μελέτης ανεξαρτησίας για τον εντοπισμό των ανεξάρτητων πηγών για κάθε θέμα.

Ακολούθως, αναπτύχθηκε μια βοηθητική διεπαφή για πρόσβαση και δοκιμή των υλοποιηθέντων μηχανισμών, το Social Sensing Portal. Η διαδικτυακή αυτή διεπαφή δίνει πρόβαση στην εφαρμογή, προσβάσιμη από τον web-browser, η οποια εκτελείται από τρείς εικονικούς υπολογιστές σε περιβάλλον Cloud Computing. Ο καθένας έχει αποθηκευμένο ένα κομμάτι του συνόλου των δεδομένων και το επεξεργάζεται ξεχωριστά. Επιπλέον, υπάρχει και ένα συντονιστικό κομμάτι της εφαρμογής υπεύθυνο να ενώνει τα κομμάτια όταν αυτά τελειώσουν την επεξεργασία τους από τον κάθε εικονικό υπολογιστή. Το κομμάτι αυτό της εφαρμογής μπορεί να ανιχνεύσει:

1. Re-tweet δέντρα με πλήθος κόμβων μεγαλύτερο από ένα κατώφλι και με το αρχικό tweet να περιλαμβάνει συγκεκριμένες λέξεις-κλειδιά
2. Μηνύματα που είχαν πλήθος απαντήσεων μεγαλύτερο από ένα κατώφλι και με το αρχικό tweet να περιλαμβάνει συγκεκριμένες λέξεις-κλειδιά

Τέλος, υλοποιήσαμε τη μεθοδολογία που περιγράψαμε πιο πάνω με το εργαλείο Social Sensing Rumors. Πρόκειται για εφαρμογή γραμμένη σε γλώσσα Java που τρέχει στο ίδιο περιβάλλον Cloud Computing όπως περιγράφτηκε πιο πάνω για το βοηθητικό εργαλείο. Λόγω του ότι αναγκαστήκαμε να απλοποιήσουμε τη διαδικασία φιλτραρίσματος των tweets τα οποία αναφέρονται στο ίδιο θέμα, δημιουργήσαμε tweets που μπορούσαν να ανιχνευτούν από τους κανόνες που θέσαμε ως παρόμοια και τα προσθέσαμε στο σύνολο δεδομένων. Έπειτα, εκτελέσαμε διάφορες δοκιμές για να επιβεβαιώσουμε πως η υλοποίηση μας εκτελείται σωστά.

6.3 Μελλοντική έρευνα

Παρακάτω παραθέτουμε μερικές εισηγήσεις που προτείνουμε για να βελτιώσουμε και να επεκτείνουμε την μεθοδολογία και το εργαλείο που αναπτύξαμε.

- *Δυνατότητα ανίχνευσης παραπληροφόρησης σε κοινωνικά δίκτυα σε περισσότερες γλώσσες, όπως στα ελληνικά.*

  Η μεθοδολογία που αναπτύξαμε βασίζει την ικανότητα ανίχνευσης της παραπληροφόρησης κυρίως στον τρόπο εξάπλωσης των μηνυμάτων που έχουν ψευδή πληροφορία. Παρόλα αυτά σε κάποια βήματά της, όπως στον εντοπισμό σημασιολογικά παρόμοιων μηνυμάτων, βασίζεται στην χρήση της γλώσσας.

  Σε έρευνα μας στο διαδίκτυο, εντοπίσαμε διάφορες προσπάθειες να δημιουργηθούν μεταφρασμένες εκδόσεις του WordNet, του σημασιολογικού λεξικού που χρησιμοποιούμε για να εντοπίσουμε συνώνυμα και παρόμοιες λέξεις στην αγγλική γλώσσα που περιγράψαμε στην ενότητα 5.1.2.1. Πιο συγκεκριμένα, εντοπίσαμε το BalkaNet (38) για την ελληνική γλώσσα και το EuroNet για άλλες ευρωπαϊκές γλώσσες (39). Δυστυχώς δεν φαίνεται πως οποιοδήποτε από τις δύο προσπάθειες δεν έχει φτάσει ακόμη σε στάδιο που να μπορεί να χρησιμοποιηθεί.

- *Βελτίωση και γενίκευση της ικανότητας για ανίχνευση σημασιολογικής ομοιότητας μεταξύ δύο μηνυμάτων.*

  Όπως έχουμε περιγράψει και στην ενότητα 5.3, το εργαλείο που έχουμε υλοποιήσει είναι ικανό να ανιχνεύει σημασιολογικά παρόμοια κείμενα που έχουν μια συγκεκριμένη μορφή.

  Θα ήταν επιθυμητό να μπορούσαμε να διευρύνουμε τον μηχανισμό ανίχνευσης μηνυμάτων ιδίου θέματος. Για τον λόγο αυτό, βρήκαμε στο Διαδίκτυο διάφορα εργαλεία που υπόσχονται πως μπορούν να εντοπίσουν σημασιολογική ομοιότητα λέξεων ή και φράσεων. Παρακάτω παρουσιάζουμε μερικά υποσχόμενα εργαλεία που ίσως μπορούν να βελτιώσουν το εργαλείο μας:

  1. *DISCO*: Προσφέρει μια κλάση στην Java που μας επιτρέπει να ανακτήσουμε την σημασιολογική ομοιότητα μεταξύ αυθαίρετων λέξεων. Οι ομοιότητες μεταξύ των λέξεων ορίζονται με βάση στατιστική ανάλυση πολύ μεγάλων συλλογών κειμένου. Το εργαλείο λειτουργεί σε όλα τα δημοφιλή λειτουργικά συστήματα, συμπεριλαμβανομένων των Windows, Linux, Solaris, και MacOS (40).

  2. *CalculateSemanticSimilarity*: Προσφέρει μια βιβλιοθήκη για Java με την οποία μπορούν να ανιχνευτούν οι σημασιολογικές ομοιότητες μεταξύ προτάσεων, ανάλογα με τις κατηγορίες των λέξεων που περιέχουν. Είναι μια βελτίωση του εργαλείου Classifier4j το οποίο χρησιμοποίει μια Vector-Space ανάλυση, η οποία όμως δεν λαμβάνει υπόψη τις σημασιολογικές έννοιες των λέξεων. Επιπλέον βελτίωση είναι πως η συγκεκριμένη βιβλιοθήκη λειτουργεί καλά με σύντομες φράσεις. (41)

- *Χρησιμοποίηση των σχολίων σε κάθε re-tweet δέντρο ως παράμετρο για το αν πρόκειται για παραπληροφόρηση.*

  Όπως αναφέραμε και στην ενότητα 6.1, σε περίπτωση που ένας μόνο χρήστης δημοσιεύσει μια αληθινή πληροφορία που διαδοθεί ευρέως, η μεθοδολογία μας θα θεωρήσει πως η πληροφορία που διαδίδεται είναι λανθασμένη αφού πρόκειται για μια είδηση με μια και μοναδική πηγή.

  Σαν πιθανή λύση, προτείνουμε την ανάλυση των σχόλιων του κάθε tweet σε ένα re-tweet δέντρο. Προηγούμενοι ερευνητές (22) παρατήρησαν πως οι χρήστες τείνουν να αμφιβάλλουν περισσότερο στα σχόλια τους όταν πρόκειται για λανθασμένη πληροφορία. Ανιχνεύοντας την αμφιβολία στα σχόλια των χρηστών μπορούμε να το χρησιμοποιήσουμε σαν επιπλέον δείκτη για την εγκυρότητα της διαδιδόμενης πληροφορίας.

- *Χρησιμοποίηση ζωντανών μηνυμάτων από το Twitter σαν είσοδο για το εργαλείο για έγκαιρη ενημέρωση για πιθανή διάδοση λανθασμένων πληροφοριών.*

  Μπορούμε να χρησιμοποιήσουμε το Streaming API του Twitter για να έχουμε πρόσβαση σε tweets που δημοσιεύονται ζωντανά για κάποιες λέξεις κλειδιά που εμείς επιλέγουμε. Όμως το εργαλείο που δημιουργήσαμε δέχεται σαν είσοδο ένα σύνολο από προκαθορισμένα μηνύματα τα οποία αναλύει χωρίς να μπορεί να δεχθεί νέα δεδομένα πριν τον τερματισμό του. Παρόλο που αναπτύξαμε το εργαλείο μας χρησιμοποιώντας αρχές της τεχνολογίας λογισμικού για να είναι επκτάσιμο, η εισαγωγή την δυνατότητας για να παίρνουμε τα μηνύματα απευθείας από το Twitter θα απαιτούσε ένα αριθμό από αλλαγές για να μπορέσει να χρησιμοποιήσει τα ζωντανά μηνύματα που θα παρείχε το Streaming API.

  Μια ενδιάμεση λύση είναι η χρησιμοποίηση ειδικών εργαλείων τα οποία μαζεύουν μηνύματα tweet για κάποιες λέξεις κλειδιά και τα αποθηκεύουν σε αρχεία. Θα μπορούσαμε να συνδυάσουμε ένα τέτοιο εργαλείο με το εργαλείο που αναπτύξαμε, έτσι ώστε να μαζεύουμε μηνύματα για ένα μικρό χρονικό διάστημα (1-2 μέρες) και έπειτα να τα θέτουμε ως είσοδο στο εργαλείο μας για ανίχνευση πιθανής διάδοσης λανθασμένης πληροφορίας. Με αυτό τον τρόπο θα είχαμε έγκαιρη ενημέρωση για πιθανή παραπληροφόρηση χωρίς να χρειαστούν ιδιαιτέρες αλλαγές στο εργαλείο.

Γλωσσάριο

Ajax: Το ακρωνύμιο σημαίνει Asynchronous Javascript And XML και είναι η τεχνική με την οποία μπορούμε να δημιουργήσουμε πολύ πιο γρήγορες και δυναμικές ιστοσελίδες περιορίζοντας τον όγκο δεδομένων που ανταλλάσσει ο server με τον browser του επισκέπτη. Η ιδιαίτερη τεχνική αυτή επιτρέπει την ανανέωση περιεχομένων μιας ιστοσελίδας χωρίς αυτή να ανανεωθεί ολόκληρη.

API: Το ακρωνύμιο σημαίνει Application Programming Interface και στα ελληνικά είναι γνωστό και ως Διεπαφή Προγραμματισμού Εφαρμογών. Πρόκειται για ένα υπολογιστικό σύστημα, βιβλιοθήκη ή διαδικτυακή εφαρμογή/υπηρεσία που παρέχετε, προκειμένου να επιτρέψει να γίνουν προς αυτό αιτήσεις από άλλα προγράμματα και ανταλλαγή δεδομένων.

