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

Υλοποίηση Αλγόριθμου Συστάσεων που κάνει χρήση Τριγραμμάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ιωάννη Α. Βιόλου

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα: Δεκέμβριος 2013

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

Υλοποίηση Αλγόριθμου Συστάσεων που κάνει χρήση Τριγραμμάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ιωάννη Α. Βιόλου

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 16η Δεκεμβρίου 2013.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Θεοδώρα Βαρβαρίγου Καθηγήτρια Ε.Μ.Π.	Ελευθέριος Καγιάφας Καθηγητής Ε.Μ.Π.	Βασίλειος Λούμος Καθηγητής Ε.Μ.Π.

Αθήνα Δεκέμβριος 2013

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

(Υπογραφή)

.....

Ιωάννης Α. Βιόλος

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

Copyright © Ιωάννης Α. Βιόλος, 2013
Με επιφύλαξη παντός δικαιώματος. All rights reserved.
Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον διδάκτορα Κωνσταντίνο Τσερπέ για την βοήθεια που μου παρείχε όλον αυτό τον καιρό για την εκπόνηση αυτής της διπλωματικής εργασίας. Ήτανε πάντοτε διαθέσιμος και πρόθυμος να μου λύσει οποιαδήποτε απορία, να με συμβουλεύσει και να μου δώσει τις σωστές κατευθύνσεις πάνω στις οποίες εργάστηκα και την υλοποίησα. Γνωρίζω καλά ότι χωρίς την δική του συμβολή αυτή η διπλωματική έρευνα δεν θα είχε τόσα πολλά να παρουσιάσει.

Έπειτα θα ήθελα να ευχαριστήσω τους καθηγητές μου και ιδιαίτερα την Καθηγήτρια Θεοδώρα Βαρβαρίγου που όλα αυτά τα χρόνια στο Πολυτεχνείο μου πρόσφεραν μια υψηλού επιπέδου ακαδημαϊκή κατάρτιση. Με τις διαλέξεις τους και τα εργαστήρια τους μεταλαμπάδευσαν την γνώση και το ενδιαφέρον του Αντικειμένου του Ηλεκτρολόγου Μηχανικού και Μηχανικού Υπολογιστών μέσα μου. Θα είναι για μένα πάντοτε ένα σημείο αναφοράς γιατί με το παράδειγμα τους χάραξαν ανεξίτηλα μέσα μου τις αρχές της επιστημονικής σκέψης και της τεχνολογικής μεθοδολογίας όπου σύμφωνα με αυτές καλείται ο σύγχρονος μηχανικός να επιλύσει προβλήματα που κάνουν καλύτερη την ζωή του Ανθρώπου.

Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου και τους κοντινούς μου φίλους που με στήριξαν ηθικά αλλά και υλικά για να ολοκληρώσω τις σπουδές μου και ως κορύφωση αυτών των σπουδών να συγγράψω την διπλωματική μου. Γνωρίζω καλά πως όλες αυτές οι στερήσεις και δυσκολίες που είχα όλον αυτόν τον καιρό δεν ήταν μόνο δικές μου αλλά και δικές τους. Πολλές φορές θα ήθελαν να είμαι μαζί τους αλλά έδειχναν κατανόηση που ήμουν απών γιατί αναγνώριζαν ότι είναι προτεραιότητα να αφιερώσω τον χρόνο μου και την ενέργεια μου στην Επιστημονική έρευνα και στην δημιουργία καινούριας Τεχνολογίας.

Περίληψη

Τα Κοινωνικά Δίκτυα είναι ένας διαρκώς αυξανόμενος επιστημονικός κλάδος. Υλοποιούνται προγραμματιστικά με έναν συνδυασμό Linux, Apache, MySQL και PHP/Python/Perl (LAMP) και συνδυάζουν τεχνολογίες όπως Στοιχεία Αξιολόγησης, Συνεργατικές Ετικέτες, Κοινωνικοί Σελιδοδείκτες, Επεξεργασία Κοινωνικών Πληροφοριών, Συσχετισμός Πελατών, Εξατομικευμένο Marketing, Συστήματα Υπόληψης, Αυτοματοποιημένες Περιλήψεις, Συστήματα Συστάσεων.

Εμείς ασχοληθήκαμε αναλυτικά με τα Συστήματα Συστάσεων. Δηλαδή αλγορίθμους και μεθόδους που μπορούν να κάνουν συστάσεις σε χρήστες. Οι συστάσεις μπορούν να είναι άλλοι χρήστες, κάποιο προϊόν, υπηρεσία, δημοσιογραφικά άρθρα και γενικώς οτιδήποτε μπορεί να ενδιαφέρει έναν χρήστη.

Διάσημοι τέτοιοι αλγόριθμοι είναι ο αλγόριθμος που χρησιμοποιεί Συνεργατικά Φίλτρα όπου ο σκοπός του είναι να δημιουργήσουμε προφίλ χρηστών για τις προτιμήσεις τους καθώς και σχέσεις μεταξύ χρηστών. Η ιδέα βασίζεται στην αρχή ότι αυτό που ενδιέφερε τους φίλους μου πολύ πιθανόν να ενδιαφέρει και εμένα. Τα Φίλτρα που Βασίζονται στο Περιεχόμενο χρησιμοποιούν την περιγραφή ενός αντικειμένου αλλά και το προφίλ ενδιαφερόντων που έχει κάθε χρήστης. Η ιδέα βασίζεται στο ότι ένας χρήστης έχει ένα πλήθος ενδιαφερόντων και κάθε αντικείμενο ικανοποιεί ένα πλήθος ενδιαφερόντων. Όσο πιο πολύ ταυτίζονται αυτά τα δύο τόσο πιο μεγάλο θα είναι το ενδιαφέρον του χρήστη προς το αντικείμενο αυτό. Άλλα Συστήματα συστάσεων κάνουν χρήση τεχνικών Εξόρυξης Γνώσης, Παραπομπές, χρήση του Αλγόριθμου k Κοντινότερων Γειτόνων. Άλλα βασίζονται στο πόση Ώρα κάθε χρήστης είδε μια ιστοσελίδα, σε Ετικέτες, Συσχετιστικούς Κανόνες, υβρίδια μεταξύ όλων αυτών και σε συστάσεις που χρησιμοποιούν την Αυτοματοποιημένη Περίληψη Πολλών Κειμένων μία μέθοδο στην οποία μπορεί να παραχθεί περίληψη από πολλά κείμενα που έχει γράψει κάθε χρήστης και έπειτα να συγκριθούν αναμεταξύ τους.

Το Σύστημα συστάσεων που υλοποιήσαμε βασίζεται στα κείμενα που δημοσιεύουν οι χρήστες. Από καθενός χρήστη τα κείμενα παράγεται ένας γράφος. Οι κόμβοι αυτού του γράφου θα είναι τα τριγράμματα που βρίσκονται στα αρχικά κείμενα και οι ακμές δηλώνουν του κατά

πόσο αυτά τα τριγράμματα είναι κοντά αναμεταξύ τους στα αρχικά κείμενα. Αν συγκρίνουμε τους γράφους που έχουν παραχθεί για κάθε χρήστη θα έχουμε μια ένδειξη του κατά πόσο έχουν κοινά ενδιαφέροντα αναμεταξύ τους οι χρήστες. Υψηλή ομοιότητα γράφων θα συνεπάγεται ότι μοιράζονται πολλά κοινά ενδιαφέροντα.

Οι μέθοδοι σύγκρισης γράφων που χρησιμοποιήσαμε ήταν η ομοιότητα περιεχομένου, όπου μας νοιάζει η ομοιότητα του γράφου χωρίς να παίρνουμε υπόψιν το βάρος των ακμών και η ομοιότητα κατά αξία όπου προσμετράται και κατά πόσο δύο κοινές ακμές έχουν το ίδιο βάρος.

Τα αποτελέσματα που βρήκαμε ήταν αρκετά ικανοποιητικά σε δοκιμές που έγιναν στο τουίτερ. Στις περισσότερες περιπτώσεις είχαμε υψηλή ομοιότητα μεταξύ χρηστών που ασχολούνται με κοινά θέματα. Χαμηλή σε χρήστες που ασχολούνται με διαφορετικά θέματα και τέλος προβλέφθηκε η σχέση μεταξύ χρηστών που ασχολούνται με συγγενικά θέματα.

Λέξεις Κλειδιά

Κοινωνικά Δίκτυα, Σύστημα Συστάσεων, Τριγράμματα, Σύγκριση Γράφων, Ομοιότητα Περιεχομένου, Ομοιότητα κατά Αξία. Συνεργατικά Φίλτρα, Φίλτρα που Βασίζονται στο Περιεχόμενο, Τουίτερ, Αυτοματοποιημένη Περίληψη Πολλών Κειμένων, Αλγόριθμος k Κοντινότερων Γειτόνων, Χρόνος Παρατήρησης, Παραπομπές στον Παγκόσμιο Ιστό

Abstract

Social Networks is an increasing scientific field . We can implement it with a combination of Linux, Apache, MySQL and PHP / Python / Perl (LAMP) and combine technologies such as Rating Features, Collaborative Tagging, Social Bookmarking, Social Information Processing, Customer Engagement, Personalized Marketing, Reputation Systems, Automated Summaries, Recommendation Systems .

We dealt in detail with Recommendation Systems . Algorithms and methods that can make recommendations to users . Recommendations may be other users , any product, service, newspaper article and generally anything that might interest a user .

Famous algorithm is the algorithm that uses Collaborative Filters where the purpose is to create user profiles for their preferences and relationships between users. The idea is based on the principle that me and my friends share the same interests. The Content Based methos combines the description of an object and the profile of interests that each user has . The idea is that a user has a number of interests and each object has a number of interests that satisfies. As much matching these two the greater will be the user's interest in that subject . Other Recommendation systems use Data Mining techniques , Referral Web, using the k-Nearest Neighbor algorithm. Others based on the Time View, in Tags , Referral Web , hybrids between the previous and the Multi-Document Summarization which is a method to be produced a summary of many texts which have been written by users and then be compared against each other .

The recommendation system which we implemented is based on texts by users . A graph is produced by each user texts . The nodes of this graph is the trigrams that found in the original texts and edges indicate whether these Trigrams are near each other in the early texts. If we compare the graphs generated for each user we will have a prediction if there are common interests among the users . High similarity between graphs would imply that the users share many common interests.

The Comparison Methods of Graph which we used was the Content Similarity , where we compute the similarity of the graph without taking into account the weight of the edges and the value similarity where it computed if two common edges have the same weight.

The results in tests conducted in tweeter were pretty satisfactory. In most cases we had high

similarity between users dealing with common issues. Low similarity between users dealing with different issues and finally our method predicted the relationship between users dealing with related issues.

Keywords

Social Networks, Recommendation System, Trigram, Comparing Graphs, Containment Similarity, Value Similarity, Collaborative Filtering, Content Based Filtering, Twiiter, Multi-Document Summarization, k-Nearest Neighbor Algorithm, View Time, Referral Web

Περιεχόμενα

Κατάλογος Σχημάτων και Πινάκων

1 Μια εισαγωγή στα Κοινωνικά Δίκτυα

Μέρα με την μέρα βλέπουμε τα κοινωνικά δίκτυα του internet να μπαίνουν όλο και πιο πολύ στην ζωή μας. Για τους περισσότερους ανθρώπους τα κοινωνικά δίκτυα είναι αναπόσπαστο κομμάτι της καθημερινότητα τους. Συνέχεια ακούμε από τους φίλους μας, τους γνωστούς μας, τους συγγενείς μας ότι έχουν έναν λογαριασμό σε κάποιο κοινωνικό δίκτυο όπως το Facebook, το Twitter, το myspace και πολλά άλλα όπου συνδέονται, επικοινωνούν, πράττουν και σχετίζονται με άλλους ανθρώπους.

Από την δική μας πλευρά ως μηχανικοί υπολογιστών αυτό είναι ένα καινούριο ερευνητικό θέμα που βλέπουμε να αναπτύσσεται και αναπτύσσουμε με μεγάλη ταχύτητα. Αναπτύσσεται γιατί οι άνθρωποι ως κοινωνικά όντα αναζητούν την κοινωνική επαφή, αναζητούν την επικοινωνία, αναζητούν την πληροφόρηση όποτε με το να τους παρέχεται ένα κοινωνικό δίκτυο θέλουν να γίνουν μέλη, να συσχετιστούν μέσα σε αυτό και να το εμπλουτίσουν με τα προσωπικά τους στοιχεία. Από την άλλη πλευρά προαπαιτούμενο είναι εμείς ως μηχανικοί και προγραμματιστές να το αναπτύξουμε, να του προσφέρουμε δυνατότητες, δίνοντάς του εργαλεία τα οποία θα κάνουν ακόμη περισσότερο τους ανθρώπους να θέλουν να γίνουν χρήστες του και να υπάρξουν μέσα σε αυτό.

Ένας καινούριος κλάδος επιστημονικού και τεχνολογικού ενδιαφέροντος έχει γεννηθεί. Σκοπός του είναι η μελέτη των κοινωνικών δικτύων. Τον τρόπο του σχετίζεσθαι των μελών του. Των δυνατοτήτων που προσφέρει στους χρήστες του. Την παραγωγή γνώσης και πληροφορίας μέσα από αυτά. Ακόμη καλύτερα την προσφορά πληροφορίας στους ανθρώπους που την χρειάζονται.

Έχει μεγάλο πρακτικό ενδιαφέρων σε ένα δίκτυο που συνδέονται χιλιάδες ή εκατομμύρια χρήστες, γράφονται εκατομμύρια κείμενα, αναρτούνται εκατομμύρια εικόνες και γενικώς γίνονται εκατομμύρια ενέργειες μέσα από τις πολλές δυνατότητες που προσφέρονται να βρούμε

έναν συστηματικό τρόπο να προτείνουμε να έρθουν σε επαφή άνθρωποι που ο ένας ενδιαφέρει για κάποιον λόγο τον άλλον (παλιοί φίλοι, συνάδελφοι, κοινά χόμπι), να μπορούμε να προτείνουμε κείμενα σε ανθρώπους που ενδιαφέρονται να τα διαβάσουν, να προτείνουμε εικόνες σε ανθρώπους που θέλουν να τις δούνε. Δίνουμε βαρύτητα στην λέξη προτείνουμε. Γιατί είναι η λέξη, είναι η έννοια που θα μας απασχολήσει περισσότερο.

Το θέμα μας είναι κατά πόσο μπορούμε με ένα συστηματοποιημένο τρόπο να κατανοήσουμε ένα χρήστη το ποίοι και τι μπορεί να τον ενδιαφέρουν από άλλους χρήστες, κείμενα, εικόνες, βίντεο, γενικώς πληροφορίες και δυνατότητες που μπορούν να υπάρχουν μέσα σε αυτό το αχανές κοινωνικό δίκτυο και να τις προτείνει σε αυτόν. Κατά αντίστοιχα μας ενδιαφέρει κάθε μία μονάδα, κείμενο, βίντεο, μουσική, εργαλείο να προταθεί σε όσους το δυνατόν περισσότερους χρήστες μπορεί να τους ενδιαφέρει και να μην προταθεί σε χρήστες που δεν τους ενδιαφέρει. Δεν θέλουμε χρήσιμη πληροφορία να είναι καταχωνιασμένη και παραμελημένη ανάμεσα σε χιλιάδες ηλεκτρονικές σελίδες ενώ θα μπορούσε να ήταν αντικείμενο ενδιαφέροντος για άλλους ανθρώπους. Αυτό έρχεται να κάνει ένα σύστημα συστάσεων (recommendation system)

Αυτή την στιγμή σχεδόν όλοι οι μεγάλοι δικτυακοί ιστότοποι όπως Google, Facebook, Amazon, YouTube χρησιμοποιούν recommendation systems για να κάνουν συστάσεις σε αυτούς που τις επισκέπτονται. Ακόμη και μια απλή ιστοσελίδα που εμπορεύεται προϊόντα θα προσπαθήσει να κατανοήσει τον κάθε επισκέπτη της και να του προβάλει παράλληλα με την περιήγηση του μέσα της, προϊόντα και διαφημίσεις που πιθανών να τον ενδιαφέρουν.

Έχουν προταθεί πληθώρα αλγορίθμων που μπορούν να υλοποιήσουν ένα Recommendation system, ο καθένας προσεγγίζοντας το θέμα από μια διαφορετική γωνία και χρησιμοποιώντας διαφορετικά κριτήρια.

Στην συνέχεια θα αποσαφηνίσουμε τον όρο social network δίνοντας μια σύντομη περιγραφή του τι είναι. Θα περιγράψουμε τι είναι ένα recommendation system και θα αναφερθούμε εκτενώς παρουσιάζοντας πολλές μεθόδους υλοποίησης τους, αναφέροντας αναλυτικά πως λειτουργούν, που υπερτερεί και μειονεκτεί κάθε μέθοδος. Τέλος θα παρουσιάσουμε κάποιες προγραμματιστικές υλοποιήσεις σε κάποια από τα recommendation system που θα έχουμε παρουσιάσει.

1.1 Ορισμός Κοινωνικού Δικτύου (Social Network)

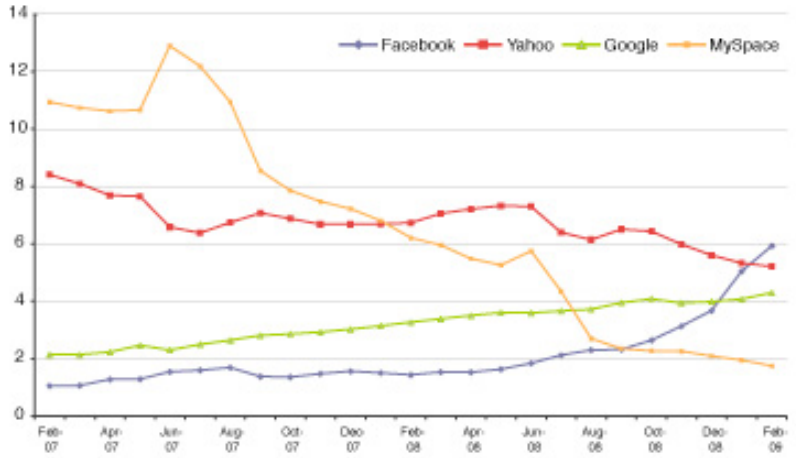
Κοινωνικό δίκτυο καλείται μια κοινωνική δομή που μπορεί να παρομοιαστεί με έναν γράφο. Αυτός ο γράφος αποτελείται από κόμβους που γενικά είναι άνθρωποι, ομάδες ανθρώπων και οργανισμοί καθώς και ακμές που συνδέουν τους κόμβους. Οι ακμές αυτές μπορούν να αντιπροσωπεύουν φιλίες, επαγγελματικές συνεργασίες εμπορικές και οικονομικές συναλλαγές

και γενικώς οποιουδήποτε είδους σχέση και ενδιαφέρον μπορεί να υπάρχει μεταξύ των κόμβων. Ο γράφος που παράγεται συνήθως είναι ένας πολύ περίπλοκος γράφος και αν αναλογιστούμε πόσα πολλά άτομα μπορεί να εκπροσωπεί καταλαβαίνουμε και πόσους πολλούς κόμβους και ακμές έχει κατά αντιστοιχία.

Σε έρευνα που παρουσιάζει μια από τις μεγαλύτερες παγκοσμίως διαφημιστικές εταιρείες η nielsen Company. Το μεγαλύτερο ποσοστό χρήσης του internet γίνεται σε social networks η χρήση του Facebook ξεπερνά ακόμη και την χρίση του Google και οποιουδήποτε άλλου δικτυακού τόπου. Τι είναι αυτό όμως που κάνει τους ανθρώπους να θέλουν να γίνουν χρήστες ενός κοινωνικού δικτύου και να αφιερώσουν σε αυτό τόσο πολύ χρόνο;

Top 10 Sectors by Share of U.S. Internet Time				
RANK	Category	Share of Time June 2010	Share of Time June 2009	% Change in Share of Time
1	Social Networks	22.7%	15.8%	43%
2	Online Games	10.2%	9.3%	10%
3	E-mail	8.3%	11.5%	-28%
4	Portals	4.4%	5.5%	-19%
5	Instant Messaging	4.0%	4.7%	-15%
6	Videos/Movies	3.9%	3.5%	12%
7	Search	3.5%	3.4%	1%
8	Software Manufacturers	3.3%	3.3%	0%
9	Multi-category Entertainment	2.8%	3.0%	-7%
10	Classifieds/Auctions	2.7%	2.7%	-2%
	Other	34.3%	37.3%	-8%
Source: The Nielsen Company				

Πίνακας 1 Ποσοστά χρήσης του Διαδικτύου



Σχήμα 1Ποσοστά χρήσης του Διαδικτύου

Οι λόγοι που θα φέρουν έναν άνθρωπο να γίνει χρήστης ενός κοινωνικού δικτύου στο internet είναι η αναζήτηση επικοινωνίας του ανθρώπου με άλλους ανθρώπους, ο σχολιασμός ενός γεγονότος σε κοινωνικό επίπεδο, η αίσθηση ότι δεν χάνεις επαφή και γνωριμίες με ανθρώπους έστω και αν τους έχεις μιλήσει μόνο ένα βράδυ στις καλοκαιρινές σου διακοπές. Μέσο ενός κοινωνικού δικτύου μπορούμε να μοιραστούμε εικόνες λόγια, ήχους σκέψεις βίντεο με οποιονδήποτε όπου και αν βρίσκεται. Ολόκληρες κοινότητες ανθρώπων που έχουν κοινά ενδιαφέρονται μπορούν να χρησιμοποιήσουν τις δυνατότητες ενός κοινωνικού δικτύου για να μοιραστούν και να ασχοληθούν περαιτέρω τα ενδιαφέροντα τους.

Τα κοινωνικά δίκτυα εμπλουτίζονται από τους ίδιους τους χρήστες αφού αυτοί είναι τα πρόσωπα που τα γεμίζουν κάθε μέρα με έναν απέραντο πλούτο πληροφοριών. Θεωρείται ότι είναι η σημαντικότερη αλλαγή από το web 1 στο web 2. Οι προγραμματιστές και οι μηχανικοί λογισμικού παίζουν έναν επικουρικό ρόλο σε όλα αυτά. Προσφέρουν δυνατότητες και εργαλεία που θα χρησιμοποιήσουν οι χρήστες για να γεμίσουν το κοινωνικό δίκτυο με προσωπικά τους στοιχεία και θα ενεργήσουν δια μέσω αυτών.

Κάθε χρήστης θα πρέπει να έχει ένα προφίλ που υποδηλώνει την παγκόσμια του ταυτότητα μέσα από αυτό μπορεί να προβάλει κάποια βασικά του χαρακτηριστικά. Που μένει, τι επαγγέλλεται, ηλικία, φωτογραφία του μέχρι και ποιες είναι οι αγαπημένες του ταινίες ή την αγαπημένη του φράση. Έπειτα το κοινωνικό δίκτυο δομείται από τις σχέσεις μεταξύ των χρηστών που υποδηλώνουν μια παγκόσμια κοινωνική αποτύπωση. Κάθε χρήστης έχει φίλους, συγγενείς συνεργάτες που συνδέεται μαζί τους και ανήκει σε ομάδες ανθρώπων. Μια σχέση μπορεί να δημιουργηθεί ακόμη και από το σύνολο χρηστών που σχολίασαν μια φωτογραφία. Τέλος κάθε χρήστης ή ομάδα χρηστών έχει κάποιες δραστηριότητες που αντανακλούν το κοινωνικό περιεχόμενο των ενεργειών που κάνουν. παράδειγμα να δηλώσει ότι του αρέσει μια φωτογραφία ενός άλλου χρήστη, να συζητήσει με άλλους χρήστες ή να δημοσιοποιήσει ένα προσωπικό του βίντεο. Παρακάτω θα κάνουμε μια σύντομη παρουσίαση των πιο γνωστών κοινωνικών δικτύων.

Βιβλιογραφία:

[http://blog.nielsen.com/nielsenwire/online_mobile/what-americans-do-online-social-media-and-games-dominate-activity/]

[http://www.intelligentpositioning.com/blog/2009/03/which-site-do-people-spend-most-time-on-facebook/]

1.2 Σύντομη παρουσίαση του Myspace

Ιδρύθηκε το 2003 στην Καλιφόρνια από τους Thomas G. Anderson και Chris DeWolfe. Τον Ιανουάριο του 2008 είχε 110 εκατομμύρια ενεργούς χρήστες ενώ τώρα λέγεται ότι έχουν μειωθεί στα 34 εκατομμύρια. Το χρησιμοποιούν όμως και άνθρωποι που δεν είναι εγγεγραμμένοι χρήστες αυξάνοντας σε 43.2 εκατομμύρια τους επισκέπτες κάθε μήνα. Το MySpace ήταν πιο δημοφιλές κοινωνικό δίκτυο από το 2006 έως το 2008 σύμφωνα με το comScore. Περιλαμβάνει χρήστες που αλληλεπιδρούν αναμεταξύ τους, προσωπικά προφίλ, ιστολόγια, φίλους, ομάδες χρηστών, βίντεο, μουσική και φωτογραφίες. Το περιβάλλον του ευνοεί την χρίση του από καλλιτέχνες ιδιαίτερα μουσικούς που θέλουν να προβάλουν την δουλειά τους. Από προγραμματιστικής πλευράς έχει ενδιαφέρον η πλατφόρμα ανάπτυξης εφαρμογών MySpace Developer Platform (MDP) που βασίζεται στο Open Social API της Google και επιτρέπει ανοιχτά σε οποιονδήποτε την γραφή και ενσωμάτωση προγραμμάτων μέσα στο MySpace. Το 2008 ο Dan Farino που ήταν ο Chief System Architect του MySpace ανακοίνωσε ότι το MySpace έστελνε 100 gigabits δεδομένων ανά δευτερόλεπτο στο internet καθώς και το ότι συστεγαζόταν σε περισσότερους από 4.500 web servers, 1.200 cache servers και 500 Database servers.

1.3 Σύντομη παρουσίαση του Facebook

Ιδρύθηκε το 2004 στην Μασαχουσέτη απο τους Mark Zuckenberg, Eduardo Saverin, Dustin Moskovitz και Chris Hughes. Τον Ιανουάριο του 2011 είχε 600 εκατομμύρια ενεργούς χρήστες. Το Facebook αυτή την στιγμή είναι το πιο διαδεδομένο κοινωνικό δίκτυο και γενικώς η ιστοσελίδα που οι χρήστες του internet περνάνε τον περισσότερο χρόνο τους. Για να το χρησιμοποιήσει κάποιος πρέπει να είναι εγγεγραμμένο μέλος του. Οι χρήστες του έχουν την δυνατότητα να γίνουν μέλη σε ομάδες χρηστών καθώς και να αναζητήσουν χρήστες κατά κοινά θέματα που τους ενδιαφέρουν, κατά πόλεις, κατά σχολεία που φοίτησαν και χώρους εργασίας. Υπάρχει η δυνατότητα να προσθέτουν φίλους, να προσκαλούν φίλους σε ένα γεγονός (event), να μιλούν αναμεταξύ τους είτε απευθείας (chat) ή να αφήνουν μηνύματα. Ιδιαίτερο ενδιαφέρον έχει μια ιστοσελίδα που αντιστοιχεί σε κάθε χρήστη που ονομάζεται τοίχος (wall) όπου εκεί μπορεί ο ίδιος ή κάποιος φίλος του χρήστη να αναρτά ένα κείμενο, μια φωτογραφία ένα βίντεο ακόμη και έναν σύνδεσμο (link). Αυτό έπειτα μπορεί να γίνει αντικείμενο περαιτέρω σχολιασμών και συζητήσεων μεταξύ των φίλων του. Αυτός ο τρόπος συνηθίζεται για να κοινοποιήσει ένα χρήστης τι τον απασχολεί αυτή την περίοδο, με τι ασχολείται και τι του αρέσει. Κάθε χρήστης έχει ένα προσωπικό προφίλ όπου δημοσιοποιεί όσα από τα προσωπικά του στοιχεία θέλει. Στοιχεία όπως φωτογραφίες, ηλικία, που μένει με τι ασχολείται, ενδιαφέροντα και άλλα. Τα στοιχεία αυτά καθώς και τι δραστηριότητες έχει μέσα στο Facebook μπορεί να επιλέξει σε ποιους θα κοινοποιούνται και σε ποιους όχι. Το Facebook έχει γίνει αντικείμενο έντονων

κριτικών και έχει απαγορευθεί σε πολλές χώρες όπως Κίνα, Πακιστάν, Συρία, Βιετνάμ, Ιράν, Μπανγκλαντές και Ουζμπεκιστάν. Πολλές φορές με την κατηγορία ότι προωθεί αντί-ισλαμικές ιδέες, θρησκευτικές και πολιτικές διακρίσεις.

1.4 Σύντομη παρουσίαση του Twitter

Ιδρύθηκε το 2006 στο Σαν Φρανσίσκο και την Καλιφόρνια από τους Jack Dorsey, Evan Williams, Biz Stone. Έχει περισσότερους από 200 εκατομμύρια χρήστες (2010). Το Twitter συνδυάζει ένα κοινωνικό δίκτυο με μια microblogging ιστοσελίδα. Η δομή του είναι ότι ένας χρήστης ακολουθεί κάποιους άλλους χρήστες που τον ενδιαφέρει το τι θα πουν και το τι κάνουν. Αντίστοιχα όσοι άλλοι χρήστες ενδιαφέρονται για αυτόν μπορούν να τον ακολουθήσουν Με το ρίμα "ακολουθώ" (follow) που είναι διαδεδομένο στους όρους του Twitter εννοούμε το εξής. Κάθε χρήστης έχει την δυνατότητα να γράψει ένα μικρο κείμενο μήκους 140 χαρακτήρων όπου μπορεί να αναφέρει κάποιο νέο, μια είδηση, κάτι που του συνέβη ή σκεφτικέ γενικώς μια πληροφορία. Όσοι χρήστες έχουν επιλέξει να τον ακολουθήσουν θα ενημερωθούν για αυτό το κείμενο και μπορούν να το διαβάσουν. Αντίστοιχα ο κάθε χρήστης μπορεί να λάβει και να διαβάσει όσα από τα κείμενα έχουν συντάξει όλοι αυτοί που έχει επιλέξει να ακολουθεί. Κάθε ένα από αυτά τα κείμενα ονομάζεται tweet. Υπολογίζεται πως κάθε μέρα γράφονται περισσότερα από 65 εκατομμύρια tweets.

1.5 Σύντομη παρουσίαση του Hi5

Ιδρύθηκε το 2003 στην Καλιφόρνια από τον Ramu Yalamanchi. Τον Ιανουάριο του 2009 ο ιστοχώρος ισχυριζόταν ότι είχε πάνω από 60 εκατομμύρια χρήστες. Έχει πολλά κοινά στοιχεία με το Facebook όπως προφίλ,το ότι οι χρήστες μπορούν να ανεβάσουν φωτογραφίες, να γράψουν σχόλια, να παίξουν online παιχνίδια. Εδώ οι φίλοι κατατάσσονται σε 1ου ή 2ου βαθμού ανάλογα με το αν είναι απευθείας συνδεδεμένοι φίλοι ή μέσω κάποιου άλλου κοινού φύλου. Ο βαθμός φτάνει και τον 3ο υποδηλώνοντας τους φίλους κάποιου που είναι 2ο βαθμού φίλος. Σύμφωνα με το ComScore το 2008 το hi5 ήταν το τρίτο πιο δημοφιλές κοινωνικό δίκτυο. Έχει δεχθεί κριτικές ότι χρησιμοποιεί μεθόδους spamming για να αποχτήσει πιο πολλούς χρήστες. Παράδειγμα όταν κάποιος χρήστης εγγράφεται αν δεν επιλέξει να κάνει uncheck μια επιλογή αυτομάτως θα σταλθεί ένα mail που θα προσκαλεί όλους του τις επαφές που έχει στο account του mail του.

1.6 Σύντομη παρουσίαση του Habbo

Ιδρύθηκε το 2000 στην Φιλανδία από τους Sampo Karjalainen και Aapo Kyrola. Έως τον Φεβρουάριο του 2011 περισσότεροι από 203 εκατομμύρια λογαριασμοί έχουν δημιουργηθεί αν και πολλοί από αυτούς δεν είναι ενεργοί. Πάντως κάθε μήνα έχει 18 εκατομμύρια επισκέπτες και εκατό χιλιάδες avatars δημιουργούνται κάθε μέρα. Το Habbo είναι ποιο πολύ δημοφιλές σε έφηβους και στις πιο νεαρές ηλικίες. Το κυρίαρχο στοιχείο είναι το Hotel στο οποίο έχουν πρόσβαση οι χρήστες. Η δομή του είναι ότι όταν ένα χρήστης μπει στο hotel βλέπει μπροστά του αυτό που οι χρήστες ονομάζουν Hotel view και από εκεί ο κάθε χρήστης μπορεί να επικοινωνήσει με όπιον θέλει μέσω του Habbo chat. Το Hotel έχει δύο ειδών δωμάτια . Τα Public rooms και τα Guest rooms. Τα Public rooms είναι δημόσια δωμάτια όπου μπορεί να μπει ο καθένας. Συχνά απεικονίζονται σαν να είναι ρεστοράν, clubs ή κινηματογράφοι. Πολλά από αυτά έχουν κάποια αυτοματοποιημένα ρομπότ τα όπια δίνουν κάποια αντικείμενα όπως ποτά και μιλάνε στους χρήστες. Πολλά δημόσια δωμάτια έχουν κάποια ομαδικά παιχνίδια. Τα Guest rooms δημιουργούνται από χρήστες. Αυτά τα δωμάτια μπορούν να σχεδιαστούν και να μορφοποιηθούν εμφανισιακά έτσι όπως θέλουν οι χρήστες. Στα δωμάτια αυτά συνήθως έχουν πρόσβαση συγκεκριμένοι χρήστες ή οι χρήστες που έχουν τον κωδικό πρόσβασης τους Η περιήγηση μέσα στα δωμάτια μπορεί να γίνει με τον navigator. Επίσης υπάρχουν δύο ειδών νομίσματος τα Credits (ή Coins) και τα Pixels με τα οπαία συνήθως αγοράζονται έπιπλα.