ASP.NET: Μια server-side γλώσσα προγραμματισμού για την ανάπτυξη και την παραγωγή δυναμικών ιστοσελίδων. Αναπτύχθηκε από τη Microsoft για να επιτρέψει στους προγραμματιστές την δημιουργία δυναμικών ιστοσελίδων, εφαρμογών και υπηρεσιών του διαδικτύου. Κυκλοφόρησε για πρώτη φορά τον Ιανουάριο του 2002.

Associated Press: Είναι το βασικό και κυρίαρχο ειδησεογραφικό πρακτορείο των ΗΠΑ. Η λειτουργία του έχει μορφή διεθνούς συνεταιριστικής οργάνωσης μη κερδοσκοπικού χαρακτήρα. Εδρεύει στη νήσο Μανχάταν και αποτελεί ένα σύνθετο και πολυδύναμο ΜΜΕ με δικό του δορυφορικό σταθμό, καλύπτοντας ανάγκες εφημερίδων, ραδιοφώνου και τηλεόρασης με ειδησεογραφικά δελτία, ανταποκρίσεις και φωτογραφικό, κινηματογραφικό και βιντεοσκοπικό αρχείο που συντάσσουν και συγκεντρώνουν συνεργαζόμενα μέλη του, διασκορπισμένα σε 245 περίπου γραφεία σε όλο τον κόσμο.

Atom / RSS: Είναι τεχνολογίες που χρησιμοποιούν την XML και επιτρέπουν στους δικτυακούς τόπους να δημιουργούν ροές μηνυμάτων στις οποίες μπορεί ένας επισκέπτης να εγγραφεί για να λαμβάνει τα τελευταία νέα και εξελίξεις που δημοσιεύει η σελίδα. Μια ροή δεδομένων αποτελεί μια περίληψη περιεχομένου που ενημερώνεται τακτικά, η οποία συνοδεύεται από συνδέσμους σε πλήρεις εκδόσεις του συγκεκριμένου περιεχομένου.

C / C++: Η C είναι μια διαδικαστική γλώσσα προγραμματισμού γενικής χρήσης η οποία αναπτύχτηκε στις αρχές της δεκαετίας 1970 στα εργαστήρια Bell Labs για να χρησιμοποιηθεί για την ανάπτυξη του λειτουργικού συστήματος UNIX. Η C++ αναπτύχθηκε ως βελτίωση της C, και οι βελτιώσεις ξεκίνησαν με την προσθήκη κλάσεων, και ακολούθησαν, μεταξύ άλλων, εικονικές συναρτήσεις, υπερφόρτωση τελεστών, πολλαπλή κληρονομικότητα, πρότυπα, κτλ.

Creative Commons Attribution: Η Creative Commons (CC) είναι μια μη κερδοσκοπική οργάνωση αφιερωμένη στην επέκταση του εύρους των πνευματικών έργων που είναι διαθέσιμα για να βασιστούν σε αυτά και άλλα έργα και να μοιραστούν νόμιμα. Η οργάνωση έχει εκδώσει διάφορες άδειες πνευματικών δικαιωμάτων γνωστές ως άδειες Creative Commons. Η άδεια 'Attribution' απαιτεί αναφορά του δημιουργού ή του χορηγού της άδειας.

Digg: Το Digg είναι ένας ιστότοπος που παρέχει εργαλεία για εύκολη ανάρτηση, δημοσίευση και διαμοίραση άρθρων, δημοσιεύσεων και νέων από το Internet. Επιτρέπει στους χρήστες να μοιράζονται οτιδήποτε θέλουν, όπως νέα, προηχογραφημένες εκπομπές (podcasts) ή βίντεο κλπ. Οι καταχωρίσεις λαμβάνουν ψήφο από τους χρήστες και αν συγκεντρώσουν πολλές ψήφους, κερδίζουν προβολή και δημοσιότητα.

Emoticons: Επίσης γνωστά και ως smileys , είναι η συντομογραφία των λέξεων emotion icons. Συνδυασμοί χαρακτήρων και σημείων στίξης που διαμορφώνουν μικρές εικόνες που οι χρήστες μπορούν να χρησιμοποιήσουν για να διαμορφώσουν πρόσωπα ανάλογα με τα συναισθήματα τους.

Facebook Like: Είναι ένα χαρακτηριστικό της υπηρεσία κοινωνικής δικτύωσης Facebook, όπου οι χρήστες μπορούν να δηλώσουν πως τους αρέσει το περιεχόμενο που είναι αναρτημένο στο κοινωνικό δίκτυο όπως σχόλια, φωτογραφίες, σύνδεσμοι, κτλ. Η λειτουργία αυτή ενεργοποιήθηκε την 9η Φεβρουαρίου του 2009.

GPS: Είναι ένα παγκόσμιο σύστημα εντοπισμού θέσης, το οποίο βασίζεται σε ένα "πλέγμα" 24 δορυφόρων της Γης, στους οποίους υπάρχουν ειδικές συσκευές, οι οποίες ονομάζονται "δέκτες GPS". Οι δέκτες αυτοί παρέχουν ακριβείς πληροφορίες για τη θέση ενός σημείου, το υψόμετρό του, την ταχύτητα και την κατεύθυνση της κίνησης του.

JavaScript: Η Javascript είναι μια γλώσσα σεναρίου (script language) που χρησιμοποιείται για να δημιουργήσουμε διαδραστικές (interactive) ιστοσελίδες. Ο κώδικας JavaScript συνήθως ενσωματώνεται στον HTML κώδικα. Η JavaScript είναι μια interpreted language, δηλαδή εκτελείται χωρίς να έχει προηγηθεί μεταγλώττιση του κώδικα.

JSON: Το JSON (JavaScript Object Notation) είναι ένα ελαφρύ πρότυπο ανταλλαγής δεδομένων. Πρόκειται για να πρότυπο κειμένου το οποίο είναι τελείως ανεξάρτητο από γλώσσες προγραμματισμού. Είναι εύκολο για τους ανθρώπους να το διαβάσουν και γράψουν. Επίσης, είναι εύκολο για τις μηχανές να το αναλύσουν (parse) και να το παράγουν (generate).

JSP: Το ακρωνύμιο σημαίνει JavaServer Pages. Είναι μια τεχνολογία που βοηθά τους προγραμματιστές να δημιουργήσουν δυναμικές ιστοσελίδες. Κυκλοφόρησε το 1999 από την Sun Microsystems, και είναι παρόμοια με την PHP, αλλά χρησιμοποιεί τη γλώσσα προγραμματισμού Java.

NLP: Το ακρωνύμιο σημαίνει Natural language processing. Είναι ένα πεδίο της επιστήμης των υπολογιστών, της τεχνητής νοημοσύνης, και της γλωσσολογίας που ασχολείται με τις αλληλεπιδράσεις μεταξύ των υπολογιστών και της ανθρώπινης γλώσσας. Πολλές προκλήσεις στην NLP περιλαμβάνουν την κατανόηση φυσικής γλώσσας, δηλαδή την δυνατότητα που επιτρέπει στους υπολογιστές να αντλήσουν νόημα από μια φυσική γλώσσα εισαγωγής.

PHP: Η PHP είναι μια γλώσσα προγραμματισμού για τη δημιουργία δυναμικών ιστοσελίδων. Μια σελίδα PHP περνά από επεξεργασία από ένα συμβατό διακομιστή του Παγκόσμιου Ιστού, ώστε να παραχθεί σε πραγματικό χρόνο το τελικό περιεχόμενο, που θα σταλεί στο πρόγραμμα περιήγησης των επισκεπτών σε μορφή κώδικα HTML.

Reddit: Το Reddit είναι ένα ειδησεογραφικό κοινωνικό δίκτυο όπου οι εγγεγραμμένοι χρήστες αναρτούν περιεχόμενο και άλλοι το ψηφίζουν θετικά ή αρνητικά διαμορφώνοντας έτσι τη θέση του στις διάφορες θεματικές σελίδες και στην πρώτη σελίδα.

REST: Το ακρωνύμιο σημαίνει Representational State Transfer και είναι ένα πρότυπο για εξ αποστάσεως κλήσεις και ανταλλαγές δεδομένων. Χρησιμοποιεί τις τέσσερις μεθόδους HTTP GET, POST, PUT και DELETE για να εκτελέσει διάφορες εργασίες. Ένα REST API είναι ένα σύνολο ενεργειών που μπορούν να εκτελεστούν με τα τέσσερα ρήματα, χρησιμοποιώντας μια διεύθυνση URI ως παράμετρο.

Server: Γνωστός στα ελληνικά ως εξυπηρετητής ή διακομιστής είναι υλικό ή / και λογισμικό που αναλαμβάνει την παροχή διάφορων υπηρεσιών, 'εξυπηρετώντας' αιτήσεις άλλων προγραμμάτων, γνωστούς ως πελάτες (clients) που μπορούν να τρέχουν στον ίδιο υπολογιστή ή σε σύνδεση μέσω δικτύου.

Smartphones: Είναι μικροί υπολογιστές που χωρούν στην παλάμη του χεριού σου και σου επιτρέπουν να κάνεις πολλά περισσότερα από μόνο φωνητικές κλήσεις και αποστολή SMS που πρόσφεραν τα απλά κινητά τηλέφωνα. Το βασικό σημείο διαφοροποίησης των Smartphones είναι η ενσωμάτωση πλήρους λειτουργικού συστήματος, μία πλήρης πλατφόρμας την οποία μπορείς να την βελτιώνεις συνεχώς εγκαθιστώντας πληθώρα εφαρμογών (Apps).

SMS: Το ακρωνύμιο σημαίνει Short Message Service και είναι υπηρεσία της κινητής τηλεφωνίας, με την οποία ο χρήστης έχει τη δυνατότητα να αποστείλει ή να παραλάβει σύντομο γραπτό μήνυμα από άλλους χρήστες, στην οθόνη του κινητού του τηλεφώνου. Ένα απλό γραπτό μήνυμα ορίζεται στους 160 αλφαριθμητικούς χαρακτήρες, συμπεριλαμβανομένων και των κενών διαστημάτων.

SOAP: Το ακρωνύμιο σημαίνει Simple Object Access Protocol και είναι ένα ελαφρύ πρωτόκολλο προορισμένο για την ανταλλαγή δομημένων πληροφοριών σε ένα αποκεντρωμένο, διανεμημένο περιβάλλον. Χρησιμοποιεί τεχνολογίες XML για να καθορίσει ένα επεκτάσιμο πλαίσιο παρέχοντας μια δομή μηνυμάτων η οποία μπορεί να ανταλλαχθεί πάνω από ποικίλα δικτυακά πρωτόκολλα.

Web-browser: Ένα λογισμικό που επιτρέπει στον χρήστη του να προβάλλει, και να αλληλεπιδρά με, κείμενα, εικόνες, βίντεο, μουσική, παιχνίδια και άλλες πληροφορίες αναρτημένες σε μια ιστοσελίδα ενός ιστότοπου στον Παγκόσμιο Ιστό ή σε ένα τοπικό δίκτυο. Το κείμενο και οι εικόνες σε μια ιστοσελίδα μπορεί να περιέχουν υπερσυνδέσμους προς άλλες ιστοσελίδες του ίδιου ή διαφορετικού ιστότοπου.