1.7 Σύντομη παρουσίαση του Linkedin

Ιδρύθηκε το 2003 στην Καλιφόρνια από τους Reid Hoffman, Allen Blue, Konstantin Guericke, Eric Ly και Jean-Luc Vaillant. Τον Μάρτιο του 2011 ξεπέρασε τα εκατό εκατομμύρια χρήστες, ενώ κάθε μήνα έχει 46,7 εκατομμύρια επισκέπτες. Είναι ένα κοινωνικό δίκτυο που επικεντρώνεται κυρίως σε επιχειρηματικά ενδιαφέροντα. Οι χρήστες του μπορούν να αναζητήσουν δουλειά, συνεργάτες και επιχειρηματικές ευκαιρίες. Από την άλλη πλευρά οι εργοδότες μπορούν να περιγράψουν μια θέση εργασίας να δεχθούν αιτήσεις από ενδιαφερόμενους και να επιλέξουν ανάμεσά τους. Μία ακόμη δυνατότητα που παρέχεται στους χρήστες του είναι να αναζητήσουν πληροφορίες για μια εταιρία που ενδιαφέρονται να εργαστούν. Παρέχονται πληροφορίες όπως το ποσοστό των θέσεων εργασίας ανά κατηγορία, την τοποθεσία της εταιρίας, μια λίστα από παρόντες και πρώην εργαζόμενους ακόμη και την αναλογία αντρών και γυναικών εργαζόμενων. Υπάρχουν και στο Linkedin groups και μάλιστα

είναι περισσότερα από 870 χιλιάδες . Τις περισσότερες φορές τέτοια γκρουπ συνιστώνται από συναδέλφους ή συνεργάτες για να συζητήσουν τα τρέχοντα θέματα που τους απασχολούν.

1.8 Σύντομη παρουσίαση του Friendster

Ιδρύθηκε το 2002 στην Καλιφόρνια από τους Jonathan Abrams και Peter chin. Είναι ένα από τα μεγαλύτερα κοινωνικά δίκτυα με περισσότερους από 115 εκατομμύρια εγγεγραμμένους χρήστες και περισσότερους από 61 εκατομμύρια επισκέπτες τον μήνα. Είναι κυρίως διαδεδομένο στην Ασία από όπου έχει και περισσότερο από το 90% των επισκεπτών του. Το Friendster υπήρχε πριν το MySpace και το Facebook. Σκοπός του είναι να επιτρέπει τους χρήστες να επικοινωνούν με άλλα μέλη με έναν τρόπο που θα συντηρούνται αυτές οι επικοινωνίες και να μοιράζονται εικόνες, βίντεο, μουσική. Ένας ακόμη σκοπός του είναι για καινούριες γνωριμίες οι οποίες μπορεί να καταλήξουν ακόμη και σε μια συναισθηματική σχέση. Επίσης χρησιμοποιείται για την ανεύρεση νέων γεγονότων, δραστηριοτήτων, χόμπι, καλλιτεχνικών δρώμενων και μουσικών συγκροτημάτων. Το Friendster όπως και τα περισσότερα από τα κοινωνικά δίκτυα που περιγράψαμε επιτρέπουν στους software developers να έχουν πρόσβαση σε ένα ανοιχτό API ούτως ώστε να προγραμματίσουν εφαρμογές που μπορούν να ενσωματωθούν.

2 Υλοποίηση ενός Κοινωνικού Δικτύου

Κοινωνικά δίκτυα όπως αυτά που αναφέραμε παραπάνω υλοποιούνται προγραμματιστικά με έναν συνδυασμό Linux, Apache, MySQL και PHP/Python/Perl. Πολλές φορές χρησιμοποιείται το ακρωνύμιο LAMP για να υποδηλώσει αυτόν τον συνδυασμό. Για να το δούμε λίγο πιο αναλυτικά Το Linux είναι ένα Unix-οειδές λειτουργικό σύστημα, το οποίο είναι ανοιχτού κώδικα, ασφαλές, αρκετά παραμετροποιήσιμο και προσαρμόσιμο στις εκάστοτε ανάγκες. Οπότε συνήθως τα Κοινωνικά δίκτυα τρέχουν τον Apache HTTP Server που είναι εγκατεστημένος σε περιβάλλον Linux. Ο Appache είναι ο επικρατέστερος web server. Είναι δωρεάν και ανοιχτού κώδικα.

Τα περισσότερα δεδομένα που μεταχειριζόμαστε στα κοινωνικά δίκτυα βρίσκονται σε βάσεις δεδομένων σχεδόν πάντα την MySQL. Ο λόγος που επιλέγεται η MySQL είναι ότι είναι γρήγορη και αξιόπιστη. Κάθε δεδομένο έχει ένα global ID για να βρίσκεται ανά πάσα στιγμή. Τα δεδομένα αρχειοθετούνται σε ένα σχήμα με βάση το πόσο συχνά χρειάζονται και το πόσο πρόσφατα είναι.

Η πιο πολύ χρησιμοποιημένη προγραμματιστική γλώσσα που έχει χρησιμοποιηθεί στο Facebook καθώς και σε άλλα κοινωνικά δίκτυα είναι η PHP. Η PHP είναι μια δυναμική τύπου typed/interpreted γλώσσα σεναρίων και είναι πολύ καλή όταν χρειαζόμαστε πολλές και γρήγορες επαναλήψεις.

Σε πολλά κοινωνικά δίκτυα όπως το Facebook, το Twitter άλλα και άλλα μεγάλα sites όπως το YouTube χρησιμοποιείται ένα σύστημα χρησιμοποίησης της μνήμης (memory caching) για να επιταχύνουν την δυναμική ανάκτηση δεδομένων. Το σύστημα αυτό ονομάζεται Memcache και βασίζεται στην ιδέα του να αποθηκεύουν δεδομένα και να τα ανακτούν στην RAM ούτως ώστε να μειώσουν τον χρόνο ανάγνωσης σε σχέση με το να προσπελαύνουμε απλά τις βάσεις δεδομένων. Το memcache λειτουργεί με το να έχει ένα πολύ μεγάλο πίνακα κατακερματισμού (hash table). Όταν ο πίνακας γεμίζει, τα τελευταία δεδομένα που έχουν χρησιμοποιηθεί, όταν ξανά χρειαστούν θα προσπελαστούν από πιο αργές μορφές αποθήκευσης όπως οι βάσεις δεδομένων.

Το memcache σύστημα έχει σχεδιαστεί με client - server αρχιτεκτονική Οι servers συντηρούν - έχουν έναν πίνακα από μοναδικά κλειδιά (keys) που αντιστοιχεί το καθένα σε κάποια δεδομένα μεγέθους το πολύ ένα megabyte. Το μέγεθος των κλειδιών είναι έως 250 bytes. Κάθε client γνωρίζει και μπορεί να επικοινωνήσει με κάθε server. Όταν ένας client επιθυμεί να προσπελάσει για να διαβάσει ή να γράψει κάποια δεδομένα που αντιστοιχούν σε ένα συγκεκριμένο κλειδί χρησιμοποιεί πρώτα μια βιβλιοθήκη για να υπολογίσει σε ποίον server θα απευθυνθεί. Έπειτα ο server θα βρεί σε πιο σημείο της RAM του θα αποθηκεύσει ή θα διαβάσει τα δεδομένα που δεικτοδοτεί το κλειδί.

Αυτή την στιγμή υπάρχουν πολλές πλατφόρμες που προσφέρουν έναν αρκετά αυτοματοποιημένο τρόπο για την δημιουργία online κοινωνικών δικτύων του είδους do it your self. Κάποιες από αυτές προσφέρουν online επιλογές για την δημιουργία, τις δυνατότητες, την εμφάνιση καθώς επίσης παρέχουν και φιλοξενία του κοινωνικού δικτύου. Από τις πιο γνωστές πλατφόρμες είναι οι Ning, KickApps, CrowdVine, GoingOn, CollectiveX, Me.com, PeopleAggregator, Haystack και ONEsite. Κάποιες άλλες πλατφόρμες μοιάζουν με την προηγούμενη κατηγορία εύκολα δημιουργούνται και παραμετροποιούνται από τον ενδιαφερόμενο αλλά χρειάζονται να εγκατασταθούν και να φιλοξενηθούν σε έναν server. Τέλος υπάρχουν εταιρίες που αναλαμβάνουν την δημιουργία κοινωνικών δικτύων ακούγοντας αναλυτικά τις απαιτήσεις των πελατών και ικανοποιώνταστες ακόμη και αν πρέπει να δημιουργήσουν κάτι προγραμματιστικά από την αρχή. Σίγουρα αν θέλουμε να έχουμε ένα κοινωνικό δίκτυο προσαρμοσμένο στις δικές μας απαιτήσεις, με περισσότερη ασφάλεια και αξιοπιστία άλλα και αντίστοιχο κόστος για την δημιουργία του θα πρέπει να απευθυνθούμε σε μια εταιρία όπως οι Affinity Circles, AlstraSoft, Blogtronix, Boonex, Broadband Mechanics, Converdge, Crowd Factory,DZOIC, GoLightly, introNetworks, Kwiqq, Leverage, Lithium, LiveWorld, Neighborhood America,Omnifuse, Pringo, Prospero, SelectMinds, Small World Labs, Social Platform, Sparta Social Networks,Telligent, ThePort, VMIX Media, Web Crossing, Web Scribble Solutions και η Webligo. Η παράθεση όλων αυτών των εταιριών έγινε για να φανεί η μεγάλη ζήτηση που υπάρχει στο να κατασκευαστούν καλά κοινωνικά δίκτυα.

Τα περισσότερα κοινωνικά δίκτυα προσφέρουν ένα ανοιχτό interface για να μπορεί όποιος ενδιαφερόμενος προγραμματιστής θέλει, να γράφει και να τρέχει εφαρμογές για αυτά. Πολύ διαδεδομένο είναι το OpenSocial σύνολο από API (Application Programming Interfaces) που έχει δημιουργηθεί τον Νοέμβριο του 2007 από την Google σε συνεργασία με πολλά κοινωνικά δίκτυα όπως το MySpace. Βασίζεται στην HTML, την JavaScript και τα Google Gadgets. Το OpenSocial έχει τα εξής τέσσερα Api: ένα γενικό API για JavaScript, ένα για ανθρώπους, σχέσεις, πληροφορίες για ανθρώπους και φίλους. Έπειτα ένα για τις δραστηριότητες που μπορεί να έχει ένας χρήστης όπως το να δημοσιοποιεί και να έχει πρόσβαση σε δεδομένα. Τέλος ένα για την αντιστοίχηση των δεδομένων που έχει ένα κοινωνικό δίκτυο με μια συμβολοακολουθία κλειδί.

Ο αντίλογος στο OpenSocial έρχεται να δοθεί από το FaceBook με το Facebook Platform. Το Facebook Platform είναι και αυτό ένα σύνολο από δυνατότητες και εργαλεία που επιτρέπει σε όπιον προγραμματιστεί ενδιαφέρεται να δημιουργήσει εφαρμογές στο Facebook. Το Facebook

Platform έχει μεγάλη απήχηση. Περισσότεροι από ένα εκατομμύριο προγραμματιστές και επιχειρηματίες έχουν ασχοληθεί μαζί του και έχουν δημιουργήσει περισσότερες από 550.000 εφαρμογές. Το πιο σημαντικό κομμάτι και η καρδιά του OpenSocial είναι το Graph API που δίνει την δυνατότητα στους προγραμματιστές να διαβάζουν και να γράφουν δεδομένα στο Facebook, έχει πρόσβαση στο social graph του Facebook και σε αντικείμενα όπως χρήστες, φωτογραφίες, γεγονότα σελίδες καθώς και τις συνδέσεις αναμεταξύ τους.

Το opensocial προσφέρει κάποια κοινωνικά plugins (social plugins) που επιτρέπουν τους φίλους ενός χρήστη να δούνε τι άρεσε, τι σχολίασε και τι μοιράστηκε αυτός ο χρήστης. Τα plugins αυτά είναι.

- Comments. Επιτρέπει στους χρήστες να σχολιάσουν οποιοδήποτε μέρος ενός site.
- Facepile. Εμφανίζει της φωτογραφίες από τα προφίλ των χρηστών που τους άρεσε μια σελίδα ή έχουν γραφεί σε ένα site.
- Like Box. Επιτρέπει στους χρήστες να κάνουν Like μια σελίδα στο Facebook και να την δουν απείθειας από μιαν άλλη ιστοσελίδα.
- Like Button. Επιτρέπει στους χρήστες να προβάλουν μια ιστοσελίδα στο προφίλ τους.
- Live stream. Επιτρέπει στους χρήστες να έχουν δραστηριότητες και σχόλια σε πραγματικό χρόνο.
- Login Button. Παρέχει ένα Login button καθώς και εμφανίζει της φωτογραφίες από τους φίλους του χρήστη που έχουν ήδη εγγραφεί στο site αυτό.
- Recommendations. Μπορεί να κάνει εξατομικευμένες συστάσεις για σελίδες σε κάποιον χρήστη από ένα site.
- Registration. Επιτρέπει στους χρήστες να εγγράφονται σε μια ιστοσελίδα χρησιμοποιώντας τον λογαριασμό τους στο Facebook.
- Activity Feed. Εμφανίζει στους χρήστες μέσα από ένα site το τι likes και σχόλια έχουν κάνει οι φίλοι τους.

Άλλες δυνατότητες που έχει το Facebook Platform είναι το Open Graph Protocol που

επιτρέπει στους προγραμματιστές να ενσωματώσουν τις σελίδες τους στο Social graph. Η Facebook Mark Language (FMBL) πού είναι μια γλώσσα σήμανσης, υποσύνολο της HTML την οποία μπορούν να διαβάσουν και να κοινοποιήσουν οι server του Facebook. Το Facebook Connect είναι ένα σύνολο από API που επιτρέπουν τα μέλη του Facebook να κάνουν Log in από άλλα site, εφαρμογές, κινητές συσκευές και παιχνίδια.

Βιβλιογραφία:

[http://developers.facebook.com/docs/plugins/]

3 Διάφορες τεχνολογίες που χρησιμοποιούν τα Κοινωνικά Δίκτυα

Τα κοινωνικά δίκτυα συνδυάζουν ένα σύνολο από τεχνολογίες, επιστημονικές ενότητες και τεχνικές ούτως ώστε να είναι αποδοτικά και να μας προσφέρουν τις βασικές τους υπηρεσίες. Πολλές φορές πίσω από συνηθισμένες εφαρμογές κρύβονται πολύπλοκοι αλγόριθμοι και πίσω από τους αλγόριθμους ολόκληρα επιστημονικά πεδία που μελετούν ποιος είναι ο πιο αποδοτικός τρόπος για να έχουμε τα καλύτερα αποτελέσματα. Ακόμη και μια απλή αναζήτηση που μπορεί να φαίνεται ότι πιο συνηθισμένο και τετριμμένο για έναν χρήστη κρύβει πολλές επιστημονικές μελέτες, έρευνα και ώρες εργασίας για τον τρόπο που θα υλοποιηθεί. Αν και οι εφαρμογές που χρησιμοποιούνται είναι πολλές και διαφέρουν από κοινωνικό δίκτυο σε κοινωνικό δίκτυο όπως και οι τρόποι υλοποίησης του. Θα αναφέρουμε συνοπτικά κάποιους από αυτούς και έπειτα θα αναλύσουμε εκτενέστερα αυτούς που μας ενδιαφέρουν περισσότερο.

3.1 Social Search Engine

Είναι ένας τύπος αναζήτησης που βασίζεται στον κοινωνικό γράφο (social graph) του ατόμου που κάνει την αναζήτηση. Η βασική ιδέα είναι ότι δεδομένα από φίλους και επαφές που έχουμε στο κοινωνικό μας δίκτυο θα είναι πιο σημαντικά από άλλα. Αυτά θα εμφανιστούν με μεγαλύτερη προτεραιότητα από ότι δεδομένα από ανθρώπους που δεν τους γνωρίζουμε. Τα δεδομένα που χρησιμοποιούνται είναι του είδους metadata, tags, social ranking, σχόλια, νέα, εικόνες βίντεο knowledge sharing και bookmarks.

3.2 Rating Features

Σχεδόν πάντα τα κοινωνικά δίκτυα προσφέρουν την δυνατότητα στους χρήστες τους να βαθμολογούν κατά πόσο τους αρέσει μια εικόνα, ένα κείμενο, ένα σχόλιο, ένας σύνδεσμος ή ένα βίντεο. Αυτή η βαθμολόγηση μπορεί να γίνει απλά με το πάτημα ενός Like και την εμφάνιση του πόσα Like είχε αυτό πους μας ενδιαφέρει. Ή πιο σύνθετα όπως η ψηφοφορία με μηδέν έως πέντε αστεράκια όπου εμφανίζεται ο μέσος όρος όλον αυτόν που ψήφισαν. Αν και η διαδικασία για το rating είναι πολύ απλή τα δεδομένα, η επεξεργασία των δεδομένων και τα συμπεράσματα που μπορούμε να έχουμε μπορεί να είναι αρκετά περίπλοκα και με πολλές χρησιμότητες.