XML: Είναι ένα σύνολο κανόνων για το σχεδιασμό μορφών κειμένου οι οποίες διευκολύνουν τη δόμηση των δεδομένων. Η XML διευκολύνει τον υπολογιστή να παράγει δεδομένα, να διαβάζει δεδομένα και να εξασφαλίζει τη σαφήνεια της δομής των δεδομένων. Είναι επεκτάσιμη, ανεξάρτητη συστήματος υλικού και μπορεί να υποστηρίξει διεθνείς και τοπικές προσαρμογές.

Βιβλιογραφικές Αναφορές

1. What is PodCasting? [Online] Lionhardt Technologies. [Cited: 05 13, 2013.] http://www.lionhardt.ca/webpodStudio/WhatIsPodCasting.aspx?RefID=ASPRedir.
2. What's a blog? [Online] Google. [Cited: 05 13, 2013.] https://www.blogger.com/tour_start.g.
3. GETTING, BRIAN. What Are “Tags” And What Is “Tagging?”. [Online] [Cited: 05 13, 2013.] http://www.practicalecommerce.com/articles/589-What-Are-Tags-And-What-Is-Tagging-.
4. *Τι ακριβώς είναι το Social bookmarking;*. [Online] [Cited: 05 13, 2013.] http://www.prosweb.gr/%CE%AC%CF%81%CE%B8%CF%81%CE%B1/item/103-social-bookmarking.
5. Social networking service. [Online] wikipedia.org. http://en.wikipedia.org/wiki/Social_networking_service.
6. Number of active users at Facebook over the years. [Online] Associated Press. http://news.yahoo.com/number-active-users-facebook-over-230449748.html.
7. YouTube says has 1 billion monthly active users. [Online] Reuters. http://www.reuters.com/article/2013/03/21/us-youtube-users-idUSBRE92K03O20130321.
8. LUNDEN, INGRID. Analyst: Twitter Passed 500M Users In June 2012, 140M Of Them In US; Jakarta ‘Biggest Tweeting’ City. [Online] http://techcrunch.com/2012/07/30/analyst-twitter-passed-500m-users-in-june-2012-140m-of-them-in-us-jakarta-biggest-tweeting-city/.
9. Kessler, Sarah. Foursquare Tops 20 Million Users. [Online] [Cited: 05 13, 2013.] http://mashable.com/2012/04/16/Foursquare-20-million/.
10. STATE OF THE MEDIA: SOCIAL MEDIA REPORT Q3. [Online] The Nielsen Company. [Cited: 05 14, 2013.] http://www.nielsen.com/us/en/reports/2011/social-media-report-q3.html.
11. STATE OF THE MEDIA: THE SOCIAL MEDIA REPORT 2012. [Online] The Nielsen Company. [Cited: 05 13, 2013.] http://www.nielsen.com/us/en/reports/2012/state-of-the-media-the-social-media-report-2012.html.
12. How to Retweet. [Online] wikiHow. [Cited: 05 14, 2013.] http://www.wikihow.com/Retweet.
13. Retweeting another person's Tweet. [Online] Twitter. [Cited: 05 14, 2013.] https://support.twitter.com/articles/20169873-retweeting-another-person-s-tweet.
14. Posting @replies and mentions. [Online] Twitter. [Cited: 05 14, 2013.] https://support.twitter.com/articles/20169871-posting-replies-and-mentions.
15. *Notes from the Underground City of Disinformation:A Conceptual Investigation.* Lee, Natascha A. Karlova and Jin Ha. Seattle : University of Washington Information School.
16. SWAN, LISA. 'War of the Worlds' terrified the nation 70 years ago. [Online] NYDailyNews. [Cited: 05 15, 2013.] http://www.nydailynews.com/entertainment/tv-movies/war-worlds-terrified-nation-70-years-article-1.304998.
17. PIERRE, NATE. Abraham Lincoln Filed a Patent for Facebook in 1845. [Online] [Cited: 05 15, 2013.] http://natestpierre.me/2012/05/08/abraham-lincoln-patent-facebook/.
18. Misinformation and Disinformation. [book auth.] Newton Lee. *Facebook Nation.* s.l. : Springer New York, 2013.
19. FBI looks into hacker attack on Associated Press. [Online] Euronews. [Cited: 05 15, 2013.] http://www.euronews.com/2013/04/24/fbi-looks-into-hacker-attack-on-associated-press/.
20. Eunsoo Seo, Prasant Mohapatra and Tarek Abdelzaher. Identifying Rumors and Their Sources in Social Networks .

21. Vahed Qazvinian, Emily Rosengren, Dragomir R. Radev and Qiaozhu Mei. Rumor has it: Identifying Misinformation in Microblogs .
22. Marcelo Mendoza, Barbara Poblete and Carlos Castillo. Twitter Under Crisis: Can we trust what we RT?
23. Eysenbach, Cynthia Chew and Gunther. Pandemics in the Age of Twitter: Content Analysis of Tweets during the 2009 H1N1 Outbreak.
24. Manuela Krieck, Johannes Dreesman, Lubomir Otrusina and Kerstin Denecke. A New Age of Public Health: Identifying Disease Outbreaks by Analyzing Tweets.
25. Java Features & Benefits. [Online] Oracle. [Cited: 4 20, 2013.] http://www.oracle.com/us/technologies/java/features/index.html.
26. ~okeanos IAAS. [Online] GRNET. [Cited: 04 20, 2013.] https://okeanos.grnet.gr/.
27. John Watson, LLC. Big Huge Thesaurus: Synonyms, antonyms, and rhymes. [Online] [Cited: 04 20, 2013.] http://words.bighugelabs.com/.
28. About WordNet. [Online] Princeton University. [Cited: 04 20, 2013.] http://wordnet.princeton.edu/.
29. OpenCalais. [Online] Thomson Reuters. [Cited: 04 20, 2013.] http://www.opencalais.com/.
30. GeoNames. [Online] [Cited: 04 20, 2013.] http://www.geonames.org/.
31. SNAP: Network datasets: 476 million Twitter tweets. [Online] Stanford University. [Cited: 04 21, 2013.] http://snap.stanford.edu/data/twitter7.html.
32. Using the Twitter Search API. [Online] Twitter. [Cited: 05 21, 2013.] https://dev.twitter.com/docs/using-search.
33. The Streaming APIs. [Online] Twitter. [Cited: 04 21, 2013.] https://dev.twitter.com/docs/streaming-apis.
34. Twitter. FAQs about Retweets. [Online] [Cited: 04 21, 2013.] https://support.twitter.com/articles/77606-faqs-about-retweets.
35. External-Memory Sorting in Java. [Online] [Cited: 04 21, 2013.] http://code.google.com/p/externalsortinginjava/.
36. Bin Laden raid was revealed on Twitter. [Online] BBC News. [Cited: 05 16, 2013.] http://www.bbc.co.uk/news/technology-13257940.
37. How many Twitter followers do they really have? [Online] Guardian News and Media Limited. [Cited: 05 16, 2013.] http://www.guardian.co.uk/technology/2012/aug/26/how-many-twitter-followers-do-they-really-have.
38. BalkaNet - Design and Development of a Multilingual Balkan WordNet. [Online] UNIVERSITY OF PATRAS. [Cited: 05 16, 2013.] http://www.dblab.upatras.gr/balkanet/.
39. EuroNet: Building a multilingual database with wordnets for several European languages. [Online] University of Amsterdam. [Cited: 05 16, 2013.] http://www.illc.uva.nl/EuroWordNet/.
40. DISCO: compute semantic similarity beetween words. [Online] linguatools. [Cited: 05 17, 2013.] http://www.linguatools.de/disco/disco_en.html.
41. Calculate Semantic Similarity. [Online] damir-olejar. [Cited: 05 17, 2013.] http://sourceforge.net/projects/semantics/?source=dlp.

ΠΑΡΑΡΤΗΜΑ: Κώδικας Υλοποίησης

Παρακάτω παραθέτουμε τον κώδικα που γράφτηκε στα πλαίσια της παρούσας εργασίας. Να σημειώσουμε πως τα κομμάτια του Social Sensing Portal σε PHP και Javascript καθώς και το βοηθητικό εργαλείο που δημιουργήσαμε για την ταξινόμηση του συνόλου δεδομένων δεν παρατίθενται στο παρόν παράρτημα.

Social Sensing Portal

Social_Sensing.java

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.Reader;
import java.io.UnsupportedEncodingException;
import java.net.InetAddress;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Properties;

public class Social_Sensing {

	public static void main(String[] args) {

		Tweet_Reader tr;
		try {
			DateFormat dateFormat = new SimpleDateFormat("yyyy.MM.dd-
HH:mm:ss");
			Date date = new Date();
			String build_version = "27";
			String computerName = InetAddress.getLocalHost().getHostName();

			System.out.println("Hello from "+computerName+",
"+dateFormat.format(date)+", version "+build_version+"\n---------------\n");

			// get the properties file
			Properties prop = new Properties();
			prop.load(new FileInputStream(args[0]));

			// get the application parameters
			int social_sensing_mode =  Integer.parseInt( prop.getProperty("social-
sensing-mode") );
			String keywords = prop.getProperty("keywords");
			String outDir = prop.getProperty(computerName+"-outDir");
			int sprHours = Integer.parseInt( prop.getProperty("spreading-hours") );
```

```
                        int sprNumTweets = Integer.parseInt( prop.getProperty("spreading-tweets")
);
                        int filterMode = Integer.parseInt( prop.getProperty("filter-mode") );
                        String extraOptions = prop.getProperty("extra");
                        String outputFilePath = outDir+computerName+"-
"+dateFormat.format(date)+".txt";

                        // overwrite parameters if they are given from the command-line
                        if (args.length > 2) {

                                // 2. output file
                                outputFilePath = args[1];

                                // 3. social sensing mode
                                social_sensing_mode = Integer.parseInt(args[2]);

                                // 4,5,6,7,8
                                keywords = args[3];
                                sprHours = Integer.parseInt(args[4]);
                                sprNumTweets = Integer.parseInt(args[5]);
                                filterMode = Integer.parseInt(args[6]);
                                extraOptions = args[7];

                                // screen output
                                System.out.println("\nNOTE: Parameters set from command
line.\n");
                        }

                        // process the extra options
                        boolean synonyms = false;
                        String[] extras = extraOptions.split(",");
                        for (String extra: extras) {
                                if (extra.equals("syn")) {
                                        synonyms = true;
                                }
                        }

                        // create the current output file
                        // HELP: http://www.roseindia.net/java/beginners/java-write-to-file.shtml
                        FileWriter fstream = new FileWriter(outputFilePath);
                        BufferedWriter out = new BufferedWriter(fstream);

                        // create the Keyword Handler object
                        Keywords_Handler kh = new Keywords_Handler(keywords, synonyms);

                        // write to the output file the details of this scan
                        out.write("Social Sensing build version "+build_version+",
"+computerName+"\n");
                        out.write("social-sensing-mode: " + social_sensing_mode + "\n");
                        out.write("keywords: " + kh.getOriginalKeywords() + "\n");
                        out.write("extended-keywords: " + kh.getExtendedKeywords() + "\n");
                        out.write("spreading-hours: " + sprHours + "\n");
                        out.write("spreading-tweets: " + sprNumTweets + "\n");
                        out.write("filter-mode: " + filterMode + "\n");
                        out.write("extra: " + extraOptions );

                        // to be removed
                        System.out.println("Synonyms: "+synonyms);

                // start processing the files for this pc
```

```
                int inFile = 1;
                String inFilepath = prop.getProperty(computerName+"-file"+inFile);
        while (inFilepath != null)
        {
            System.out.println("--------- NOW PROCESSING "+         inFilepath+" ---------" );
            out.write("\n\n---------- "+inFilepath+" ----------\n");

                        tr = new Tweet_Reader(inFilepath);

                        switch (social_sensing_mode) {
                        case 1:
                                First_Filter ff = new First_Filter(tr, filterMode, out, kh,
sprHours, sprNumTweets);
                                ff.do_filter();
                                break;

                        default:
                                System.out.println("2nd Filter activated");
                                Second_Filter sf = new Second_Filter(tr, filterMode, out, kh,
sprHours, sprNumTweets);
                                sf.do_filter();
                                break;
                        }


                        inFile++;
                        inFilepath = prop.getProperty(computerName+"-file"+inFile);
        }

                System.out.println("done.");

                date = new Date();
                System.out.println(dateFormat.format(date));

                // close the output file
                out.write(dateFormat.format(date)+"\n");
        out.close();

            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }


            /*
             // CODE FOR TESTING REGEX
              // String to be scanned to find the pattern.
                String line = "RT @user1, @SeekingAlpha, @mik @hello: RT @kapoios RT
@niko Best Stock Bargains!! <<< yes! [no!]";
                // line = "RT @ZDNetBlogs: Microsoft .Net RIA Services: Not until 2010 -
http://tinyurl.com/lhvvn7l";

                String pattern = "(?:RT +(?:@(\\w+)[:,]* +)+)+([^\\[]*)";

              //Create a Pattern object
              Pattern r = Pattern.compile(pattern);

              // Now create matcher object.
              Matcher m = r.matcher(line);
```

```
                    if (m.find( )) {

                    System.out.println("Found value: " + m.group(0) );
                    System.out.println("Found value: " + m.group(1) );
                    System.out.println("Found value: " + m.group(2) );
                            String authors[] = m.group(1).split(" +");
                            String author = authors[authors.length-1].replaceAll(":|,|@", "");
                            System.out.println(author);

                    } else { System.out.println("NO MATCH"); }
                   */



        }
}
```

Tweet.java

```
import java.io.ObjectInputStream.GetField;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Tweet {

	private Date mDate;
	private String mDt, mUser, mMsg, mRT = null;



	public Tweet(String dt, String user, String msg) {

		// save the Username, Message and Date-Time as text
		mDt = getTxtAfterTab(dt);
		mUser = getTxtAfterTab(user).replace("http://twitter.com/", "");
		mMsg = getTxtAfterTab(msg);

		// convert the date string to a Date object
		// http://www.tutorialspoint.com/java/java_date_time.htm
		// alternative: http://joda-time.sourceforge.net/
		try {
			mDate = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").parse(mDt);
		} catch (ParseException e) {
			e.printStackTrace();
			printTweet();
		}
	}

	public void printTweet() {
		System.out.println("Date: " + mDt);
		System.out.println("User: " + mUser);
		System.out.println("Msg: " + mMsg);
		System.out.println();
	}

	public String toString() {
		// return "\n\nDate: " + mDt + "\nUser: " + mUser + "\nMsg: " + mMsg +"\n";
		return "\n\n" + mDt + ", " + mUser + ", " + mMsg + "\n";

	}

	public String getDt() {
		return mDt;
	}

	public String getUser() {
		return mUser;
	}

	public String getMsg() {
		return mMsg;
	}
```

```
	public Date getDate() {
		return mDate;
	}

	public String getRT() {
		return mRT;
	}

	public void setRT(String mRT) {
		this.mRT = new String(mRT);
	}

	public void setDt(String mDt) {
		this.mDt = mDt;
	}

	private String getTxtAfterTab(String txt) {
		String parts[] = txt.split("\t");
		if (parts.length == 1) {
			return txt;
		} else
			return parts[1];
	}

	public static long secondsBetweenTweets(Tweet oldT, Tweet newT) {
		long diff = newT.getDate().getTime() - oldT.getDate().getTime();
		return diff / 1000;
	}

}
```

Tweet_Reader.java

```
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Scanner;

public class Tweet_Reader {

	private FileInputStream fis;
	private BufferedReader reader = null;
	private Scanner scanner;
	private int mTotal;
	private int mCurrentTweet;

	public Tweet_Reader(String fileName) throws IOException {

		// Read the data file
		// http://javarevisited.blogspot.com/2012/07/read-file-line-by-line-java-example-
scanner.html#ixzz2Hftxy85J
		fis = new FileInputStream(fileName);
		scanner = new Scanner(fis);
		reader = new BufferedReader(new InputStreamReader(fis));

		String firstLine = reader.readLine();
		if ( firstLine.contains(":") )
		{
			String[] parts = firstLine.split(":");
			mTotal = Integer.parseInt( parts[1] );
			mCurrentTweet = 0;
		}
		else
			throw new RuntimeException("Could not find total number of messages.
File format is invalid.");

	}

	public boolean hasNextTweet() {

		if (mCurrentTweet < mTotal)
			return true;
		else
			return false;

		// return scanner.hasNext();
	}

	public int getCurrentTweet() {
		return mCurrentTweet;
	}

	public int getTotalTweet() {
		return mTotal;
	}
```

```
	public Tweet nextTweet() throws IOException {

		String nxtLine,dt,user,msg;

		// skip all the lines that do not start with T prefixe
		nxtLine = reader.readLine();
		while ( (nxtLine != null) && (!nxtLine.startsWith("T\t")) )
			nxtLine = reader.readLine();

		if (nxtLine == null) return null; // detect EOF

		// once a non-blank line found read the next 2 lines
		dt = nxtLine;
		user = reader.readLine().toLowerCase();
		msg = reader.readLine();

		mCurrentTweet++;
		Tweet t = new Tweet(dt, user, msg);

		return t;

		/*
		// Detecting anomalies in the input file format
		String s1,s2,s3,s4;
		while (scanner.hasNextLine())
		{
			s1 = scanner.nextLine();
			s2 = scanner.nextLine();
			s3 = scanner.nextLine();
			s4 = scanner.nextLine();
			if (!s4.equals(""))
			{
				System.out.println("s1: "+ s1 + "\ns2: " + s2 + "\ns3: " + s3 + "\ns4: "
+ s4 + "\n" );
				s1 = scanner.nextLine();
				s2 = scanner.nextLine();
				s3 = scanner.nextLine();
				s4 = scanner.nextLine();
				System.out.println("s1: "+ s1 + "\ns2: " + s2 + "\ns3: " + s3 + "\ns4: "
+ s4 + "\n" );

				System.exit(0);
			}
		}

		return null;
		*/
	}

	public void closeReader() {
		scanner.close();
	}

}
```

Keywords_Handler.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;
import java.util.regex.Matcher;
import java.util.regex.Pattern;


public class Keywords_Handler {

	private Pattern keywordPattern;
	private String originalKeywords = "", extendedKeywords = "";

	public Keywords_Handler(String _keywords, boolean synonyms) {

		originalKeywords = _keywords;

		if (_keywords.startsWith("!!")) {
			String regex = _keywords.substring(2);
			System.out.println("Custom regex activated: "+regex);
			keywordPattern = Pattern.compile(regex);
		}
		else {
			String finalPattern = "";
			String andParts[] = _keywords.split("[|]");
			for(String andPart: andParts) {
				if (synonyms) andPart += getSynonymsList(andPart); // add
synonyms, if enabled
				String keywords[] = andPart.split(",");
				String pattern = "";
				for (int i=0; i<keywords.length; i++)
					pattern += "[^a-zA-Z0-
9]+"+keywords[i].toLowerCase()+"[^a-zA-Z0-9]+|";
				pattern = pattern.substring(0, pattern.length()-1); // remove the last
"|"
				finalPattern += "(?=.*?("+pattern+"))";
			}

			// create and compile the regex needed for keyword filtering
			finalPattern = "^"+finalPattern+".*$";
			keywordPattern = Pattern.compile(finalPattern);
			// System.out.println("Regex activated: "+finalPattern);

		}
	}

	public boolean keyword_filter(String msg) {

		// Now create matcher object.
		Matcher m = keywordPattern.matcher(msg.toLowerCase());
		return m.find();
```

```
	}

	public String getOriginalKeywords() {
		return originalKeywords;
	}

	public String getExtendedKeywords() {
		return extendedKeywords;
	}

	/**
	  * Returns a comma-separated String that contains any synonyms found for the given
keywords
	  *
	  * @param _keywords A comma-separated String that contains the keywords that need to
be queried
	  * @return this A comma-operated String with all the EXTRA synonyms keywords found
	  */
	private String getSynonymsList(String _keywords) {
		String finalKeywords = "";
		String keywords[] = _keywords.split(",");
		for(String keyword: keywords) {
			finalKeywords += BigHugeThesaurus(keyword);
		}
		extendedKeywords += finalKeywords;
		return finalKeywords;
	}

	/**
	  * Queries the BigHugThesaurus for a word and return the extra keywords
	  *
	  * @param word The word to query
	  * @return this A comma-operated String with all the EXTRA synonyms found
	  */
	private String BigHugeThesaurus(String word) {

		/* example server response for "depressed":
			adjective|syn|blue
			adjective|syn|dispirited
			adjective|syn|downcast
			adjective|syn|downhearted
		 */

		// for holding the final keywords
		String finalWords = "";

		try {
			InputStream input = new
URL("http://words.bighugelabs.com/api/2/a268c68441ee3be284ac3069b122791a/"+URLEncoder.en
code(word, "UTF-8")+"/").openStream();
			BufferedReader br = new BufferedReader( new InputStreamReader(input)
);

			// only add the words that are synonyms
			String strLine;
			while((strLine = br.readLine()) != null) {
				String parts[] = strLine.split("[|]");
				if (parts[1].equals("syn")) {
					finalWords += ","+parts[2];
				}
```

```
                        }
                        br.close();

                        return finalWords;

                } catch (MalformedURLException e1) {
                        e1.printStackTrace();
                } catch (UnsupportedEncodingException e1) {
                        e1.printStackTrace();
                } catch (IOException e1) {
                        // if this exception occurs, it just means that the requested word has no
synonyms => do not print anything
                }

                return finalWords;

        }

}
```