3.3 Collaborative Tagging

Είναι ένα σύστημα για αυτόματη ταξινόμηση, διαχείριση και δημιουργία ετικετών (tags) από απλούς χρήστες. Κάθε χρήστης ενός κοινωνικού δικτύου μπορεί στα δεδομένα που ανεβάζει ή βλέπει να βάζει ένα tag το οποίο να προσδιορίζει την κατηγορία στην οποία ανήκουν. Κάθε φωτογραφία, μουσικό κομμάτι, σχόλιο, κείμενο, σύνδεσμος μπορεί να συνοδεύεται από μια ετικέτα που να προσδιορίζει το περιεχόμενο του. Έπειτα αναζητήσεις και συσχετίσεις μεταξύ των δεδομένων μπορούν να γίνουν με βάση αυτές τις ετικέτες. Το Collaborative tagging είναι μια σημαντικότατη μέθοδος για την υλοποίηση του σημασιολογικού ιστού και την μετάβαση στο web 3, στο οποίο κάθε περιεχόμενο μιας ιστοσελίδας θα έχει μεταδεδομένα που θα μπορούν να διαβαστούν από τον υπολογιστή (machine-readable metadata).

3.4 Social Bookmarking

Είναι μια μέθοδος για να αποθηκεύουν, να διαχειρίζονται και να μοιράζονται bookmarks. Sites που ενδιαφέρουν χρήστες και δεν θέλουν να χάσουν το URL τους ούτως ώστε είτε να τα ξανά επισκεφτούν είτε να ενημερώσουν τους φίλους τους για αυτά, μπορούν να σωθούν και να δημοσιοποιηθούν μέσω ενός κοινωνικού δικτύου. Έπειτα όποιος ενδιαφέρεται μπορεί να δει τον σύνδεσμο, να ενημερώσει κατά πόσο του αρέσει και να το σχολιάσει. Μέσω των bookmark που έχει ένας χρήστης μπορούμε να καταλάβουμε τα ενδιαφέροντα του, με ποιους άλλους χρήστες σχετίζεται ακόμη και οι διάφορες ιστοσελίδες που έχουν γίνει bookmark σε ποιους χρήστες

απευθύνονται.

3.5 Social Information Processing

Πολλοί χρήστες γίνονται μέλη σε ένα group ενός κοινωνικό δίκτυο όπου μοιράζονται κοινά ενδιαφέροντα, συγκεντρώνουν και οργανώνουν γνώση γύρω από τα θέματα που τους ενώνουν. Οι συζητήσεις, οι δημοσιεύσεις, οι ψηφοφορίες και γενικώς οι δραστηριότητες που μπορούν να έχουν οι χρήστες ενός group μπορούν να είναι δεδομένα με τα οποία θα παραχθεί γνώση.

3.6 Customer Engagement

Έχει να κάνει με την αλληλεπίδραση πελατών αναμεταξύ τους ή την αλληλεπίδραση πελατών με μια εταιρία. Τα κοινωνικά δίκτυα μπορούν να προσφέρουν το περιβάλλον σε καταναλωτές και εταιρίες ούτως ώστε οι καταναλωτές να ενημερώνονται για τα διάφορα προϊόντα που τους ενδιαφέρουν και οι εταιρείες να ενημερώνονται από τους καταναλωτές για την άποψη που έχουν για τα προϊόντα τους καθώς και για τις ανάγκες της αγοράς που μπορούν να καλύψουν.

3.7 Personalized Marketing

Ο σκοπός του Personalized marketing σε ένα κοινωνικό δίκτυο είναι να προβάλει στον κάθε χρήστη το προϊόν που έχει πιο πολύ ανάγκη και του ταιριάζει σύμφωνα με τις προσωπικές του προτιμήσεις και τον χαρακτήρα του. Ένα κοινωνικό δίκτυο μπορεί να έχει πολλά δεδομένα και πληροφορίες για κάθε χρήστη ούτως ώστε να ξέρει της ανάγκες του και πια προϊόντα και υπηρεσίες μπορούν να του ταιριάζουν.

3.8 Reputation System

Υπολογίζει για ένα αντικείμενο, χρήστη, υπηρεσία, εταιρία, προϊόν και γενικώς για οποιαδήποτε πληροφορία και δεδομένα την υπόληψη που υπάρχει για αυτά. Δηλαδή είναι ένα μέτρο αξιοπιστίας μέσα στην κοινότητα ενός κοινωνικού δικτύου. Οι απόψεις για ένα αντικείμενο, που μπορούν να είναι οποιασδήποτε μορφής, περνάνε σαν βαθμοί ψηφοφορίας από ένα κέντρο όπου ένας κατάλληλος αλγόριθμος (reputation algorithm) υπολογίζει δυναμικά την φήμη - υπόληψη του.

Κάποια Reputation systems βασίζονται σε άμεσες ενέργειες χρηστών όπως ψηφοφορίες, άλλα βασίζονται στην μελέτη των ενεργειών και της συμπεριφοράς του χρήστη μέσα στο κοινωνικό δίκτυο. Οι πράξεις του αυτές μεταφράζονται έμμεσα σε τιμές που αντιστοιχούν με την υπόληψη που υπάρχει για τα διάφορα αντικείμενα και χρήστες.

Θα μπορούσαμε να χωρίσουμε τα Reputation systems σε πέντε κατηγορίες: ranking systems, rating systems και collaborative filtering systems, Implicit Peer-based, Explicit Peer-based.

- Στα ranking systems μετράμε κατά πόσο στα μέλη της κοινότητας αλληλεπιδρούν με έναν χρήστη. Δεδομένα που παίρνουμε υπόψιν μας είναι το πλίθος των μελών που είναι συσχετισμένα με τον χρήστη, η συχνότητα των επισκέψεων στα δεδομένα που έχει δημοσιεύσει ο χρήστης, πόσες πολλές απαντήσεις - κριτικές έχει λάβει σε σχέση με το πόσα καινούρια θέματα έχει δημοσιεύσει.

- Στα Rating systems υπολογίζουμε πόσο πολύ και για πόσο καιρό ένα αντικείμενο ή ένας άνθρωπος είναι μέρος μιας ομάδας. Αυτό μπορεί να προκύψει ρητά από τις ψηφοφορίες των χρηστών. Για κάθε αντικείμενο που μας ενδιαφέρει έχουμε μια σταθμισμένη τιμή η οποία προκύπτει είτε από ένα σύνολο ψήφων είτε από την αναλογία θετικών έναντι αρνητικών κριτικών.

- Στα colaborative filtering systems μετράμε κατά πόσο τα ενδιαφέροντα και οι ασχολίες ενός χρήστη ταιριάζουν με τα ενδιαφέροντα και της ασχολίες ενός συγκεκριμένου προσώπου. Αυτό μπορεί να μετρηθεί με το κατά πόσο τα δύο πρόσωπα που μελετάμε έχουν κοινές απόψεις και προτιμήσεις για διάφορα θέματα και αντικείμενα.

- Στα Implicit Peer-based systems μελετάμε πόσο συχνά ένας χρήστης αλληλεπιδρά με έναν ή περισσότερους φίλους του. Τα Implicit Peer-based systems βασίζονται στην αρχή ότι οι ταινίες που βλέπουμε, τα προϊόντα που αγοράζουμε και γενικώς πολλές από τις πράξεις

μας επηρεάζονται και καθοδηγούνται κατά ένα μεγάλο ποσοστό από το τι μας έχουν προτείνει οι φίλοι μας. Οπότε το σύστημα αυτό παρατηρεί το τι κάνουν οι φίλοι μας και μας προτείνει αντίστοιχες δραστηριότητες

- Στα Explicit Peer-based systems βλέπουμε πάλι μια αντίστοιχη συμπεριφορά, μόνο που εδώ η επιλογή των φίλων και αυτών που εκτιμούμε την άποψή τους γίνεται ρητά. Οι χρήστες διαλέγουν ομάδες από φίλους ή άτομα που εμπιστεύονται την γνώμη τους και προβάλλονται οι αντίστοιχες συστάσεις.

  [papper: Finding others Online: Reputation Systems for Social Online Spaces]

3.9 Automatic Summarization

Σε ένα κοινωνικό δίκτυο που έχει εκατομμύρια χρήστες και μπορεί να έχει στα δεδομένα του πολλά εκατομμύρια κείμενα θα ήταν πολύ χρήσιμο αν μπορούσαμε με έναν αυτοματοποιημένο τρόπο να έχουμε μια περίληψη κάθε κειμένου χωρίς να χρειάζεται κάποιος άνθρωπος να τα διαβάσει. Αυτό έρχεται να εκπληρώσει το Automatic summarization. Την δημιουργία μιας πιο σύντομης εκδοχής ενός κειμένου από ένα πρόγραμμα. Αυτή η σύντομη εκδοχή θα διατηρεί τα ποιο σημαντικά σημεία του αρχικού κειμένου.

Μια μέθοδος που μπορούμε να ακολουθήσουμε για να δημιουργήσουμε την περίληψή μας είναι να διαλέξουμε ένα υποσύνολο από τις λέξεις, φράσεις και προτάσεις που έχει το αρχικό κείμενο, να τα συνθέσουμε και με αυτά να σχηματίσουμε την περίληψη. Εννοείται πως με κάποιο τρόπο θα έχουμε διαλέξει τις πιο σημαντικές και αντιπροσωπευτικές λέξεις, φράσεις και προτάσεις. Αυτές οι μέθοδοι ονομάζονται extractive methods.

Ποιο προχωρημένες αφαιρετικές μέθοδοι λειτουργούν με το να δημιουργήσουν αρχικά μια σημασιολογική αναπαράσταση του νοήματος του κειμένου και έπειτα με βάση αυτήν την αναπαράσταση να παράγουν μια περίληψη. Με αυτόν τον τρόπο χρησιμοποιούμε λέξεις και εκφράσεις που μπορεί να μην υπάρχουν στο αρχικό κείμενο αλλά μπορούν να εκφράσουν το νόημα του. Αφαιρετικές μέθοδοι σαν αυτές είναι αρκετά πιο δύσκολες από τις extractive methods για αυτό δεν χρησιμοποιούνται τόσο συχνά. Σημεία που πρέπει να προσέξουμε κατά την δημιουργία της περίληψης μας είναι είναι το μήκος, το στυλ γραφής και το συντακτικό που θα χρησιμοποιήσουμε.

3.10 Multi-Document Summarization

Είναι η διαδικασία κατά την οποία παράγουμε ένα συνοπτικό και περιεκτικό κείμενο που εσωκλείει τα βασικά σημεία και τις πληροφορίες από πολλά κείμενα που είναι γραμμένα γύρω από το ίδιο θέμα. Σκοπός του multi-document summarization είναι να δώσει την δυνατότητα στον αναγνώστη του πολύ σύντομα να μάθει και να εξοικειωθεί με το περιεχόμενο πολλών κειμένων. Τεχνικές όπως η multi-document summarization καθώς και η automatic summarization επιτρέπουν στο κοινωνικό δίκτυο να μην αντιλαμβάνεται ένα κείμενο στατικά απλώς ως μια ακολουθία χαρακτήρων που συντάσσει ένας χρήστης. Αλλά να δίνει την δυνατότητα να βγάζει συμπεράσματα για το τι γράφει και με το τι ασχολείται κάθε χρήστης. Οτιδήποτε γράφει ένας χρήστης μπορεί να γίνει μια πηγή γνώσης. Το multi-document summarization είναι ένα σύστημα κατανόησης και επεξεργασίας του κοινωνικού γίγνεσθαι.

Η δημιουργία ενός multi-document summarization κειμένου είναι πολύ πιο δύσκολη από ένα Automatic summarization κείμενο όχι μόνο γιατί ο όγκος των κειμένων είναι πολύ μεγαλύτερος αλλά και γιατί σίγουρα θα υπάρχουν έστω και μικρές θεματικές αποκλίσεις μεταξύ των κειμένων. Σκοπός πάντα είναι να συνδυαστούν τα κύρια θέματα ολοκληρωμένα, να είναι αναγνώσιμα και περιεκτικά.

Μια Multi-document περίληψη για να είναι πετυχημένη πρέπει να ικανοποιεί όσο το δυνατόν περισσότερο τα έξεις κριτήρια:

- Να έχει μια καθαρή δομή η οποία να δίνει την δυνατότητα στον αναγνώστη να περιηγείται εύκολα μέσα στην περίληψη για να βρίσκει το κομμάτι ακριβώς που τον ενδιαφέρει.
- Το κείμενο να είναι διαιρεμένο σε σωστές παραγράφους.
- Να υπάρχει σταδιακή μετάβαση από πιο γενικές σε πιο συγκεκριμένες θεματικές ενότητες.
- Το κείμενο να διακρίνεται από εύκολη αναγνωσιμότητα.
- Να μην υπάρχουν άσχετα κομμάτια πληροφορίας που δεν ταιριάζουν με το υπόλοιπο κείμενο.
- Να μην επαναλαμβάνονται οι ίδιες πληροφορίες.
- Να υπάρχει σαφής και ξεκάθαρη έκφρασή του τι θέλει να ειπωθεί κάθε φορά.

3.11 Recommendation Systems.

Είναι συστήματα τα οποία κάνουν συστάσεις για διάφορα αντικείμενα και θέματα όπως βιβλία, άρθρα στο διαδίκτυο, καταναλωτικά προϊόντα, φίλοι σε ένα κοινωνικό δίκτυο, εικόνες και γενικώς οτιδήποτε θα μπορούσε να ενδιαφέρει έναν χρήστη και θα ήθελε να ενημερωθεί για αυτό. Κυρίως θα αναφερθούμε σε καταναλωτικά είδη αλλά είναι ίδιες οι μέθοδοι που θα προταθούν και για τα υπόλοιπα αντικείμενα. Οτιδήποτε είναι υποψήφιο να συσταθεί θα το αποκαλούμε συχνά απλά ως αντικείμενο. Ένας ορισμός που θα μπορούσε να γραφθεί για το τι είναι ένα Recommendation system είναι ο ακόλουθος.

Έστω I={i| "i" είναι ένα αντικείμενο} είναι το σύνολο των αντικειμένων που παρουσιάστηκαν σε έναν ιστότοπο. x ένας χρήστης που περιηγήθηκε και αλληλεπίδρασε με τον ιστότοπο αυτό. Το Recommendation system ορίζετε ως το σύστημα που θα βρει μι βαθμολογημένη λίστα από αντικείμενα I_x. Η οποία ονομάζεται λίστα επιθυμίας (wish-list), στην οποία τα αντικείμενα I_x έχουν βαθμολογηθεί σύμφωνα με το ενδιαφέρον που έχει ο χρήστης x για αυτά.

Recommendation systems συναντάμε συνέχεια στην περιήγηση μας σε ένα κοινωνικό δίκτυο καθώς και στον παγκόσμιο ιστό. Πίσω από κάθε διαφήμιση που μας προβάλει το google, πίσω από κάθε προϊόν που μας συστήνει το amazon και το ebay, πίσω από κάθε εικόνα, άρθρο, φίλο, δραστηριότητα, σύνδεσμο που μας προτείνει το facebook βρίσκεται ένα recommendation system όπου πολύ προσεκτικά μελέτησε της κινήσεις μας, μας συσχέτισε με άλλους χρήστες και μας πρόβαλε μια σύσταση που πιστεύει ότι μας αρμόζει περισσότερο.

Σκοπός ενός recommendation system είναι να διαχειρίζεται τον μεγάλο όγκο πληροφοριών και δυνατοτήτων που υπάρχουν ούτως ώστε να βοηθήσει έναν χρήστη στο να επιλέξει αυτό που τον ενδιαφέρει.

Οι ερευνητές εξερευνούν ένα μεγάλο εύρος από χρήσεις και εφαρμογές των recommendation συστημάτων που ξεπερνούν κατά πολύ τις τωρινές μας προσδοκίες. Έξυπνοι τουριστικοί οδηγοί καθώς και οδηγοί για εξόδους μπορούν να μας συστήνουν επισκέψεις και βόλτες προσαρμοσμένες στις προτιμήσεις μας. Συστήματα πλοήγησης αυτοκινήτων μπορούν να λαμβάνουν υπόψιν τους ποιες διαδρομές θα μας είναι ποιο ευχάριστες και να τις προτείνουν. Σε ένα ποιο αφαιρετικό επίπεδο θα μπορούσε να ειπωθεί ότι, οποιαδήποτε συσκευή χρησιμοποιείται

από ένα σύνολο ανθρώπων και υπάρχει η δυνατότητα καθένας να την χρησιμοποιεί με διαφορετικό τρόπο ανάλογα με την προσωπικότητα του, θα μπορούσε να του προσφέρονται συστάσεις από ένα recommendation σύστημα με τους τρόπους χρήσης που θα του ταιριάζουν καλύτερα. Φανταστείτε κάθε φορά που αγοράζεται ένα τρόφιμο από το super market, ένα recommendation system να σας προτείνει τις συνταγές μαγειρικής που σας αρέσουν περισσότερο και το περιλαμβάνουν ως συστατικό. Ένα recommendation σύστημα μπορεί να χρησιμοποιηθεί ως βοήθεια σε οποιουδήποτε είδους επιλογή.