First_Filter.java

```
import java.io.BufferedWriter;
import java.io.IOException;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.Queue;
import java.util.Vector;
import java.util.regex.Matcher;
import java.util.regex.Pattern;


public class First_Filter {

	private Tweet_Reader tr;
	private int mode,sprHours,sprNumTweets;
	private BufferedWriter output;
	private Keywords_Handler kh;

	// mode
	// 1. Filter with keyword first, then RT filtering
	// 2. RT filtering, then keyword filtering

	public First_Filter(Tweet_Reader _tr, int _mode, BufferedWriter _out, Keywords_Handler
_kh, int _sprHours, int _sprNumTweets) {
		tr = _tr;
		mode = _mode;
		output = _out;
		kh = _kh;
		sprHours = _sprHours;
		sprNumTweets = _sprNumTweets;
	}


	public void do_filter() throws IOException {

		// HELP
		// Java RegEx: http://www.tutorialspoint.com/java/java_regular_expressions.htm
		// Twitter RT RegEx: http://stackoverflow.com/questions/558105/string-separation-
in-required-format-pythonic-way-with-or-w-o-regex

		// queue for holding the first appearance of each tweet
	Queue<Tweet> qFirstAppear = new LinkedList<Tweet>();

	// hashtable for holding the number each retweet has been retweeted
	Hashtable<String,Integer> CountRT = new Hashtable<String,Integer>();
	Hashtable<String,Vector<String>> htAuthors = new Hashtable<String,Vector<String>>();

		// Create a Pattern object
		String pattern = "(?:RT +(?:@(\\w+)[:,]* +)+)+(.*)";
		Pattern r = Pattern.compile(pattern);

		Tweet oldT, newT = tr.nextTweet();
		oldT = new Tweet("2009-06-01 00:00:00", "nothing", "nothing");
		while (newT != null) {
```

```
                        // only process tweets if found in chronological order
                        if (newT.getDate().after(oldT.getDate()) ||
newT.getDate().equals(oldT.getDate())) {

                                if (newT.getDate().getDay() != oldT.getDate().getDay())
                                        System.out.println("---------- "+newT.getDate()+" ----------
----------");

                                // Now create matcher object.
                                Matcher m = r.matcher(newT.getMsg());

                                // if RegEx match is successful -> a RT is found
                                if (m.find()) {

                                        String msg = m.group(2); // holds the RT body
                                        String author = m.group(1); // holds the author of the RT

                                        newT.setRT(msg); // save the RT msg for later use

                                        // UPDATE THE RT COUNTS
                                        if (!CountRT.containsKey(msg)) { // if this is the 1st time a
RT is found
                                                // Count RT
                                                int i = 1;
                                                CountRT.put(msg, i);

                                                // Authors vector
                                                Vector<String> v = new Vector<String>();
                                                v.add(author + " | " + newT.getUser() + ": " +
newT.getMsg());
                                                htAuthors.put(msg, v);

                                        }
                                        else { // not the first time a RT is found
                                                CountRT.put(msg, CountRT.get(msg)+1 );
                                                //int i = CountRT.get(msg);
                                                //if (i > 40)
                                                //      System.out.println(i+":
"+msg+"\n"+newT.getUser()+": "+newT.getMsg());

                                                // Authors vector
                                                Vector<String> v = htAuthors.get(msg);
                                                v.add(author + " | " + newT.getUser() + ": " +
newT.getMsg());
                                                htAuthors.put(msg, v );
                                        }
                                }

                                // REMOVE THE tweets THAT PASSED THE TIMEFRAME
                                qFirstAppear.add(newT);
                                Tweet top = qFirstAppear.peek();
                                while ( Tweet.secondsBetweenTweets(top, newT) >
(sprHours*60*60) )
                                {
                                        top = qFirstAppear.poll(); // remove the top (oldest) tweet
                                        // System.out.println(top.getMsg());
                                        if (top.getRT() != null)
                                        {
                                                if (CountRT.containsKey(top.getRT()))
                                                {
```

```
                                                        int count = CountRT.get(top.getRT());
                                                        if (count > sprNumTweets)
                                                                //TODO In mode 1, no keyword
filtering is executed in the RT body!
                                                                if ( (mode == 1) || ((mode == 2) &&
kh.keyword_filter(top.getRT())) )
                                                                {
                                                                        String temp = "count:
"+count+", date: "+top.getDt()+": "+top.getRT();
                                                                        System.out.println(temp);
                                                                        output.write(temp+"\n");

                                                                        // Authors vector
                                                                        Vector<String> v =
htAuthors.get(top.getRT());
                                                                        Iterator<String> it =
v.iterator();
                                                                String authors;
                                                                        for (; it.hasNext();) {
                                                                    authors = it.next();

        System.out.println("- "+authors);
                                                                    output.write("- "+authors+"\n");
                                                                }

                                                                    }
                                                                CountRT.remove(top.getRT());
                                                                }
                                                        }
                                                        top = qFirstAppear.peek();
                                                }

                                                        oldT = newT;
                                        }
                                        else
                                                        System.out.println("old: " + oldT.getDt() + " ,new: "+ newT.getDt() +
" ,id: " + tr.getCurrentTweet());

                                        // fetch next tweet
                                        do {
                                                        newT = tr.nextTweet();
                                        } while ( (newT != null) && (mode == 1) &&
(kh.keyword_filter(newT.getMsg()) == false) );
```

Second_Filter.java

```
import java.io.BufferedWriter;
import java.io.IOException;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.Queue;
import java.util.Set;
import java.util.TreeSet;
import java.util.Vector;
import java.util.regex.Matcher;
import java.util.regex.Pattern;


public class Second_Filter {

	private Tweet_Reader tr;
	private int filterMode,sprHours,sprNumTweets;
	private BufferedWriter output;
	private Keywords_Handler kh;

	public Second_Filter(Tweet_Reader _tr, int _filterMode, BufferedWriter _out,
Keywords_Handler _kh, int _sprHours, int _sprNumTweets) {
		tr = _tr;
		output = _out;
		kh = _kh;
		sprHours = _sprHours;
		sprNumTweets = _sprNumTweets;
		filterMode = _filterMode;
	}


	public void do_filter() throws IOException {

		// HELP
		// Java RegEx: http://www.tutorialspoint.com/java/java_regular_expressions.htm
		// Twitter RT RegEx: http://stackoverflow.com/questions/558105/string-separation-
in-required-format-pythonic-way-with-or-w-o-regex

		// queue for holding the first appearance of each Original Tweet
	Queue<Tweet> qFirstAppear = new LinkedList<Tweet>();

		// set for holding the users of Original Tweets
	Set<String> watchUsers = new TreeSet<String>();

	// hashtable for holding the replies for each Original Tweet
	Hashtable<String,Vector<String>> htReplies = new Hashtable<String,Vector<String>>();

	// regex for finding the @reply
	Pattern replyPattern = Pattern.compile("^@([A-Za-z0-9_]+)");

		Tweet oldT, newT = tr.nextTweet();
		oldT = new Tweet("2009-06-01 00:00:00", "nothing", "nothing");
		while (newT != null) {

			// only process tweets if found in chronological order
			if (newT.getDate().after(oldT.getDate()) ||
newT.getDate().equals(oldT.getDate())) {
```

```
                        if (newT.getDate().getDay() != oldT.getDate().getDay())
                                System.out.println("---------- "+newT.getDate()+" ----------
-----");

                        String msg = newT.getMsg();
                        if (!msg.startsWith("@") && !msg.contains("RT")) { // possible
Original Tweet
                                if (kh.keyword_filter(msg)) { // tweet contains some of the
keywords => Original Tweet
                                        qFirstAppear.add(newT);
                                        watchUsers.add(newT.getUser());
                                }
                        }
                        else if (!msg.contains("RT")) { // possible reply for an Original Tweet

                                /*
                                Iterator<Tweet> originalTweet = qFirstAppear.iterator();
                                boolean found = false;
                                while ((found == false) && (originalTweet.hasNext())){ //
loop through all the Original Tweets
                                        if (msg.startsWith("@"+temp.getUser()+" ")) {
                                */
                                Matcher m = replyPattern.matcher(msg);
                                if (m.find()) {

                                        String author = m.group(1); // holds to who the reply
is about
                                        if (watchUsers.contains(author)) {

                                                // UPDATE THE replies COUNTS
                                                // if
(!htReplies.containsKey(temp.getUser())) { // if this is the 1st time an Original Tweet has a reply
                                                if (!htReplies.containsKey(author)) { // if this
is the 1st time an Original Tweet has a reply
                                                        Vector<String> v = new
Vector<String>();

                                                        v.add(newT.getUser()+": "+msg);
                                                        // htReplies.put(temp.getUser(), v);
                                                        htReplies.put(author, v);
                                                }
                                                else { // not the first time an Original Tweet
is found
                                                        // Vector<String> v=
htReplies.get(temp.getUser());
                                                        Vector<String> v=
htReplies.get(author);

                                                        v.add(newT.getUser()+": "+msg);
                                                        htReplies.put(author, v );
                                                        // htReplies.put(temp.getUser(), v );
                                                        // System.out.println("--=--
"+v.size());
                                                }
                                                //found = true;
                                                //break; // stop the iteration through the
Original Tweets
                                        }
                                }
                        }
```

```
                                // REMOVE THE tweets THAT PASSED THE TIMEFRAME
                                Tweet top = qFirstAppear.peek();
                                while ( (top != null) && (Tweet.secondsBetweenTweets(top, newT)
> (sprHours*60*60)) )
                                {
                                        top = qFirstAppear.poll(); // remove the top (oldest) Original
Tweet
                                        watchUsers.remove(top.getUser());
                                        // System.out.println("--=-- "+top.getMsg());
                                        if (htReplies.containsKey(top.getUser()))
                                        {
                                                Vector<String> v = htReplies.get(top.getUser());
                                                htReplies.remove(top.getUser());
                                                if (v.size() > sprNumTweets) {
                                                        String temp = "count: "+v.size()+", date:
"+top.getDt()+"newT: "+newT.getDt()+": "+top.getUser()+": "+top.getMsg();
                                                        System.out.println(temp);
                                                        output.write(temp+"\n");

                                                        if (filterMode == 1) {
                                                                // print all the replies
                                                                Iterator<String> it = v.iterator();
                                                        String reply;
                                                                for (; it.hasNext();) {
                                                        reply = it.next();
                                                                        System.out.println("-
"+reply);
                                                        output.write("- "+reply+"\n");
                                                        }
                                                        }

                                                // print blank line after the replies
                                                System.out.println("");
                                        output.write("\n");
                                                }
                                        }
                                        top = qFirstAppear.peek();
                                }

                                        oldT = newT;
                        }
                        else
                                System.out.println("old: " + oldT.getDt() + " ,new: "+ newT.getDt() +
" ,id: " + tr.getCurrentTweet());

                        // fetch next tweet
                        newT = tr.nextTweet();
                }
        }



}
```