Αυτό που κάνει συνήθως ένα recommendation system είναι να συγκρίνει τα στοιχεία από το προφίλ ενός χρήστη με κάποια στοιχεία αναφοράς. Σκοπός αυτής της σύγκρισης είναι να μπορεί να προβλεφθεί το ενδιαφέρον που θα είχε ο χρήστης για ένα αντικείμενο το οποίο ακόμη δεν έχει δει. Τα δεδομένα πάνω στα οποία θα βασιστεί το recommendation system μπορούν να προέρχονται είτε από το ίδιο το αντικείμενο είτε από το κοινωνικό περιβάλλον που βρίσκεται ο χρήστης.

Τα δεδομένα για να φτιάξουμε το προφίλ του χρήστη χωρίζονται σε δύο μεγάλες κατηγορίες. Τα δεδομένα που μας δίνει ο χρήστης άμεσα και αυτά που συλλέγουμε έμμεσα. Άμεσα δεδομένα μπορεί να μας δώσει ο χρήστης με πολλούς τρόπους όπως:

- Να του ζητήσουμε να βαθμονομήσει – ψηφίσει ένα δείγμα αντικειμένων σε μια προκαθορισμένη κλίμακα.
- Να του ζητήσουμε να κατατάξει ένα δείγμα αντικειμένων με σειρά ανάλογη του πόσο του αρέσει το κάθε ένα.
- Να του παρουσιάσουμε μια σειρά αντικειμένων και να επιλέξει ένα.
- Να του ζητήσουμε να δημιουργήσει μια λίστα με όλα τα αντικείμενα που του αρέσουν.

Έμμεσα μπορούμε να συλλέξουμε δεδομένα από έναν χρήστη ακολουθώντας κάποιες από τις επόμενες μεθόδους:

- Να παρακολουθήσουμε όλα τα αντικείμενα που ένας χρήστης είδε στον υπολογιστή του.
- Να συγκρίνουμε τους χρόνους που έκανε ένας χρήστης βλέποντας κάθε αντικείμενο.
- Να κρατήσουμε αρχείο με τα αντικείμενα που ένας χρήστης αγόρασε από ένα online κατάστημα.

- Να δημιουργήσουμε μια λίστα με τα αντικείμενα έχει επιλέξει ο χρήστης ως επιθυμητά ή αγαπημένα.

- Να αναλύσουμε ενός χρήστη της επαφές και της δραστηριότητες που έχει στο κοινωνικό δίκτυο και με αυτόν τον τρόπο να ανακαλύψουμε τι του αρέσει και τι όχι.

Τα δεδομένα που έχουμε μαζέψει με έναν από τους προηγούμενους τρόπους, τα recommendation system τα επεξεργάζονται και τα συγκρίνουν μαζί με άλλα δεδομένα που έχουν μαζευτεί από άλλους χρήστες με σκοπό να δημιουργήσουν μια λίστα με αντικείμενα που θα συστήσουν στον χρήστη.

Υπάρχουν διάφοροι παράγοντες που μπορούν να επηρεάσουν τον βαθμό που ενδιαφέρεται ένας χρήστης για ένα αντικείμενο που συστήνεται και πρέπει γενικώς να τους λάβουμε υπόψιν μας.

- Η σχέση μεταξύ του προσώπου που προσφέρει μια σύσταση με αυτόν που λαμβάνει την σύσταση μπορεί να παίξει καθοριστικό παράγοντα στην κρίση και την προτίμηση για την σύσταση αυτή.

- Οι αποφάσεις που παίρνει ένας χρήστης για το ποιες συστάσεις θα επιλέξει μπορούν να επηρεαστούν από αντικειμενικούς παράγοντες (επαγγελματικές ανάγκες) ή υποκειμενικούς παράγοντες (αισθητική, προσωπικές προτιμήσεις).

- Προηγούμενη εμπειρία στο παρελθόν ενός χρήστη με κάποιον που του έχει κάνει συστάσεις μπορεί να έχει καθοριστικό ρόλο στο πως θα εκλάβει και το αν θα είναι θετικά ή αρνητικά προκατειλημμένος τις επόμενες συστάσεις που θα του κάνει.

- Οι απόψεις της κοινής γνώμης και των άλλων χρηστών επηρεάζουν πολύ την κρίση ενός χρήστη και ως προς το αν θα ακολουθήσει μια συγκεκριμένη σύσταση και για το αν θα εμπιστευτεί να ακολουθεί ένα σύστημα συστάσεων.

- Αν το άτομο ή το σύστημα που κάνουν τις συστάσεις έχουν κάποια προσωπικά κίνητρα αυτό θα έχει μια αρνητική επίπτωση στην ποιότητα των συστάσεων που θα γίνει.

Το σύστημα συστάσεων θα πρέπει να ισορροπήσει μεταξύ του να έχει μια διακριτική παρουσία ούτως ώστε να μην κουράζει των χρήστη με καταιγισμό πληροφοριών και ταυτόχρονα

οι συστάσεις να παρουσιάζονται με έναν εμφανές και ελκυστικό τρόπο.

Σε μια σύσταση που προέρχεται από έναν φίλο του χρήστη θα δοθεί περισσότερη σημασία από ότι μια σύσταση που προέρχεται αδιαφανώς από ένα recommendation system ακόμη και αν η δεύτερη είναι πιο ακριβής.

Η σχέση των κοινωνικών δικτύων με το recommendation system είναι μεγάλη. Το ένα επηρεάζεται και επωφελείται από το άλλο με σκοπό να αυξήσουν την συμμετοχή και την συμβολή των χρηστών καθώς και την ποιότητα των πληροφοριών που ανταλλάσσονται. Μια σειρά από θέματα που είναι ακόμη ανοιχτά και παρουσιάζουν μεγάλο ενδιαφέρον προκύπτουν από την συνύπαρξη αυτών των δύο. Θέματα μείζονος σημασίας είναι:

- Η εμπιστοσύνη στις συστάσεις που δέχεται ο χρήστης θα αυξηθεί αν γνωρίζει ότι αυτές παράγονται με βάση το κοινωνικό δίκτυο των προσωπικών φίλων του;

- Κατά πόσο αξίζει οι χρήστες που δεν γνωρίζονται και ταιριάζουν τα προφίλ τους σύμφωνα με κάποιο recommendation system, να τους φέρουμε σε επαφή για να αλληλογνωριστούν, με την έννοια ότι έχουν παρόμοια ενδιαφέροντα, ασχολίες και προσωπικότητα;

- Ποία είναι τα κριτήρια με τα οποία μπορούμε να πούμε ότι κάποιοι χρήστες ταιριάζουν αναμεταξύ τους και πως μπορούμε να χρησιμοποιήσουμε την σχέση αυτή για να παράγουμε συστάσεις;

- Η γνώση για το ότι οι κινήσεις των χρηστών μέσα σε ένα κοινωνικό δίκτυο αποθηκεύονται και επεξεργάζονται με σκοπό να γίνουν συστάσεις μπορεί να εγείρει αντιρρήσεις λόγο του ιδιωτικού απόρρητου και να απομακρυνθούν από το κοινωνικό δίκτυο.

Θα μπορούσαμε να κατατάξουμε τις βασικές κατηγορίες στις όποιες βασίζονται οι αρχιτεκτονικές σχεδιασμού ενός recommendation system σε τρεις. Στα Content based filtering, Collaborative filtering και link-based συστήματα. Οι περισσότερες υλοποιήσεις ενός recommendation συστήματος στηρίζονται σε μια από αυτές ή είναι υβρίδια που τις συνδυάζουν.

- Τα Content based filtering συστήματα αξιοποιούν τεχνικές μηχανικής μάθησης για να αναλύσουν ιστοσελίδες, ειδήσεις από το usenet, e-mail και γενικώς οποιουδήποτε τύπου ηλεκτρονικού περιεχομένου πληροφορία επιδέχεται αυτοματοποιημένη λεκτική ανάλυση.

- Τα Collaborative filtering συστήματα βασίζονται στην βαθμολόγηση των αντικειμένων από χρήστες. Έπειτα χρησιμοποιούν την υπόθεση ότι οι χρήστες θα ενδιαφέροντα και στο μέλλον για παρόμοια αντικείμενα με αυτά που βαθμολόγησαν υψηλά.

- Τα Link-based συστήματα ανακαλύπτουν σχέσεις μεταξύ αντικειμένων. Έπειτα χρησιμοποιούν γραφοθεωρητικούς αλγόριθμους για να βρουν τα αντικείμενα όπου είναι υποδειγματικά και σχετίζονται περισσότερο με άλλα αντικείμενα.

Παρακάτω θα δούμε recommendation systems που υλοποιούν κάθε μία από αυτές τις κατηγορίες και θα τα αναλύσουμε πλήρως. Έπειτα θα δούμε κάποια που τις συνδυάζουν. Ακόμη θα συναντήσουμε κάποια recommendation systems που είναι πολύ ενδιαφέροντα αλλά δεν ανήκουν εμφανώς σε κάποια από αυτές τις κατηγορίες.

Βιβλιογραφία:
[Who do trust? Combining Recommender Systems and Social Networking for Better Advice από τον Philip Bonhard]
[Ubiquitous Recommendation Systems από τον David W. McDonald]
[Recommender Systems απο τους Paul Resnick και Hal R. Varian]

4 Αλγόριθμοι, Τεχνικές και Μέθοδοι που υλοποιούν Recommendation Systems.

Στην συνέχεια θα περιγράψουμε και θα αναπτύξουμε περιγραφικά διάφορους αλγόριθμους και τεχνικές για την υλοποίηση ενός recommendation system. Προφανώς ένα πρόβλημα στην επιστήμη των υπολογιστών μπορεί να έχει πολλές λύσεις. Κάθε λύση μπορεί να είναι αποτέλεσμα μιας διαφορετικής προσέγγισης. Κάθε φορά με τα δεδομένα που μπορούμε να έχουμε και τις τεχνικές για να συλλέξουμε πληροφορίες για τον εκάστοτε χρήστη και κοινωνικό δίκτυο κάποιος αλγόριθμος μπορεί να ενδείκνυται περισσότερο από κάποιον άλλον.

4.1 Παρουσίαση της Collaborative Filtering αρχιτεκτονικής

Collaborative filtering είναι μια μέθοδος για να επεξεργαζόμαστε δεδομένα με σκοπό να δημιουργήσουμε προφίλ χρηστών για τις προτιμήσεις τους. Μπορούμε έπειτα να κάνουμε προβλέψεις για τα ενδιαφέροντα ενός χρήστη συσχετίζοντας το προφίλ του με τα προφίλ άλλων χρηστών. Η βασική ιδέα στην οποία στηρίζεται αυτή η μέθοδος είναι ότι χρήστες που συμφωνήσαν για κάποια θέματα στο παρελθόν θα ξανά συμφωνήσουν για άλλα θέματα στο μέλλον.

Η μέθοδος αυτή πραγματοποιείται ως εξής. Από τις κινήσεις και επιλογές ενός χρήστη δημιουργείται ένα rating pattern το οποίο υποδηλώνει το τι του αρέσει και τι δεν του αρέσει. Η δημιουργία του rating pattern μπορεί να γίνει με κάποιο τρόπο όπως το ποια αντικείμενα επέλεξε να βάλει στο καλάθι αγορών του ή ποια αντικείμενα έκανε κλικ να τα δει.

Rating patterns έχουμε δημιουργήσει για κάθε χρήστη που έχει εισέλθει στο σύστημά μας και έχει κάνει κάποιες επιλογές. Όταν για κάποιο χρήστη θα ενδιαφερόμαστε να κάνουμε κάποια

πρόβλεψη θα συγκρίνουμε το rating pattern του με τα rating patterns των άλλων χρηστών αναζητώντας αυτά που είναι όμοια.

Το να βρούμε τα ποιο όμοια rating pattern συνεπάγεται ότι έχουμε βρει και παρόμοιους χρήστες. Μεταξύ αυτών των παρόμοιων χρηστών θα εφαρμόσουμε κάποιο κριτήριο του είδους "χρήστες που αγόρασαν το χ αγόρασαν και το ψ προϊόν". Έτσι υπολογίζουμε και κάνουμε συστάσεις για τον χρήστη που μας ενδιαφέρει.

Ένας εναλλακτικός τρόπος για να υλοποιήσουμε την μέθοδο collaborative filtering είναι να δημιουργήσουμε έναν δισδιάστατο πίνακα όπου και στις δύο διαστάσεις του να έχουμε αντικείμενα. Όταν δύο αντικείμενα συσχετίζονται αυτό θα υποδηλώνεται στο αντίστοιχο κελί έτσι θα έχουμε σχέσεις μεταξύ αντικειμένων Χρησιμοποιώντας τις σχέσεις του πινάκα και τα δεδομένα που έχουμε για τον χρήστη που μας ενδιαφέρει μπορούμε να συμπεράνουμε το τι ακόμη μπορεί να τον ενδιαφέρει

Για να είναι αξιόπιστες οι επιλογές που θα έχουμε, θα πρέπει να έχουμε έναν αρκετά μεγάλο αριθμό χρηστών, που έκαναν επιλογές μέσα στο σύστημά μας. Η μέθοδος Collaborative filtering θα μπορεί να δώσει αξιόπιστα αποτελέσματα αφού περάσουμε αυτή την κρίσιμη μάζα χρηστών.

Υπάρχουν δύο βασικοί τύποι collaborative filter ο memory based (βασισμένος στην μνήμη) και ο model based (βασισμένος στο μοντέλο)

4.1.1 Memory-Based Collaborative Filtering αρχιτεκτονική

Ο Memory-Based μηχανισμός χρησιμοποιεί τα δεδομένα που έχουν ρητά βαθμολογηθεί από τον χρήστη για να υπολογίσει την ομοιότητα που υπάρχει μεταξύ των χρηστών ή αντικειμένων Με αυτόν τον τρόπο μπορούμε να κάνουμε συστάσεις. Ποιο συγκεκριμένα υπάρχει ο neighborhood - based αλγόριθμος ο οποίος παίρνοντας υπόψιν όλες τις βαθμολογήσεις που έχουν γίνει, υπολογίζει την ομοιότητα μεταξύ δύο χρηστών ή αντικειμένων και παράγει μια πρόβλεψη. Ένας άλλος ποιο περίπλοκος αλγόριθμος είναι ο top-N recommendation ο οποίος προσδιορίζει τους N πιο όμοιους χρήστες προς τον χρήστη πους μας ενδιαφέρει, βρίσκοντας την ομοιότητα που υπάρχει μεταξύ μοντελοποιημένων διανυσμάτων που αντιπροσωπεύουν κάθε χρήστη. Έπειτα συγκρίνουμε τον πίνακα αντικειμένων που έχει κάθε ένας από αυτούς τους N - χρήστες και υπολογίζουμε το σύνολο
των αντικειμένων που θα συσταθούν.

4.1.2 Model-Based Collaborative Filtering αρχιτεκτονικής

Στην Model-Based αρχιτεκτονική αναπτύσσουμε μοντέλα χρησιμοποιώντας τεχνικές εξόρυξης δεδομένων και αλγόριθμους μηχανικής μάθησης. Μέσω αυτών μπορούμε να κάνουμε προβλέψεις για το ποιες θα ήταν οι καλύτερες συστάσεις. Τέτοια μοντέλα υπάρχουν πολλά. Ενδεικτικά κάποια από αυτά είναι Bayesian Networks, clustering models, latent semantic models καθώς επίσης και τα singular value decomposition, probabilistic latent semantic analysis, Multiple Multiplicative Factor, Latent Dirichlet allocation και το markov decision process based model. Αυτές οι μέθοδοι είναι πιο ολιστικές και καταφέρνουν να αποκαλύψουν λανθάνων και κριμένα στοιχεία. Τα περισσότερα από τα μοντέλα βασίζονται στο να δημιουργήσουν μια ταξινόμηση και ομαδοποίηση για να ανακαλύψουμε την συμπεριφορά του χρήστη με βάση τα δεδομένα που έχουμε για αυτόν. Κοινές συμπεριφορές μεταξύ χρηστών μας επιτρέπουν να κάνουμε προβλέψεις για περαιτέρω συστάσεις σε προϊόντα ή υπηρεσίες που μπορεί να μην έχουν ακόμη επιλέξει οι ίδιοι αλλά έχουν επιλέξει άτομα τα οποία βρίσκονται μαζί στην ίδια ομαδοποίηση και κατάταξη.

4.1.3 Υβριδικές Collaborative Filtering αρχιτεκτονικές

Εκτός από αυτές τις δύο μεθόδους Memory-Based και Model-Based μπορούν να υπάρξουν υβριδικές μέθοδοι που να ενσωματώνουν στοιχεία και από τις δύο μεθόδους για να αντιμετωπίσουν τους περιορισμούς που έχει κάθε μέθοδος ξεχωριστά και να βελτιώσουν την αποδοτικότητα των συστάσεων. Προβλήματα όπως το να υπάρχουν ελάχιστες αναφορές, το πως να χειριστούμε τους καινούριους χρήστες καθώς και η υποκειμενικότητα των ανθρώπινων ψηφοφοριών που έχουμε στα Memory-Based μοντέλα μπορούν να αντιμετωπιστούν με κόστος την αύξηση της πολυπλοκότητα των συστημάτων που δημιουργούμε.