Social Sensing Rumors

Social_Sensing_Rumors.java

```
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.net.InetAddress;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.Properties;
import java.util.Vector;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import GenericTree.GenericTree;


public class Social_Sensing_Rumors {

	private static FileWriter fstream;
	private static BufferedWriter out;

	/** Main function for parsing configuration from command line or config file.
	 *
	 * @param The config file. You can parse the required parameters from command line too.
	 */
	public static void main(String[] args) {

		Tweet_Reader tr;
		try {
			DateFormat dateFormat = new SimpleDateFormat("yyyy.MM.dd-
HH:mm:ss");
			Date date = new Date();
			String build_version = "1";
			String computerName = InetAddress.getLocalHost().getHostName();


			// get the properties file
			Properties prop = new Properties();
			prop.load(new FileInputStream(args[0]));

			// get the application parameters
			int social_sensing_mode =  Integer.parseInt( prop.getProperty("social-
sensing-rumors-mode") );
			String keywords = prop.getProperty("keywords");
			String outDir = prop.getProperty(computerName+"-outDir");
			String extraOptions = prop.getProperty("extra");
			String outputFilePath = outDir+computerName+"-
"+dateFormat.format(date)+".txt";

			// overwrite parameters if they are given from the command-line
```

```
                        if (args.length > 2) {

                                // 2. output file
                                outputFilePath = args[1];

                                // 3. social sensing mode
                                social_sensing_mode = Integer.parseInt(args[2]);

                                // 4
                                keywords = args[3];
                                extraOptions = args[4];

                                // screen output
                                System.out.println("\nNOTE: Parameters set from command
line.\n");
                        }


                        // create the current output file
                        // HELP: http://www.roseindia.net/java/beginners/java-write-to-file.shtml
                        fstream = new FileWriter(outputFilePath);
                        out = new BufferedWriter(fstream);

                        // create the Keyword Handler object
                        Keywords_Handler kh = new Keywords_Handler(keywords, false);

                        // write to the output file the details of this scan
                        o(3, "Social Sensing Rumors, "+computerName+",
"+dateFormat.format(date)+", build version "+build_version);
                        o(3, "social-sensing-rumors-mode: " + social_sensing_mode);
                        o(3, "keywords: " + kh.getOriginalKeywords());
                        o(3, "extra: " + extraOptions );
                        o(3, "");

                // start processing the files for this pc
                        int inFile = 1;
                        String inFilepath = prop.getProperty(computerName+"-file"+inFile);
                while (inFilepath != null)
                {
                    o(3, "--------- PROCESSING "+inFilepath+" ---------" );

                                tr = new Tweet_Reader(inFilepath);

                                // Step 1+2: Filter the tweets and create RT-trees
                                Trees_Creator tc = new Trees_Creator(tr, kh);
                                Hashtable<String,GenericTree<Tweet>> trees = tc.create_trees();

                                o(3, "");
                                o(3, "-----------------------------------------------------------------");
                                o(3, "Step 1+2: Filter the tweets and create RT-trees");
                                o(3, "-----------------------------------------------------------------");
                                o(3, Trees_Creator.stringTreesWithDepth(trees));

                                // ------------------------------------------------------------------

                                // Step 3: Find semantically related tweets
                                Semantic_Scan ss = new Semantic_Scan(trees);
                                Vector<Vector<String>> pairs = ss.semantic_scan();

                                o(3, "");
```

```
                                o(3, "-----------------------------------------------------------------");
                                o(3, "Step 3: Find semantically related tweets");
                                o(3, "-----------------------------------------------------------------");
                                o(3, Semantic_Scan.stringPairs(pairs, trees));

                                // -----------------------------------------------------------------

                                // Step 4: Find out and print any relations between tweet-roots

                                o(3, "");
                                o(3, "-----------------------------------------------------------------");
                                o(3, "Step 4: Find out and print any relations between tweet-roots");
                                o(3, "-----------------------------------------------------------------");

                                Independence_Scan is = new Independence_Scan(trees, pairs,
out);
                                is.independence_scan();



                                inFile++;
                                inFilepath = prop.getProperty(computerName+"-file"+inFile);
                        }

                                System.out.println("done.");

                                date = new Date();
                                System.out.println(dateFormat.format(date));

                                // close the output file
                                out.write(dateFormat.format(date)+"\n");
                        out.close();

                            } catch (FileNotFoundException e) {
                                e.printStackTrace();
                            } catch (IOException e) {
                                e.printStackTrace();
                            }



        }

        /**     Static function for printing on Screen, Output file or both
         *
         * @param mode          1=Screen, 2=OutputFile, 3=Both
         * @param toPrint       String to print
         */
        public static void o(int mode, String toPrint) {

                try {

                        switch (mode) {
                        case 1:
                                System.out.println(toPrint);
                                break;

                        case 2:

                                out.write(toPrint+"\n");
```

```
                    break;

                case 3:
                    System.out.println(toPrint);
                    out.write(toPrint+"\n");
                    break;
                }

            } catch (IOException e) {
                e.printStackTrace();
            }
        }

}
```

Trees_Creator.java

```
import java.io.BufferedWriter;
import java.io.IOException;
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.Queue;
import java.util.Vector;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import mx.bigdata.jcalais.CalaisClient;
import mx.bigdata.jcalais.CalaisObject;
import mx.bigdata.jcalais.CalaisResponse;
import mx.bigdata.jcalais.rest.CalaisRestClient;

import org.geonames.Toponym;
import org.geonames.ToponymSearchCriteria;
import org.geonames.ToponymSearchResult;
import org.geonames.WebService;

import GenericTree.GenericTree;
import GenericTree.GenericTreeNode;

/** Responsible for getting a Tweet_Reader object and returning a list of Re-Tweet Spreading trees
 *
 * @author Mike Yerou
 */
public class Trees_Creator {

	private Tweet_Reader tr;
	private Keywords_Handler kh;

	private Pattern patternRT = Pattern.compile("(?:(?:R/T|RT|via)[:,]* +(?:@(?:\\w+)[:,]*
+)+)+(.*)");
	private Pattern patternRTauthors = Pattern.compile("(?:(?:R/T|RT|via)[:,]* +(?:@(\\w+)[:,]*
+)+)");



	/** Constructor for getting the necessary objects for creating the trees
	 *
	 * @param _tr    Tweet reader object responsible for parsing the tweets
	 * @param _out BufferedWriter object for writing to the output file
	 * @param _kh  Object for making the keyword filteirng
	 */
	public Trees_Creator(Tweet_Reader _tr, Keywords_Handler _kh) {
		tr = _tr;
		kh = _kh;
	}

	/** Print a hashtable that contains many trees
	 *
	 * @param trees		The hashtable containing the trees
	 */
	public static void printTrees (Hashtable<String,GenericTree<Tweet>> trees) {
```

```
		Enumeration<String> e = trees.keys();
	   while(e.hasMoreElements()) {
		String key = e.nextElement();
		GenericTree<Tweet> tree = trees.get(key);
		System.out.println(tree.toStringWithDepth());
		}
	}

	/**	Create a String containing all the trees in a human readable form
	 *
	 * @param trees		the hashtable containing the trees
	 * @return	a String containing all the trees in a human-readable form
	 */
	public static String stringTreesWithDepth(Hashtable<String,GenericTree<Tweet>> trees) {
		String res = "";
		Enumeration<String> e = trees.keys();
		int count = 0;
	   while(e.hasMoreElements()) {
		String key = e.nextElement();
		GenericTree<Tweet> tree = trees.get(key);
			// res += "\n"+Trees_Creator.auxiliaryStringWithDepth(tree.getRoot(),"","");
			res += "\n- Tree "+(count++)+":
"+tree.getRoot().getData().getMsg()+"\n"+Trees_Creator.auxiliaryStringWithDepth(tree.getRoot(),"","
");
	   }
	   return res;
	}


	/** Find the RT part of a twitter message
	 *
	 * @param msg The twitter message
	 * @return The RT part of message
	 */
	private String getRT (String msg) {

		// Create matcher object.
		Matcher m = patternRT.matcher(msg);

		// if RegEx match is successful -> a RT is found
		if (m.find())
			return m.group(1);
		else
			return null;
	}

	/** Returns the authors of a RT, in a vector
	 *
	 * @param msg The twitter message
	 * @return The authors of a retweet.
	 */
	private Vector<String> getRTauthors (String msg) {

		// the object to be returned
      Vector<String> sk = new Vector<String>();

		// Create matcher object.
		Matcher m = patternRTauthors.matcher(msg);

		// if RegEx match is successful -> a new RT author found
```

```
                while (m.find())
                        sk.add(m.group(1).toLowerCase());

                if (!sk.isEmpty())
                        return sk;
                else
                        return null;
        }

        /** Recursive function to find the node of a User in a RT-spreading tree
         *
         * @param currentNode The node that the search will begin
         * @param UserToFind  The user you are looking for
         * @return The node that the has a tweet from that user
         */
   public static GenericTreeNode<Tweet> auxiliaryFind(GenericTreeNode<Tweet> currentNode,
String UserToFind) {
      GenericTreeNode<Tweet> returnNode = null;
      int i = 0;

      if (currentNode.getData().getUser().equals(UserToFind)) {
         returnNode = currentNode;
      }

      else if(currentNode.hasChildren()) {
         i = 0;
         while(returnNode == null && i < currentNode.getNumberOfChildren()) {
            returnNode = auxiliaryFind(currentNode.getChildAt(i), UserToFind);
            i++;
         }
      }

      return returnNode;
   }

   /**     Recursive function to find the distance of a User in a RT-spreading tree
    *
    * @param currentNode        the node to start the search from
    * @param UserToFind         the use we are looking for
    * @param depth                      the starting value of the distance
    * @return     the distance from currentNode to UserToFind
    */
   public static int auxiliaryFindWithDepth(GenericTreeNode<Tweet> currentNode, String
UserToFind, int depth) {
      int resultDepth = -1;

      if (currentNode.getData().getUser().equals(UserToFind)) {
          resultDepth = depth;
      }
      else if(currentNode.hasChildren()) {
         int i = 0;
         while(resultDepth == -1 && i < currentNode.getNumberOfChildren()) {
          resultDepth = auxiliaryFindWithDepth(currentNode.getChildAt(i), UserToFind, depth+1);
            i++;
         }
      }

      return resultDepth;
   }
```

```
    /**     Recursive function to print a RT-tree in a human-readable way
     *
     * @param currentNode       the node to start the search from
     * @param currentString     the starting value of the distance
     * @param currentTab        the starting value of the tab distance
     * @return      a string containing the human-readable form of the tree
     */
    public static String auxiliaryStringWithDepth(GenericTreeNode<Tweet> currentNode, String
currentString, String currentTab) {

            //currentString += currentTab + currentNode.getData().getUser() + ": " +
currentNode.getData().getDt() + ": " + currentNode.getData().getMsg() + "\n";
            currentString += currentTab + currentNode.getData().getUser() + ": " +
currentNode.getData().getDt() + "\n";

        if (!currentNode.hasChildren()) {
            return currentString;
        }
        else {
           int i = 0;
           while(i < currentNode.getNumberOfChildren()) {
            currentString = auxiliaryStringWithDepth(currentNode.getChildAt(i), currentString,
currentTab+"\t");
               i++;
           }
           return currentString;
        }

    }

            /** Create all the RT-spreading trees
             *
             * @return A hashtable containing all the RT-spreading trees
             */
            public Hashtable<String,GenericTree<Tweet>> create_trees() {

        // hashtable for holding the number each retweet has been retweeted
        Hashtable<String,GenericTree<Tweet>> trees = new
Hashtable<String,GenericTree<Tweet>>();

        // Queue for keeping a buffer of tweets in order to find the original user of a RT
        Queue<Tweet> qBuffer = new LinkedList<Tweet>();


                        try {

                                Tweet oldT, newT = tr.nextTweet();
                                oldT = new Tweet("2009-06-01 00:00:00", "nothing", "nothing");
                                while (newT != null) {

                                        // only process tweets if found in chronological order
                                        if (newT.getDate().after(oldT.getDate()) ||
newT.getDate().equals(oldT.getDate())) {

                                                qBuffer.add(newT);

                                                if (newT.getDate().getDay() != oldT.getDate().getDay())
                                                        System.out.println("---------- "+newT.getDate()+" -
----------");
```

```
                                String RT = getRT(newT.getMsg());
                                Vector<String> authors = getRTauthors(newT.getMsg());

                                // Check if a RT is found
                                if ((RT != null) && (authors != null)) {
                                        newT.setRT(RT);

                                        if (!trees.containsKey(RT)) { // this is the 1st time a
RT is found
                                                GenericTree<Tweet> tree = new
GenericTree<Tweet>();

                                                // search if the original twitter of a RT is in
the buffer
                                                GenericTreeNode<Tweet> root = null;
                                                Iterator<Tweet> it = qBuffer.iterator();
                                                while (it.hasNext() && (root == null)) {
                                                        Tweet t = it.next();
                                                        if (RT.equals(t.getMsg()))
                                                                root = new
GenericTreeNode<Tweet>( t );
                                                }
                                                if (root == null)
                                                        root = new
GenericTreeNode<Tweet>( new Tweet(newT.getDt(), authors.lastElement(), RT) );
                                                GenericTreeNode<Tweet> child = new
GenericTreeNode<Tweet>( newT );

                                                root.addChild(child);
                                                tree.setRoot(root);

                                                trees.put(RT, tree);
                                        }
                                        else {  // not the first time a RT is found
                                                GenericTreeNode<Tweet> child = new
GenericTreeNode<Tweet>( newT );
                                                GenericTree<Tweet> tree = trees.get(RT);

                                                // TODO Ti na kanw an den vriskw ton
ameso patera?!
                                                // search the authors list in the tree, to find
the father
                                                boolean found = false;
                                                Iterator<String> it = authors.iterator();
                                                for (;it.hasNext();) {
                                                        String curAuthor = it.next();
                                                        GenericTreeNode<Tweet> father =
auxiliaryFind(tree.getRoot(), curAuthor );
                                                        if (father != null) { // father found!
                                                                father.addChild(child);
                                                                trees.put(RT, tree);
                                                                found = true;
                                                                break;
                                                        }
                                                        else { // father not found -> create
the father and add the child
                                                                GenericTreeNode<Tweet>
newFather = new GenericTreeNode<Tweet>( new Tweet(newT.getDt(), curAuthor, newT.getRT()) );
                                                                newFather.addChild(child);
                                                                child = newFather;
```

```
                                                        }
                                                }

                                                // in case no father is found from the
authors list, add it to the root
                                                if (!found) {
                                                        tree.getRoot().addChild(child);
                                                        System.out.println("No Father:
"+newT.toString().replaceAll("\n", ""));
                                                }

                                        }

                                }

                                // Remove Tweets from the buffer when a limit is exceeded
                                if (qBuffer.size() >= 500) {
                                        qBuffer.poll();
                                }

                                // set the current Tweet as the last processed tweet
                                oldT = newT;
                        }
                        else
                                System.out.println("old: " + oldT.getDt() + " ,new: "+
newT.getDt() + " ,id: " + tr.getCurrentTweet());

                        // fetch next tweet
                        do {
                                newT = tr.nextTweet();
                        } while ( (newT != null) && (kh.keyword_filter(newT.getMsg()) ==
false) );

                }
        } catch (IOException e) {
                e.printStackTrace();
        }

        return trees;


    }

}
```