Βιβλιογραφία:
[Recommender Systems based on Collaborative Filtering απο τον Zheng Wen]
[Recommender Systems: A GroupLens Perspective απο τους Joseph A. Konstan , John Riedl, AI Borchers και Jonathan L. Herlocker]

4.2 Παρουσίαση των Content Based Systems

Τα Content Based Recommendation systems είναι συστήματα που βασίζονται στην περιγραφή ενός αντικειμένου και στο προφίλ ενδιαφερόντων που έχει κάθε χρήστης. Το προφίλ του χρήστη περιγράφει τον τύπο των αντικειμένων που ενδιαφέρουν τον χρήστη. Έπειτα υπάρχει μια μέθοδος όπου συγκρίνει όλα τα αντικείμενα με το προφίλ του κάθε χρήστη και αποφασίζει πια τον ενδιαφέρουν για να του συσταθούν.

Αυτό που κάνουν τα Content Based Recommendation Systems είναι να αναλύουν τις περιγραφές των αντικειμένων που έχει δει ο χρήστης καθώς και αυτά που δεν έχει δει, με σκοπό να αναγνωρίζει από αυτά που δεν έχει δει ακόμη αυτά που τον ενδιαφέρουν.

Το πρώτο θέμα που θα μας απασχολήσει είναι ο τρόπος που αναπαριστάται η περιγραφή καθενός αντικειμένου εσωτερικά στο σύστημα μας. Αυτό που θέλουμε είναι να έχουμε τα δεδομένα μας σε μια δομημένη μορφή. Ο καλύτερος τρόπος είναι να τα αποθηκεύσουμε σε tables μιας βάσης δεδομένων. Οι στήλες θα περιγράφουν τις ιδιότητες του αντικειμένου. Κάθε αντικείμενο θα περιγράφεται από μια εγγραφή και θα ξεχωρίζονται αναμεταξύ τους από ένα μοναδικό ID.

Πολλές φορές δεν μας δίνονται οι περιγραφές των δεδομένων σε δομημένη μορφή. Μπορεί να είναι απλώς ένα κείμενο που περιγράφει ένα αντικείμενο. Τότε καλούμαστε εμείς να μετατρέψουμε το απλό κείμενο σε μια δομημένη αναπαράσταση. Πολλές λέξεις κυρίως αυτές που μας ενδιαφέρουν μπορούμε να τις αντιμετωπίζουμε ως ιδιότητες. Η ιδιότητα θα είναι boolean και θα υποδηλώνει αν υπάρχει μέσα στα κείμενο ή θα είναι κάποιος ακέραιος και θα υποδηλώνει το πόσες φορές συναντήσαμε την λέξη αυτή στο κείμενο.

Ένα επόμενο θέμα που προκύπτει στην διαδικασία αυτή είναι ότι λέξεις όπως “επεξεργάζομαι”, “επεξεργαστής”, “επεξεργασία” θα ξεχωρίσουν ως διαφορετικές έννοιες. Θα ήταν προτιμότερο να της ενοποιήσουμε σε έναν όρο. Αυτό επιτυγχάνεται με το να εργαστούμε και να θεωρήσουμε ως όρους που μας ενδιαφέρουν τις ρίζες των λέξεων. Έπειτα μπορούμε να κάνουμε χρήση μιας μεθόδου που μετρά την συχνότητα που συναντήθηκε ένας όρος αντιστρόφως ανάλογα με τον αριθμό των κειμένων. Όσο ποιο μεγάλος είναι ο αριθμός w(t,d) σημαίνει πως τόσο πιο συχνά συναντήθηκε ο αντίστοιχος όρος και τόσο πιο πολύ χαρακτηρίζει το αντικείμενο. Ο τύπος είναι

$$w(t,d) = \frac{tf_{t,d} \log\left(\frac{N}{df_t}\right)}{\sqrt{\sum_i (tf_{t_i,d})^2 \log\left(\frac{N}{df_{t_i}}\right)^2}}$$

όπου
d είναι ένα κείμενο
t είναι ένας όρος μέσα σε ένα κείμενο
N είναι το πλήθος των κειμένων
$tf_{t,d}$ είναι η συχνότητα που υπάρχει ο όρος t στο συγκεκριμένο κείμενο d
df_t είναι ο αριθμός των κειμένων στην συλλογή των N κειμένων που περιέχουν τον t όρο.

Αφού αναπαραστήσουμε την περιγραφή καθενός αντικειμένου, μας ενδιαφέρει να δημιουργήσουμε προφίλ που θα αντιπροσωπεύουν τα ενδιαφέροντα που μπορεί να έχει κάθε χρήστης. Τα προφίλ μπορούν να κατασκευάζονται κυρίως από δύο τύπους πληροφοριών. Τις προτιμήσεις των χρηστών και το ιστορικό των χρηστών.

Τα προφίλ που βασίζονται στις προτιμήσεις των χρηστών περιγράφουν τους τύπους των αντικειμένων που ενδιαφέρουν τον χρήστη. Με κάποιον τρόπο θα έχει τα n αντικείμενα που ενδιαφέρουν περισσότερο κάθε χρήστη και θα είναι σε θέση να προβλέψει κατά πόσο τον ενδιαφέρει το κάθε αντικείμενο που υπάρχει στο σύστημα.

Η επιλογή των n αντικειμένων που αρέσουν σε κάθε χρήστη μπορεί να γίνει ρητά και άμεσα με τον χρήστη να δηλώνει σε μία φόρμα που θα του δοθεί τα κουτάκια (checkbox) με τα θεματικά ενδιαφέροντα που τον αντιπροσωπεύουν ποιο πολύ. Μια άλλη αντίστοιχη περίπτωση είναι να υπάρχει μια φόρμα όπου ο χρήστης θα μπορεί με ελεύθερο κείμενο που θα γράφει να περιγράψει τα αντικείμενα που τον ενδιαφέρουν. Έπειτα οι λέξεις που έχουν καταχωρηθεί θα ταιριάξουν με τις περιγραφές αντικειμένων και θα επιλεγούν αυτά που ταιριάζουν περισσότερο.

Αυτή η μέθοδος έχει το μειονέκτημα ότι ζητείται από τον χρήστη να καταχωρίσει πληροφορίες στο σύστημα ρητά και αυτό μπορεί να τον κουράσει με αποτέλεσμα να μην θέλει να υποβληθεί σε αυτήν την διαδικασία. Αυτό βέβαια ξεπερνιέται με ποιο σύνθετες και προχωρημένες μεθόδους που θα εξετάσουμε σε επόμενα recommendation systems όπου θα πάρουμε τις πληροφορίες που χρειαζόμαστε με έναν άμεσο τρόπο παρακολουθώντας απλός την περιήγηση του χρήστη στους διάφορους ιστότοπους.

Ο άλλος τρόπος είναι να βασιζόμαστε στο ιστορικό του χρήστη και να καταγράφουμε όλες τις αλληλεπιδράσεις του χρήστη με το σύστημα όπως όλα τα αντικείμενα που επισκέφτηκε, αγόρασε και αναζήτησε. Υπάρχουν πολλές χρήσεις του ιστορικού των αλληλεπιδράσεων του

χρήστη. Το σύστημα μπορεί να έχει καταγεγραμμένα τα αντικείμενα που έχει δει και ενδιαφέρουν αυτόν τον καιρό τον χρήστη. Όσα αντικείμενα έχει ήδη επισκεφτεί ή έχει αγοράσει ο χρήστης θα φιλτραριστούν και δεν θα προταθούν για να συσταθούν αλλά μπορούν να είναι τα δεδομένα που θα εκπαιδεύσουν έναν αλγόριθμο μηχανικής μάθησης.

Το ιστορικό ενός χρήστη μπορεί να χρησιμοποιηθεί από ένα recommendation σύστημα βασισμένο σε συγκεκριμένους κανόνες (ruled-based) για να παράγει συστάσεις. Αυτό σημαίνει πως οι συστάσεις που θα γίνουν βασίζονται σε συγκεκριμένους κανόνες που εφαρμόζονται πάνω στα ιστορικά δεδομένα του χρήστη. Παράδειγμα κανόνα για να διευκρινίσουμε το τι εννοούμε είναι για κάθε βιβλίο ή ταινία που έχει δει ο χρήστης να συστήνουμε την συνέχεια της. Αν έχει δει την ταινία ο Άρχοντας των δακτυλιδιών το πρώτο μέρος είναι πολύ πιθανόν έπειτα να ενδιαφερθεί και για τον Άρχοντα των δακτυλιδιών το δεύτερο μέρος.

Αφού μαζέψουμε τα δεδομένα που κρίνουμε για κάθε χρήστη περνάμε στην επόμενη φάση όπου θα πρέπει να τα μοντελοποιήσουμε και να τα επεξεργαστούμε. Για κάθε χρήστη θα πρέπει να παράγουμε ένα μοντέλο των προτιμήσεων του, έπειτα να χρησιμοποιήσουμε αλγόριθμους κατηγοριοποίησης με σκοπό κάθε φορά που δίνεται ένα καινούριο αντικείμενο το σύστημα να είναι σε θέση να προβλέψει τον βαθμό που τον ενδιαφέρει. Υπάρχουν πολλοί αλγόριθμοι που μπορούν να ολοκληρώσουν το content based recommendation σύστημα μας. Ο κάθε ένας από τους αλγόριθμους αυτούς χρειάζεται μια εκτενής παρουσίαση για να κατανοηθεί και να υλοποιηθεί. Παρακάτω θα περιγράψουμε κάποιους από αυτούς δηλώντας απλός την βασική τους ιδέα,

4.2.1 Δέντρα Απόφασης (Decision Trees)

Τα δεδομένα που έχουμε μαζέψει μπορούμε με μια αναδρομική μέθοδο να τα διαμερίζουμε σε υπό - ομάδες δεδομένων έως ότου να φτάσουμε στο σημείο κάθε υπό - ομάδα να αντιστοιχεί σε μια κλάση. Με αυτόν τον τρόπο θα φτιαχτεί ένα δέντρο αποφάσεων. Η διαμέριση θα πραγματοποιηθεί με κριτήριο όπως το αν υπάρχει ή όχι μια συγκεκριμένη φράση ή λέξη. Αφού φτιαχτεί το δέντρο απόφασης από τα δεδομένα του προφίλ του χρήστη, μπορούμε για κάθε αντικείμενο να αποφανθούμε κατά πόσο αξίζει ή όχι να συσταθεί.

4.2.2 Μέθοδοι Κοντινότερου Γείτονα (Nearest Neighbor)

Με την χρήση του αλγόριθμου κοντινότερου γείτονα συγκρίνουμε κάθε αντικείμενο που

έχουμε στο σύστημα μας με τα αντικείμενα που υπάρχουν στο προφίλ του χρήστη. Τα αντικείμενα που είναι στο προφίλ του χρήστη ξέρουμε σε τι βαθμό τον ενδιαφέρουν. Από αυτή την σύγκριση βρίσκουμε τα k ποιο όμοια αντικείμενα με κάθε αντικείμενο του συστήματος. Τέλος αφού γνωρίζουμε το πόσο ενδιαφέρεται ο χρήστης για αυτά τα k αντικείμενα θα έχουμε μια ένδειξη για το πόσο ενδιαφέρεται για το κάθε άλλο αντικείμενο.

4.2.3 Ανατροφοδότηση Συνάφειας (Relavance Feedback)

Στην μέθοδο αυτή κάθε φορά προσφέρεται στο σύστημα μια λίστα με τα χαρακτηριστικά που ενδιαφέρουν τον χρήστη και δέχεται ως απάντηση τα αντικείμενα που ταιριάζουν ποιο καλά σε αυτά τα χαρακτηριστικά. Αυτό που γίνεται στην συνέχεια είναι να μπορεί ο χρήστης να ανατροφοδοτεί το σύστημα με το να δηλώνει κατά πόσο ήταν επιτυχημένη και τον ενδιέφερε κάθε σύσταση που πήρε. Αν τον ενδιέφεραν τα κριτήρια που έγινε η επιλογή ενισχύονται αν όχι τα κριτήρια πρέπει να αλλάξουν.

4.2.4 Άλλες Μέθοδοι

Υπάρχουν και άλλες μέθοδοι που ενσωματώνουν αλγόριθμους γραμμικής ταξινόμησης (Linear Classifiers), πιθανολογικές μέθοδοι (Probablistic methods), υβριδικές που συνδυάζονται με άλλες μεθόδους και πολλές άλλες. Το κοινό όλων αυτών είναι ότι προσπαθούν να βρουν από τις ιδιότητες που έχουν τα αντικείμενα, ποια είναι παρόμοια. Έπειτα γνωρίζοντας από το προφίλ του χρήστη πια αντικείμενα του άρεσαν να του συσταθούν παρόμοια αντικείμενα.

Βιβλιογραφία:
[Content-based Recommendation Systems από τους Michael J. Pazzani και Daniel Billsus]

4.3 Recommendation Systems βασισμένα σε Τεχνικές Εξόρυξης Δεδομένων προσαρμοσμένα σε Συνδέσμους

Θα εξετάσουμε ένα recommendation system που σχεδιάστηκε αρχικά με σκοπό να

εξειδικευθεί πάνω στις ανάγκες κάθε φοιτητή σε μια εκπαιδευτική πλατφόρμα στο διαδίκτυο. Εδώ θα χρησιμοποιηθούν τεχνικές εξόρυξης δεδομένων. Η γνώση που υπάρχει καταχωρήθηκε από την δράση και τα χαρακτηριστικά των χρηστών.

Σκοπός της εξόρυξης δεδομένων είναι να ανακαλύψει νέα, ενδιαφέροντα και χρήσιμη γνώση χρησιμοποιώντας ένα πλήθος από τεχνικές όπως κατηγοριοποίηση, ομαδοποίηση, κανόνες συσχετισμών και να ανακαλύψει πρότυπα για το από πια αντικείμενα ακολουθείτε κάθε αντικείμενο (sequential patern discovery). Θα χρησιμοποιηθεί κάθε είδους γνώση και πληροφορία γύρο από τους χρήστες με σκοπό να δημιουργηθούν μοντέλα χρηστών. Έπειτα θα χρησιμοποιηθούν αυτά τα μοντέλα για να εξατομικεύσουν κάθε σύσταση.

Μια από τις ποιο σημαντικές τεχνικές στην εξόρυξη δεδομένων που χρησιμοποιούμε είναι η ομαδοποίηση (clustering). Σκοπός μας στην ομαδοποίηση είναι να μαζέψουμε παρόμοια αντικείμενα και να τα ομαδοποιήσουμε σε κλάσεις. Κάθε αντικείμενο έχει μια εγγραφή διαστάσεων που το περιγράφει. Θα ομαδοποιήσουμε τις εγγραφές μαζί με βάση το πόσο κοντά είναι και κατά πόσο συνδέονται στον χώρο των διαστάσεων. Η βασική αρχή πάνω στην οποία στηρίζεται η ομαδοποίηση είναι να μεγιστοποιηθεί η ομοιότητα μεταξύ των αντικειμένων μιας ομάδας και η ομοιότητα μεταξύ των ομάδων αντικειμένων να είναι η ελάχιστη. Για αυτή την διαδικασία μπορούν να χρησιμοποιηθούν αλγόριθμοι βασισμένοι στις λειτουργίες και την ιεραρχία (hierarchican and function based algorithms). Ένας από τους ποιο κατάλληλους αλγορίθμους είναι ο k-means. Ο k-means αλγόριθμος βρίσκει το κέντρο ή την μέση τιμή κάθε ομάδας αντικειμένων μέσα στον n-διάστατο χώρο και προσπαθεί να επιλέξει ως αντικείμενα που ανήκουν στην ομάδα αυτή, τα αντικείμενα που έχουν την ελάχιστη απόσταση από το κέντρο.

Η ακολουθιακή μοντελοποίηση είναι μια τεχνική εξόρυξης δεδομένων εξίσου σημαντική. Σκοπός εδώ είναι να μελετήσουμε της δραστηριότητες και επιλογές των χρηστών και να αναγνωρίσουμε μοτίβα συμπεριφοράς μεταξύ χρηστών. Αν σε κάποιο χρήστη του αναλογεί ένα μοτίβο συμπεριφοράς είναι αναμενόμενο ποια θα είναι τα επόμενα αντικείμενα που θα τον ενδιαφέρουν. Την επιλογή αντικειμένων που θα κάνουμε θα την στηρίξουμε στο ότι και οι προηγούμενοι χρήστες που είχαν το ίδιο μοτίβο συμπεριφοράς επέλεξαν αυτά τα αντικείμενα.

Ας το ορίσουμε ποιο αυστηρά:

> Έστω ένα σύνολο από ακολουθίες αντικειμένων όπου κάθε ακολουθία αποτελείται από μια λίστα αντικειμένων. Και έστω ένα ελάχιστο σύνολο προτιμήσεων του χρήστη, που έχει ήδη επιλέξει. Σκοπός της ακολουθιακής μοντελοποίησης είναι να βρει όλες τις πιθανές ανακολουθίες αντικειμένων που μπορεί να επιλέξει στην συνέχεια ο χρήστης.