Semantic_Scan.java

```
import java.io.IOException;
import java.util.Enumeration;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Set;
import java.util.Vector;

import mx.bigdata.jcalais.CalaisClient;
import mx.bigdata.jcalais.CalaisConfig;
import mx.bigdata.jcalais.CalaisObject;
import mx.bigdata.jcalais.CalaisResponse;
import mx.bigdata.jcalais.rest.CalaisRestClient;

import org.geonames.ToponymSearchCriteria;
import org.geonames.ToponymSearchResult;
import org.geonames.WebService;

import GenericTree.GenericTree;


/**	Responsible for getting all the trees and grouping them if the roots are semantically related
 *
 * @author Mike Yerou
 */
public class Semantic_Scan {

	Hashtable<String,GenericTree<Tweet>> trees;
	Keywords_Handler kh;

	/** Constructor for getting the necessary objects for semantically analyzing the tweets
	 *
	 * @param _trees		A hashtable of trees containing all the trees
	 */
	public Semantic_Scan(Hashtable<String,GenericTree<Tweet>> _trees) {
		trees = _trees;

		// initialize the keyword handler to check for co-occurence
		kh = new Keywords_Handler("died", false);
	}


	/**	Responsible for semantically analyze the RT trees and finding which roots are
semantically related
	 *
	 *	@return		A vector of vectors containing the semantically related RT-trees
	 */
	public Vector<Vector<String>> semantic_scan() {

		// a set to save which root-tweets are available for matching
		Set<String> avTweets = new HashSet<String>();

		// a vector to keep all the tweet-roots
		Vector<String> roots = new Vector<String>();

	   // iterate through all the trees in the hashtable and get the tweet-roots
		Enumeration<String> e = trees.keys();
```

```
            while(e.hasMoreElements()) {
                String key = e.nextElement();
                GenericTree<Tweet> tree = trees.get(key);
                roots.add(tree.getRoot().getData().getMsg());
                }
            avTweets.addAll(roots);

            // check all the tweet-roots for semantical relation
            Vector<Vector<String>> pairs = new Vector<Vector<String>>(); // vector to hold the
tweet-root pairs
            for (String orginal_tweet: roots) {

                // skip the tweet-root if it is not available for pairing
                if (avTweets.contains(orginal_tweet))
                                avTweets.remove(orginal_tweet);
                else
                        continue;

                // for each available tweet-root, iterate through the rest of available tweet-roots for
matching
                Vector<String> v = new Vector<String>();
                v.add(orginal_tweet); // a new vector to hold the possible pair
                for (String b_tweet: avTweets) {
                        if (areTheTweetsRelated(orginal_tweet, b_tweet))
                                v.add(b_tweet);
                }

                // if no pair is found -> add the original_tweet back to the available ones
                if (v.size() == 1) {
                        avTweets.add(orginal_tweet);
                }
                else { // else remove all the matches from the available list
                        avTweets.removeAll(v);
                        pairs.add(v);
                }

            }

            return pairs;

        }


        /**     Check if the b-tweet is related to original-tweet
         *
         *      @param      a       The Original Tweet
         *      @param      b       The b-tweet
         *      @return     Check if 2 tweets are related
         */
        private boolean areTheTweetsRelated(String a, String b) {

                // TODO bug! the keyword_filter cannot detect a keyword if the keyword is the last
word!

                Set<String> hsA, hsB; // the country set for the original tweet and the b-tweet
                hsA = getCountriesFromTweet(a);
                hsB = getCountriesFromTweet(b);

                // TODO Equals or Contains?!
```

```
		if (hsA.equals(hsB) && !hsB.isEmpty() && kh.keyword_filter(a) &&
kh.keyword_filter(b)) {
			return true;
		}
		else
			return false;
	}

	/**	Analyze the tweet and find out which countries contains. Save results for future use
	 *
	 *	@param	a	The tweet to analyze
	 *	@return	A set containing the countries that the tweet contains
	 */
	private Hashtable<String, Set<String>> tweetCountries = new Hashtable<String,
Set<String>>();
	private Set<String> getCountriesFromTweet(String a) {

		// check if the original tweet has been analyzed about its countries before
		if (tweetCountries.containsKey(a))
			return tweetCountries.get(a);
		else {
			// get the countries that appear in the original tweet
			Set<String> hs = new HashSet<String>();
			Vector<String> v = getCalaisEntity(a, "Country");
			if (v != null) hs.addAll(v);

			// get the cities that appear in the original tweet, and find their country
			v = getCalaisEntity(a, "City");
			if (v != null)
				for (String city : v) {
					String country = getCountry(city);
					if (country != null) hs.add(country);
				}

			// get the states that appear in the original tweet and find their country
			v = getCalaisEntity(a, "ProvinceOrState");
			if (v != null)
				for (String state : v) {
					String country = getCountry(state);
					if (country != null) hs.add(country);
				}

			tweetCountries.put(a, hs);
			return hs;
		}
	}

	/**	Responsible for retrieving the country of a given city using GeoNames.org and
saving the results for future use
	 *
	 *	@param	query	The city to be queried
	 *	@return	The country that the city belongs to
	 */
	private Hashtable<String, String> countries = new Hashtable<String, String>();
	private String getCountry(String query) {

		// return the result if already has been queried
		if (countries.containsKey(query)) {
			return countries.get(query);
		}
```

```
		try {

		   WebService.setUserName("netcyrax"); // 30,000 queries / daily

		   ToponymSearchCriteria searchCriteria = new ToponymSearchCriteria();
		   searchCriteria.setName(query);

		   ToponymSearchResult searchResult;

			searchResult = WebService.search(searchCriteria);

			// save the first result for future use
			String firstRes = "";
			if (searchResult.getToponyms().size() == 0)
				firstRes = null;
			else
				firstRes = searchResult.getToponyms().get(0).getCountryName();
			countries.put(query, firstRes);


			/*
		   for (Toponym toponym : searchResult.getToponyms()) {
		     System.out.println(toponym.getName()+" ||| "+ toponym.getCountryName());
		   }
		   */

		} catch (Exception e) {
			e.printStackTrace();
		}

		return countries.get(query);
	}

	/**	Responsible for semantically analyze the RT using OpenCalais and saving the
results for future use
	 *
	 *	@param		RT	The text that we want to semantically analyze
	 *	@param		Entity	The Entity values that we want to receive
	 *	@return		A vector containing all the values for the specified Entity of the
specified text
	 */
	private Hashtable<String, Hashtable<String, Vector<String>>> calaisEntities = new
Hashtable<String, Hashtable<String, Vector<String>>>();
	private Vector<String> getCalaisEntity(String RT, String Entity) {

		// return the Vector if already the RT and Entity was queried
		if (calaisEntities.containsKey(RT)) {
			Hashtable<String, Vector<String>> htRT = calaisEntities.get(RT);

			if (htRT.containsKey(Entity)) {
				return htRT.get(Entity);
			}
			else {
				return null;
			}
		}

		// Set OpenCalais connection options
```

```
		CalaisConfig config = new CalaisConfig();
		config.set(CalaisConfig.ConnParam.CONNECT_TIMEOUT, 15000);
		config.set(CalaisConfig.ConnParam.READ_TIMEOUT, 15000);

			// Query OpenCalais for Named-Entity of the RT
			try {

				CalaisClient client = new CalaisRestClient("gytxbev2gz2258rqcmz3d9mc");
				CalaisResponse response;
				response = client.analyze(RT);

				Hashtable<String, Vector<String>> ht = new Hashtable<String,
Vector<String>>();
			Vector<String> v;
			   for (CalaisObject entity : response.getEntities()) {

				String type = entity.getField("_type");
				String value = entity.getField("name");

				if (!ht.containsKey(type)) {
					v = new Vector<String>();
				}
				else {
					v = ht.get(type);
				}
				v.addElement(value);
				ht.put(type, v);
			   }
			   calaisEntities.put(RT, ht);

			   return calaisEntities.get(RT).get(Entity);

			} catch (IOException e) {
				e.printStackTrace();
				return getCalaisEntity( RT, Entity);
			}
		}

		/**	Prints the pairs that were detected as semantically related
		 *
		 *	@param		pairs	A Vector of Vectors containing all the groups of
semantically related tweets
		 */
		public static void printPairs (Vector<Vector<String>> pairs) {
			System.out.println("\n\npairs:");
			for (Vector<String> v: pairs) {
				System.out.println(v);
			}
		}

		/**	Prints the pairs that were detected as semantically related
		 *
		 *	@param		pairs	A Vector of Vectors containing all the groups of
semantically related tweets
		 *	@param		trees	A hashtable of trees
		 *	@return		the pairs in a printable form
		 */
		public static String stringPairs (Vector<Vector<String>> pairs,
Hashtable<String,GenericTree<Tweet>> trees) {
			String res = "\nSemantically Related Groups:\n\n";
```

```
		for (int i=0; i<pairs.size(); i++) {
			res += "Group "+i+":\n";
			res += stringPair(pairs.get(i), trees);
		}
		return res;
	}

	/**	Prints a single pair (group)
	 *
	 *	@param		pairs	A Vector containing the group
	 *	@param		trees	A hashtable of trees
	 *	@return		the pair in a printable form
	 */
	public static String stringPair (Vector<String> v, Hashtable<String,GenericTree<Tweet>>
trees) {
		String res = "";
		for (String s: v) {
			res += "- "+trees.get(s).getRoot().getData().getUser()+": "+s+"\n";
		}
		res += "\n";

		return res;
	}

}
```