Για να βρεθεί η ακολουθία των αντικειμένων που είδε κάθε χρήστης συνήθως χρησιμοποιούνται log files στον server του συστήματος. Το πρόβλημα της εξόρυξης ακολουθιών

από την περιήγηση στο διαδίκτυο ενός χρήστη είναι η αναγνώριση των σχέσεων από ένα μεγάλο αριθμό ακολουθιών που έχουν κάνει οι χρήστες με την πάροδο του χρόνου. Κάθε ακολουθία χρηστών είναι ένα σύνολο από αντικείμενα που είδε ο χρήστης, διατεταγμένα χρονικά.

Η όλη διαδικασία για να παραχθούν συστάσεις που είναι βασισμένες στις τεχνικές εξόρυξης βασίζονται σε τρία στάδια. Την προετοιμασία των δεδομένων, την ανακάλυψη σχέσεων μεταξύ των δεδομένων και τέλος με βάση αυτές τις σχέσεις να γίνουν συστάσεις. Τα πρώτα δύο στάδια μπορούν να γίνουν off-line αλλά το στάδιο που θα παραχθούν οι συστάσεις θα γίνει on-line.

Η προετοιμασία των δεδομένων συντελείται με το να δεχθεί τα log files και τα προφίλ των χρηστών από τον server και να τα μετατρέψει σε κατάλληλης μορφής δεδομένα τα οποία έπειτα θα μπορούμε να τα επεξεργαστούμε. Συνήθως χρησιμοποιείται μια βάση δεδομένων.

Η ανακάλυψη σχέσεων μεταξύ των δεδομένων γίνεται με κάποια από τις τεχνικές εξόρυξης δεδομένων που έχουμε αναφέρει: την ομαδοποίηση, την εύρεση επαναλαμβανόμενων μοτίβων από ακολουθίες αντικειμένων και τους κανόνες συσχετισμού.

Η παραγωγή των συστάσεων γίνεται βασισμένη στις σχέσεις μεταξύ των δεδομένων που βρήκαμε στο προηγούμενο στάδιο. Όταν ένας χρήστης θα έχει επισκεφτεί μια σειρά αντικειμένων που συσχετίζονται κάπως, επόμενο είναι να συσταθούν και τα υπόλοιπα αντικείμενα που συσχετίζονται μαζί τους και δεν τα έχει επισκεφτεί ακόμη ο χρήστης.

Το σύστημα συστάσεων μπορεί να διαφοροποιηθεί στην αρχιτεκτονική του με βάση τον τρόπο που θα γίνεται η εξόρυξη των δεδομένων. Η μία προσέγγιση είναι να χρησιμοποιεί τις πληροφορίες των χρηστών του που είναι αποθηκευμένες στα log files. Στη συνέχεια να εφαρμόσει έναν αλγόριθμο εξόρυξης δεδομένων πάνω στις συνεδρίες που είχαν οι χρήστες και να ανακαλύψει το ποιο συχνό μοτίβο περιήγησης πάνω στα αντικείμενα που έχει το σύστημα. Έπειτα με βάση το μοτίβο που έχουν κάποιοι χρήστες μπορούν να γίνουν προβλέψεις για τις συστάσεις που θα ενδιαφέρουν κάθε χρήστη. Σε αυτή την απλή περίπτωση που χρησιμοποιείται μόνο ένας αλγόριθμος εξόρυξης δεδομένων συνήθως επιλέγεται ο ακολουθιακός αλγόριθμος. Ένα πρόβλημα αυτής της μεθόδου είναι ότι οι καινούριοι χρήστες λαμβάνουν συστάσεις που βασίζονται μόνο στην παρούσα περιήγηση που κάνουν.

Η ποιο προχωρημένη αρχιτεκτονική συστημάτων συστάσεων χρησιμοποιεί επιπλέον πληροφορίες γύρω από τους χρήστες όπως τα προφίλ τους. Επίσης είναι σε θέση να χρησιμοποιηθούν πολλοί αλγόριθμοι εξορύξεως δεδομένων όπως της ομαδοποίησης και των επακόλουθων μοτίβων. Σε μια τέτοια περίπτωση πρώτα θα σχηματιστούν οι ομάδες των χρηστών με βάση την κοινή τους συμπεριφορά μέσα στο σύστημα. Έπειτα θα βρεθούν τα ακολουθιακά μοτίβα κάθε μιας κλάσης.

Από ότι καταλαβαίνουμε η ποιο σημαντική διαδικασία της μεθόδου είναι το πώς θα γίνει η εξόρυξη των δεδομένων και θα παραχθούν οι νόμοι με τους οποίους θα γίνουν οι συστάσεις σε κάθε ομάδα. Αρχικά από τα log files για όλες τις συνεδρίες των χρηστών δημιουργούμε ένα log file για κάθε χρήστη. Έπειτα κάνουμε την ομαδοποίηση των χρηστών με παραμέτρους τις σελίδες που επισκέφτηκαν

Για την ομαδοποίηση μπορούμε να χρησιμοποιήσουμε τον k-means αλγόριθμο. Ο k-means αλγόριθμος διαμερίζει n χρήστες σε k ομάδες όπου ο κάθε χρήστης ανήκει στην ομάδα που έχει ποιο κοντά τον μέσο όρο της σε αυτόν.

Στην συνέχεια θα χρησιμοποιηθεί κάποιος αλγόριθμος για να βρεθεί ένα μοτίβο συμπεριφοράς για κάθε ομάδα. Αλγόριθμοι αυτής της κατηγορίας είναι οι AprioriAll, GSP και PrefixSpan. Από αυτό το μοτίβο συμπεριφοράς μπορούν να εξαχθούν οι κανόνες με βάση τους οποίους θα γίνονται οι συστάσεις σε κάθε ομάδα.

Στην διαδικασία όπου γίνονται οι συστάσεις υπολογίζονται οι ενεργοί χρήστες σε συνδυασμό με τα δεδομένα που εξορύχθηκαν για να παρέχουν εξατομικευμένες συστάσεις. Ποιο συγκεκριμένα η διαδικασία είναι η εξής. Ανατρέχουμε στο προφίλ του χρήστη για να δούμε ποιες σελίδες επισκέφτηκε. Υπολογίζουμε με μια μέθοδο ελάχιστης απόστασης την σχέση των σελίδων που επισκέφτηκε με το κέντρο κάθε μιας ομάδας σε έναν n-διάστατο χώρο και τον κατατάσσουμε στην ομάδα που του ταιριάζει περισσότερο. Έπειτα του γίνονται οι συστάσεις με βάση τους νόμους που υπάρχουν σε κάθε ομάδα.

Βιβλιογραφία:
[Personalized Links Recommendation Based on Data Mining in Adaptive Educational Hypermedia Systems από τους Cristóbal Romero , Sebastián Ventura , Jose Antonio Delgado και Paul De Bra]

4.4 Recommendation System που χρησιμοποιεί τον k-Nearest Neighbor Αλγόριθμο.

Ένας πολύ χρήσιμος αλγόριθμος που μπορεί να είναι η καρδιά ενός recommendation συστήματος είναι ο k-nearest neighbor. Πολλά recommendation συστήματα που κυρίως βασίζονται σε collaborative filtering προσεγγίσεις των χρησιμοποιούν. Αρχικά θα περιγράψουμε την δομή του αλγορίθμου και έπειτα θα εξετάσουμε ένα recommendation system που τον χρησιμοποιεί.

4.4.1 Περιγραφή του k-Nearest Neighbor Αλγορίθμου

Στην γενική μορφή του αλγορίθμου μπορούμε να ταξινομήσουμε και να ομαδοποιήσουμε αντικείμενα ή χρήστες εξίσου. Εμείς για την περαιτέρω περιγραφή του αλγορίθμου θα χρησιμοποιήσουμε τον όρο αντικείμενα αλλά δηλώνουμε ότι είτε αναφέρουμε αντικείμενα είτε

χρήστες η διαδικασία είναι η ίδια.

Ο k-nearest neighbor αλγόριθμος χρησιμοποιεί μια από τις πιο βασικές και απλές μεθόδους ταξινόμησης - κατάτμησης ακόμη και αν η γνώση που υπάρχει από πριν για τα δεδομένα είναι ελάχιστη ή ακόμη και καθόλου. Ανήκει στην κατηγορία των instance-based αλγόριθμων μάθησης. Αυτό σημαίνει ότι η διαδικασία μάθησης αφορά απλώς την σωστή αποθήκευση των δεδομένων εκπαίδευσης. Τα δεδομένα εκπαίδευσης είναι τα δεδομένα τα οποία θα λειτουργήσουν ως σημείο αναφοράς και σύμφωνα με αυτά θα επεξεργαστούμε και θα διαχειριστούμε τα περαιτέρω δεδομένα που θα εκλάβουμε (νέο instance).

Κάθε φορά που έρχεται ένα νέο instance ξεκινά μια διαδικασία επεξεργασίας αυτού μαζί με τα δεδομένα εκπαίδευσης. Αυτός είναι ο λόγος που ο αλγόριθμος ανήκει στην κατηγορία των Lazy Learning. Κάθε φορά που ένα νέο instance πρόκειται να ταξινομηθεί, υπολογίζεται η ομοιότητα του με κάθε ένα από τα αποθηκευμένα δεδομένα εκπαίδευσης.

Ο k-nearest neighbor αλγόριθμος βασίζεται σε μια συνάρτηση απόστασης. Συνήθως χρησιμοποιούμε την Ευκλείδεια απόσταση ή την απόσταση συνημίτονου, μεταξύ κάθε αντικειμένου εκπαίδευσης και του αντικειμένου που πρόκειται να ταξινομηθεί.

Ο αλγόριθμος k-nearest neighbor τοποθετεί το καινούριο αντικείμενο κοντά σε k αντικείμενα που ταιριάζει περισσότερο, όπου το k είναι ένας μικρός θετικός ακέραιος τον οποίο θα τον έχουμε ως παράμετρο. Με αυτόν τον τρόπο έχουμε τοποθετήσει ένα αντικείμενο όπου έχει γύρω του k γείτονες. Το ποιοι θα είναι αυτοί οι γείτονες θα το βρούμε υπολογίζοντας το με κάποια συνάρτηση απόστασης. Τα αντικείμενα τα αναπαριστούμε με διανύσματα όπου κάθε συνιστώσα τους είναι μια παράμετρος του αντικειμένου Σε κάθε παράμετρο θα χρησιμοποιήσουμε κάποιον συντελεστή βαρύτητας. Ο συντελεστής βαρύτητας μπορεί να είναι και μηδενικός αν μια παράμετρος δεν μας ενδιαφέρει να την συνυπολογίσουμε

Στη συνέχεια υπολογίζεται η ομοιότητα (similarity) κάθε αντικειμένου με το αντικείμενο το οποίο πρόκειται να ταξινομηθεί. Αυτό είναι δυνατό με την χρήση κάποιας συνάρτησης που υπολογίζει την απόσταση μεταξύ δυο διανυσμάτων. Μερικά παραδείγματα είναι: cosine similarity(ομοιότητα συνημίτονου), extended cosine similarity(Jaccard coefficient), Euclidean distance(ευκλείδεια απόσταση) και Manhattan distance.

Από τους προηγούμενους υπολογισμούς θα εξαχθούν τα k αντικείμενα τα οποία έχουν την μεγαλύτερη ομοιότητα με το αντικείμενο που πρόκειται να ταξινομηθεί. Όσο πιο όμοια είναι δύο αντικείμενα τόσο πιο μικρό θα είναι το αποτέλεσμα της συνάρτησης καθώς και η απόσταση τους. Αυτά τα k αντικείμενα είναι οι k κοντινότεροι γείτονες. Η κλάση στην οποία θα τοποθετηθεί το αντικείμενο θα είναι η κλάση όπου βρίσκονται οι περισσότεροι κοντινοί γείτονες.

Οι Cover και Hart έδειξαν ότι κάτω από συγκεκριμένες υποθέσεις το λάθος του k-nearest neighbor αλγόριθμου περιορίζεται στο μισό του λάθους του Bayes. Τα θετικά του αλγορίθμου αυτού είναι ότι είναι εύκολος στο να τον καταλάβει κάποιος και να τον υλοποιήσει. Έπειτα δουλεύει καλά σε περιπτώσεις multi-model κλάσεων και σε εφαρμογές όπου κάποιο αντικείμενο μπορεί να ανήκει σε περισσότερο από μια κλάσεις. Επίσης, έχει υψηλή αποδοτικότητα ακόμη και σε περιπτώσεις όπου τα δεδομένα εκπαίδευσης (training data) περιέχουν θόρυβο (noisy) και σε

περιπτώσεις όπου τα δεδομένα εκπαίδευσης είναι πολλά.

Πρέπει να δοθεί αρκετή σημασία στο k του αλγορίθμου γιατί επηρεάζει την αποδοτικότητα του ταξινομητή και τις περισσότερες φορές είναι δύσκολο να προσδιοριστεί από πριν ποια τιμή του k δίνει τα βέλτιστα αποτελέσματα. Μικρό k δίνει αποτελέσματα που είναι ευαίσθητα σε θορυβώδη δεδομένα. Μεγάλο k μπορεί να έχει ως αποτέλεσμα πολλά από τα γειτονικά αντικείμενα να μην συσχετίζονται πραγματικά. Οι Robert M. Bell και Yehuda Koren θεωρούν πως τυπικές τιμές του k που μπορούν να χρησιμοποιηθούν είναι μεταξύ του 20 έως το 50.

Μέλημα μας είναι επίσης ποια συνάρτηση απόστασης να χρησιμοποιήσουμε. Διαφορετικές συναρτήσεις απόστασης μπορεί να δίνουν αρκετά διαφορετικά αποτελέσματα

Δύο μειονεκτήματα που είναι δύσκολο να τα αντιμετωπίσουμε είναι πρώτον ότι ο k-nearest neighbor αλγόριθμος είναι instance-based αλγόριθμος μάθησης, και αυτό σημαίνει ότι δεν μπορεί να γίνει καμία εκπαίδευση, μέχρι να φτάσει κάποιο αντικείμενο για να ταξινομηθεί. Και δεύτερον το ότι έχει να υπολογίσει την απόσταση κάθε όρου του κειμένου που θα ταξινομηθεί με όλα τα αντικείμενα εκπαίδευσης έχει σαν αποτέλεσμα ένα μεγάλο κόστος υπολογισμού.

Όπως αναφέραμε στην αρχή στον k-nearest neighbor αλγόριθμο μπορούν να βασιστούν και να υλοποιηθούν πολλά recommendation συστήματα. Ο λόγος είναι ότι μέσω του k-nn μπορούν να ομαδοποιηθούν τα αντικείμενα ή οι χρήστες που είναι παρόμοιοι. Με αυτόν τον τρόπο από εκεί που είχαμε μια σειρά από αντικείμενα έχουμε τώρα έναν αριθμό συνόλων που όλα τα αντικείμενα μέσα σε κάθε σύνολο έχουν μια ομοιότητα αναμεταξύ τους.

Μια γενική ιδέα για το πως θα μπορούσαμε να χρησιμοποιήσουμε αυτά τα σύνολα είναι να βασιστούμε στην ιδέα ότι αν ένας χρήστης ενδιαφέρθηκε για κάποια αντικείμενα που ανήκουν σε ένα σύνολο επόμενο είναι να ενδιαφέρεται και για τα υπόλοιπα αντικείμενα του συνόλου αυτού. Αντίστοιχα αν έχουμε κατατάξει στα σύνολα παρόμοιους χρήστες και ένας χρήστης ανήκει σε ένα σύνολο χρηστών, τα αντικείμενα που θα είναι πιο πιθανόν να τον ενδιαφέρουν θα είναι αυτά που επισκέφτηκαν οι περισσότεροι από τους άλλους χρήστες του συνόλου αυτού. Βέβαια ο k-nn αλγόριθμος θα μπορούσε να εφαρμοστεί και με άλλους τρόπους για την επιλογή συστάσεων.

4.4.2 Ένα Recommendation System που χρησιμοποιεί τον k-Nearest Neighbor αλγόριθμο με έναν βελτιωμένο τρόπο.

Μια ενδιαφέρουσα χρήση του k-nn αλγορίθμου είναι αυτή που προτάθηκε από τους Robert M. Bell και Yehuda Koren στα ερευνητικά εργαστήρια της AT&T. Αυτή η μέθοδος έχει τα εξής στάδια. Αρχικά κάνουμε μια κανονικοποίηση στα δεδομένα έπειτα εφαρμόζουμε τον k-nn

αλγόριθμο και τέλος κάνουμε έναν υπολογισμό μιας τιμής βάρους για κάθε αντικείμενο που υποδηλώνει το κατά πόσο ταιριάζει για να συσταθεί.

Υπάρχουν κάποιοι παράγοντες που μπορούν συστηματικά να επηρεάζουν τις βαθμολογίες που δίνονται για τα αντικείμενα και θα πρέπει να τους λάβουμε υπόψιν μας και να τους εξαλείψουμε πριν ξεκινήσουμε να εφαρμόζουμε τον k-nn αλγόριθμο. Αυτοί είναι ότι κάποιοι χρήστες μπορεί να δίνουν συνεχώς υψηλότερες βαθμολογίες από ότι οι υπόλοιποι ή κάποια αντικείμενα λόγο της φύσης τους να αποδέχονται υψηλότερες βαθμολογίες από τα υπόλοιπα χωρίς αυτό να συνεπάγεται ότι έχουν και την αντίστοιχη προτίμηση.