Independence_Scan.java

```
import java.io.BufferedWriter;
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.Vector;

import GenericTree.GenericTree;
import GenericTree.GenericTreeNode;

/** Responsible for returning an independence analysis of the RT-trees
 *
 * @author Mike Yerou
 */
public class Independence_Scan {

	Hashtable<String,GenericTree<Tweet>> trees;
	Vector<Vector<String>> pairs;
	private BufferedWriter output;

	/** Constructor for getting the necessary objects for doing an independence analysis
	 *
	 * @param _pairs	all the semantically related pairs
	 */
	public Independence_Scan(Hashtable<String,GenericTree<Tweet>> _trees,
Vector<Vector<String>> _pairs, BufferedWriter _output) {
		pairs = _pairs;
		output = _output;
		trees = _trees;
	}

	public void independence_scan() {

		int curGroup = 0;

		// for each group, analyze the tweets separately
		for (Vector<String> pair: pairs) {

			Social_Sensing_Rumors.o(3, "Now processing Group "+(curGroup++)+":
\n"+Semantic_Scan.stringPair(pair,trees));

			for (int i=0; i<pair.size()-1; i++)
				for (int j=i+1; j<pair.size(); j++) {

					// find the actual Tweet objects
					GenericTreeNode<Tweet> R1 =
trees.get(pair.get(i)).getRoot();
					GenericTreeNode<Tweet> R2 =
trees.get(pair.get(j)).getRoot();

					analysisRoots_4_1(R1, R2);
					analysisRoots_4_1(R2, R1);

					analysisRoots_4_3(R1, R2);
					analysisRoots_4_3(R2, R1);

				}
```

```
                        Social_Sensing_Rumors.o(3, "-------------------------\n");
                }

        }

        /**     4.1. Ελέγχουμε αν η R1 βρίσκεται στο δέντρο διάδοσης της R2 για το σημασιολογικά κοντινό tweet που εντοπίστηκε στο βήμα 3 (και το ανάποδο).
         *      Αν βρίσκεται, τότε συγκρίνουμε τη χρονική στιγμή κατά την οποία έκανε retweet με αυτήν που έκανε το post.
         *
         * @param R1   The tweet-root 1
         * @param R2   The tweet-root 2
         */
        private void analysisRoots_4_1(GenericTreeNode<Tweet> R1, GenericTreeNode<Tweet> R2) {
                GenericTreeNode<Tweet> foundR2 = Trees_Creator.auxiliaryFind(R1, R2.getData().getUser());
                if (foundR2 != null) {
                        if (foundR2.getData().getDate().before( R2.getData().getDate() )) {
                                Social_Sensing_Rumors.o(3, "Possible dependency 4.1 (η R1 βρίσκεται στο δέντρο διάδοσης της R2 για το σημασιολογικά κοντινό tweet που εντοπίστηκε στο βήμα 3):");
                                Social_Sensing_Rumors.o(3, "R1: "+R1.getData().getDt()+": "+R1.getData().getUser()+": "+R1.getData().getMsg());
                                Social_Sensing_Rumors.o(3, "R2: "+R2.getData().getDt()+": "+R2.getData().getUser()+": "+R2.getData().getMsg());
                                Social_Sensing_Rumors.o(3, "Found R2: "+foundR2.getData().getDt()+": "+foundR2.getData().getUser()+": "+foundR2.getData().getMsg());
                                Social_Sensing_Rumors.o(3, "");
                        }

                }
        }

        /**     4.2. Ελέγχουμε αν η R1 βρίσκεται στο δέντρο διάδοσης της R2 σε οποιοδήποτε άλλο δέντρο (και το ανάποδο).
         *      4.3. Ελέγχουμε αν η R1 και η R2 συνυπάρχουν σε οποιοδήποτε δέντρο διάδοσης, αν βρίσκονται στο ίδιο μονοπάτι από την ρίζα και υπολογίζουμε την απόστασή τους.
         *
         * @param R1   The tweet-root 1
         * @param R2   The tweet-root 2
         */
        private void analysisRoots_4_3(GenericTreeNode<Tweet> R1, GenericTreeNode<Tweet> R2) {

                // iterate through all the trees
                int curTree = -1;
                Enumeration<String> e = trees.keys();
           while(e.hasMoreElements()) {
                curTree++;
                String key = e.nextElement();
                GenericTree<Tweet> tree = trees.get(key);

                // do not check for existance in the root's trees
                        if (!tree.getRoot().equals(R1) && !tree.getRoot().equals(R2)) {

                                // try to find the R1 in the tree
                                GenericTreeNode<Tweet> foundR1 = Trees_Creator.auxiliaryFind(tree.getRoot(), R1.getData().getUser());
                                if (foundR1 != null) {
```

```
                                        // try to find if the R2 is accessible from R1 and calculate
their distance
                                        int depth = Trees_Creator.auxiliaryFindWithDepth(foundR1,
R2.getData().getUser(), 0);
                                        // TODO Should i check for date of post as well?!
                                        if (depth != -1) {
                                                GenericTreeNode<Tweet> foundR2 =
Trees_Creator.auxiliaryFind(tree.getRoot(), R2.getData().getUser());

                                                Social_Sensing_Rumors.o(3, "Possible
dependency 4.3 (η R1 βρίσκεται στο δέντρο διάδοσης της R2 σε οποιοδήποτε άλλο δέντρο, βρίσκονται
στο ίδιο μονοπάτι από την ρίζα και υπολογίζουμε την απόστασή τους):");
                                                Social_Sensing_Rumors.o(3, "in Tree "+curTree+":
"+tree.getRoot().getData().getMsg());
                                                Social_Sensing_Rumors.o(3, "R1:
"+foundR1.getData().getDt()+": "+foundR1.getData().getUser()+": "+foundR1.getData().getMsg());
                                                Social_Sensing_Rumors.o(3, "R2:
"+foundR2.getData().getDt()+": "+foundR2.getData().getUser()+": "+foundR2.getData().getMsg());
                                                Social_Sensing_Rumors.o(3, "Distance: "+depth);
                                                Social_Sensing_Rumors.o(3, "");
                                        }
                                        else {
                                                GenericTreeNode<Tweet> foundR2 =
Trees_Creator.auxiliaryFind(tree.getRoot(), R2.getData().getUser());
                                                if (foundR2 != null) {
                                                        Social_Sensing_Rumors.o(3, "Possible
dependency 4.2 (η R1 και η R2 συνυπάρχουν σε οποιοδήποτε δέντρο διάδοσης):");
                                                        Social_Sensing_Rumors.o(3, "in Tree
"+curTree+": "+tree.getRoot().getData().getMsg());
                                                        Social_Sensing_Rumors.o(3, "R1:
"+foundR1.getData().getDt()+": "+foundR1.getData().getUser()+": "+foundR1.getData().getMsg());
                                                        Social_Sensing_Rumors.o(3, "R2:
"+foundR2.getData().getDt()+": "+foundR2.getData().getUser()+": "+foundR2.getData().getMsg());
                                                        Social_Sensing_Rumors.o(3, "");
                                                }
                                        }
                                }
                        }
                }
        }
}
```