Από τις τιμές όπως το πλίθος και ο μέσος όρος των βαθμολογιών που έχει ένα αντικείμενο ή που έχει καταχωρήσει ένας χρήστης, μπορούμε να ξεχωρίσουμε τους χρήστες που τους αρέσουν τα ποιο δημοφιλή αντικείμενα από τα πιο εξεζητημένα για αυτό και οι βαθμολογίες τους είναι αντίστοιχες.

Συστηματικές αποκλίσεις στης βαθμολογίες των αντικειμένων έχουμε και από κάποια χαρακτηριστικά των βαθμολογιών όπως το πότε δόθηκαν αυτές οι βαθμολογίες από τους χρήστες. Παραδείγματος χάριν είναι φυσικό οι βαθμολογίες που παίρνει μια ταινία μόλις κυκλοφορήσει να είναι ψηλότερες από ότι έπειτα από μερικά χρόνια.

Αφού αφαιρέσουμε όλες τις επιρροές που αλλοιώνουν την αντικειμενική βαθμολογία των αντικειμένων μπορούμε να ξεκινήσουμε να εφαρμόζουμε τον k-nn αλγόριθμο στα δεδομένα. Για κάθε αντικείμενο i που υπάρχει και για κάθε χρήστη u μας ενδιαφέρει να υπολογίσουμε τον βαθμό ενδιαφέροντος που θα έχει αυτός ο χρήστης για το αντικείμενο αυτό. Αυτή την τιμή την συμβολίζουμε με r_{ui}.

Για κάθε αντικείμενο i που υπάρχει θα βρούμε με τον αλγόριθμο k-nn τα k πιο παρόμοια αντικείμενα που υπάρχουν και από αυτά τα k αντικείμενα θα υπολογίσουμε το ενδιαφέρον που έχει ο χρήστης για αυτό το i αντικείμενο. Τα αντικείμενα με το πιο ψηλός r_{ui} είναι αυτά που αξίζει περισσότερο να συσταθούν.

Το r_{ui} το πόσο ενδιαφέρεται ο χρήστης u για ένα αντικείμενο i που δεν το έχει επισκεφτεί υπολογίζεται από τον τύπο

$$r_{uj} \leftarrow \sum_{j \subset N(i;u)} w_{ij} \cdot r_{uj}$$

Όπου

N(i;u) είναι το σύνολο των k ποιο παρόμοιων αντικειμένων στο i αντικείμενο από αυτά που έχει επισκεφτεί ο χρήστης.

r_{uj} θα είναι τα αντικείμενα που έχει επισκεφτεί ο χρήστης και ξέρουμε τον βαθμό του πόσο ενδιαφέρεται για αυτά.

w_{ij} Είναι μια τιμή που υποδηλώνει το κατά πόσο συσχετίζεται το i αντικείμενο με κάθε ένα από τα N(i;u). Το w_{ij} υπολογίζεται από τους τύπους.

$$\widehat{A}w = \widehat{b}$$

$$\widehat{A}_{jk} = \frac{|U(j,k)| \cdot Ajk + \beta \cdot avg}{|U(j,k)| + \beta}$$

$$\widehat{b}_{j} = \frac{|U(i,j)| \cdot \overline{b}_{j} + \beta \cdot avg}{|U(i,j)| + \beta}$$

Όπου
U(j,k) είναι το σύνολο των χρηστών που βαθμολόγησαν τα αντικείμενα j και k.
β είναι μια σταθερά συρρίκνωσης που την παίρνουμε ίση με 500
avg είναι ο μέσος όρος όλων των δυνατών A_{gk}

$$\overline{A}_{jk} = \frac{\sum_{u \in U(j,k)} r_{uj} \cdot r_{uk}}{|U(j,k)|}$$

$$\overline{b}_{j} = \frac{\sum_{u \in U(i,j)} r_{uj} \cdot r_{ui}}{|U(i,j)|}$$

Βιβλιογραφία:
[Improved Neighborhood-based Collaborative Filtering απο τους Robert M. Bell και Yehuda Koren]
[A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features απο τους Scott Cost και Steven Salzberg]
[A Fuzzy K-Nearest Neighbor Algorithm απο τους James M. Keller, Michael R. Gray και James A. Giens]

4.5 Recommendation System βασισμένο στο View Time

Μια πολύ ενδιαφέρουσα μέθοδος για να υλοποιήσουμε ένα recommendation system είναι να βασιστούμε στο viewing time. Δηλαδή στον χρόνο που διαθέτει ένας χρήστης στο να δει μια ιστοσελίδα. Αυτή η μέθοδος έχει πολλά θετικά.

- Έχει διαφάνεια (transparency) δηλαδή η σύσταση μπορεί να γίνει χωρίς να ασχοληθεί και να ενημερωθεί ο χρήστης για την όλη διαδικασία των ενεργειών που έγιναν από το recommendation system.

- Είναι ανεξάρτητη από χρήστες (user independent).

- Οι συστάσεις που γίνονται σε έναν χρήστη δεν εξαρτώνται από την συμπεριφορά που είχαν στο σύστημα άλλοι χρήστες.

- Δεν χρειάζονται πολλά δεδομένα πλοήγησης ή δεδομένα που έχουν αποθηκευθεί στο ιστορικό του χρήστη.

- Συνδυάζουμε μόνο τα τωρινά δεδομένα πλοήγησης με τα αντικείμενα που έχουμε ως υποψήφια για να κάνουμε την σωστή σύσταση.

Ας παρουσιάσουμε την μέθοδο που βασίζεται στον χρόνο που είδε ένας χρήστης μια ιστοσελίδα αναλυτικά. Αρχικά πιστεύομε ότι οι άνθρωποι βλέπουν περισσότερη ώρα αντικείμενα που τους ενδιαφέρουν από ότι αντικείμενα που δεν τους ενδιαφέρουν. Ένα αντικείμενο που ενδιαφέρει τον χρήστη θα το κοιτάξει περισσότερη ώρα για να σκεφτεί τις λεπτομέρειες του, να σκεφτεί μια πιθανή του χρήση καθώς και ο τρόπος με τον οποίο θα προβεί στην αγορά του. Αντίθετα ένα αντικείμενο που δεν ενδιαφέρει έναν χρήστη δεν υπάρχουν πολλοί λόγοι για να το μελετήσει και να το κοιτάξει.

Ακόμη και να ξοδέψει κάποιον χρόνο για να δει ένα αντικείμενο που δεν τον ενδιαφέρει αυτός μπορεί να είναι για να δει τι προϊόντα υπάρχουν στην αγορά ή μέχρι να καταλάβει τι ακριβώς είναι αυτό το αντικείμενο. Όπως και να έχει αναμένουμε ο χρόνος που ένας χρήστης θα δει ένα αντικείμενο που τον ενδιαφέρει θα είναι περισσότερος από τον χρόνο που θα δει ένα αντικείμενο που δεν τον ενδιαφέρει.

Σε ένα επόμενο στάδιο αυτής της μεθόδου θα μπορούσαμε να πούμε ότι ακόμη και αν δύο ή περισσότερα αντικείμενα πραγματικά ενδιαφέρουν έναν χρήστη το κριτήριο με το οποίο θα μπορέσουμε να ταξινομήσουμε και να ξεχωρίσουμε πόσο πολύ ενδιαφέρει ένα αντικείμενο σε σχέση με ένα άλλο είναι πάλι ο χρόνος που θα έχει δαπανήσει ο χρήστης στο κάθε ένα.

Αυτή η μέθοδος βασίζεται στην επιστήμη της ψυχολογίας. Συγκεκριμένα ψυχολογικές μελέτες έχουν αναφέρει τον συσχετισμό μεταξύ του χρόνου που βλέπει κάποιος ένα αντικείμενο με την προτίμηση του για αυτό. Ο Berlyne και ο Lawrence το 1964 βρήκαν έπειτα από πειράματα ότι το πόσο αρέσει ένα σχήμα σε έναν άνθρωπο που το βλέπει συσχετίζεται από το πόση ώρα το κοιτάζει. Οι Faw και Nunnaly το 1967 κάναν το εξής πείραμα. Ζήτησαν από 30 ανθρώπους να δουν μια συλλογή από εικόνες και να ψηφίσουν αυτή που τους αρέσει περισσότερο. 27 από τους 30 επέλεξαν να ψηφίσουν την εικόνα που κοίταζαν πιο πολύ ώρα. Σε πιο πρόσφατα πειράματα Ο Konstan το 1997 βρήκε μια θετική συσχέτιση μεταξύ του χρόνου που διαβάζει ένας χρήστης ειδήσεις από ένα διαδικτυακό τόπο σε σχέση με το βαθμό που ψήφισε για το πόσο ενδιαφέρον βρήκε τα άρθρα που διάβασε.

Οι Jeffrey Pearsons, Paul Ralph και η Katherine Gallagher κάναν ένα πείραμα όπου είχαν ένα online shop για να δουν την συμπεριφορά των εθελοντών καταναλωτών όσον αφορά το time view και τις προτιμήσεις τους. Τα αποτελέσματα που είχαν ήταν ότι ο μέσος χρόνος που οι χρήστες έβλεπαν αντικείμενα που δεν τα έβαλαν στο εικονικό καλάθι αγορών τους ήταν περίπου 7 δευτερόλεπτα. Ενώ ο μέσος χρόνος που έβλεπαν αντικείμενα που τα έβαζαν στο καλάθι αγορών ήταν περίπου 14 δευτερόλεπτα. Αυτό το πείραμα επιβεβαιώνει την ορθότητα της μεθόδου αυτής.

Εννοείται πως υπάρχουν και άλλοι παράγοντες εκτός από την προτίμηση που έχει ένας χρήστης για ένα προϊόν που μπορεί να επηρεάσουν το time view. Κάποιοι από αυτούς τους παράγοντες είναι το πόσες εικονικές λεπτομέρειες υπάρχουν στο online shop, το μέγεθος του κειμένου που περιγράφει το προϊόν, οι εικόνες που μπορεί να υπάρχουν και να αποσπούν την προσοχή του χρήστη. Ακόμη και ο περιηγητής ή η διαδικασία της παραγγελίας μπορεί να καταναλώσουν χρόνο ο οποίος μπορεί να προσμετρηθεί στον time view χρόνο.

Θα περιγράψουμε πως λειτουργεί ένα recommender system που βασίζεται στο time view των χρηστών. Οι Jeffrey Pearsons, Paul Ralph και η Katherine Gallagher ονόμασαν τον αλγόριθμο που υλοποιεί το σύστημα αυτό DESIRE από τα αρχικά Desirability Estimator and Structured Information Recommendation Engine. Αυτό που θέλουμε να κάνει το σύστημα μας είναι όταν ένας χρήστης βλέπει ένα online κατάλογο σε κάθε σελίδα να πραγματοποιείτε με διαφάνεια μια βαθμολογία του πόσο ενδιέφερε το προϊόν αυτής της σελίδας σε σχέση με τα προϊόντα των άλλων σελίδων. Το δεδομένο που θα προκαλέσει αυτή την βαθμολογία είναι ο χρόνος που είδε την κάθε σελίδα. Αυτές τις βαθμολογίες των αντικειμένων μπορούμε να τις κρατήσουμε σε έναν πίνακα

Κάθε αντικείμενο που έχουμε το αναλύουμε σε κάποιες συγκεκριμένες ιδιότητες και βαθμονομούμε κάθε μια από αυτές τις ιδιότητες με μια τιμή. Από τον πίνακα που έχουμε τις βαθμολογίες των αντικειμένων που έχει δει ο χρήστης κατασκευάζουμε έναν ιδανικό συνδυασμό τιμών για κάθε ιδιότητα. Το επόμενο στάδιο είναι να συγκρίνουμε τον ιδανικό συνδυασμό τιμών των ιδιοτήτων με τις τιμές που έχουν οι ιδιότητες για κάθε αντικείμενο. Με αυτόν τον τρόπο μπορούμε να κατατάξουμε τα αντικείμενα με κριτήριο του ποια είναι πιο κοντά στην ιδανική προτίμηση του χρήστη. Ως τελικό αποτέλεσμα μπορούμε να παρουσιάσουμε ή ένα top-n αντικειμένων ή όλα τα αντικείμενα που ξεπερνάνε έναν βαθμό ομοιότητας και πάνω.

Οι ιδιότητες ενός αντικειμένου που βαθμονομούνται χωρίζονται σε δύο κατηγορίες. Ιδιότητες κειμένου και αριθμητικές ιδιότητες. Ιδιότητες κειμένου είναι όλες οι ονομαστικές και κατηγοριοποίησης, ιδιότητες όπως το όνομα ενός προϊόντος. Αριθμητικές ιδιότητες είναι όλες οι ιδιότητες που έχουν να κάνουν με αριθμητικά διαστήματα ή αναλογίες όπως είναι η τιμή ενός προϊόντος. Όλες οι ιδιότητες δεν έχουν την ίδια σημασία για αυτό και δεν τις συνυπολογίζουμε στον ίδιο βαθμό. Το πόσο επιθυμητό είναι ένα αντικείμενο υπολογίζεται παίρνοντας υπόψιν την ομοιότητα κάθε ιδιότητας σε σχέση με το ιδανικό αντικείμενο. Αυτό το πολλαπλασιάζουμε με έναν συντελεστή βαρύτητας ανάλογο με το πόσο σημαντική είναι η εκάστοτε ιδιότητα.

Παρακάτω παραθέτουμε αναλυτικά τον αλγόριθμο έτσι όπως τον σχεδίασαν οι Οι Jeffrey Pearsons, Paul Ralph και η Katherine Gallagher στο Using Viewing Time to Infer User Preference in Recommender Systems

Stage 1 Input ● Item-Attribute Matrix, 1A (N items by M atributes) ● Positive Example threshold Output ● None Steps: 1. For each numeric attribute in 1A, calculate its z-score assuming a normal distribution Stage 2 Input ● Item-Preferance Matrix (T items by 2 attributes, the item ID and the preference) Output ● Top-n recommendations set or Better-than-n recommendation set for customer c Ωc ● Desirability of each recommendation in Ωc Steps: 1. Describe the customer's preferences in terms	2. Find a representation of the similarity between the target and p in terms of textual attributes. This representation will be a set of individual similarit indices. 2.1. Recall that the customer object contains a weighting from (-1 to 1 inclusive) for each textual value. 2.2. For each value in p: 2.2.1. Calculate the similarity between the customer's target item and p, in terms of the current element, as the weighting given to the current value in the Targer (in T_V). If the value is not in T_V leave it out. 2.2.2. Normalize this weighting by adding 1 and dividing by 2 2.3. Construct the set of all these individual similarity indices by S_V, (fo Similarity of Values) 3. Find a representation of the similarity between the target and p in terms of numeric attributes. This representation will be a set of individua similarity indices. 3.1. For each numeric element: 3.1.1. Find z-scores for "ideal

of preferences for item characteristics.

1.1. Recall: V is the set of all the values that the textual attributes can hold. Construct the customerobject, containing a set, T_V (for Target Values), as follows:

1.1.1. For every element in V, construct a corresponding element in T_V. This element is the average weight for all the products in the training set that contained the value in question, or 0 if none of the training set items contained it.

1.2. Recall, β is the set of all numeric attributes. The customer object contains an ideal quantity for each numeric attribute.

1.2.1. The ideal quantity is a weighted mean of the attribute's value in all of the positive examples. We define a positive example as one whose weight is greater than some threshold between -1 and 1.

1.2.2. Construct the set of the current customer's ideal quantities T_m (for Target Means).

1.2.3. Each element in β has a corresponding element in T_V defined as: $\frac{\sum wq}{\sum w}$, for each positive example, In English, the element in T_m corresponding to an attribute, a, is given by: the sum of the weight (w) of the each positive example times the quantity (q) assigned to a in that example, divided by the sum of the weights of all the positive examples.

Repeat Steps 2,3 and 4 for each quantity" in the target.

3.1.2. The absolute value of the difference between the z-score of the ideal quantity and the z-score of the actual quantity in a item is the dissimilarity, which has a maximum value of6. To get the similarity between the item and thetarget divide the dissimilarity by six and subtract the answer from 1.

3.2. Construct the set of all these individual similarity indices S_m (for Similarity Means)

4. Combine S_V and S_m creating a single index tha represents the similarity between the item and the target.

4.1. Since some attributes are more important to the buying decision than others, the similarity is a weighted mean. R denotes the set of Relative attribute weightings.

4.1.1. Divide R into two subsets, R_V and R_m , whicj correspond to S_V and S_m respectively.

4.2. Calculate the cumulative similarity index, between p and the target as:

$$\frac{\sum S_V R_V + \sum S_m R_m}{\sum R}$$

That is, multiply each individua similarity index by its relative weighting, and then divide the sum of these products by the sum of the relative weightings. This gives a cumulative similarity index

5. We now have a cumulative similarity index fo each item. Let Ω denote the top-n recommendations, or all of the recommendations with desirability greater that some threshold. Return Ω.

item, p ε I	

Πίνακας: 2Αλγόριθμος του Recommendation Systemπου είναι βασισμένο στο View Time

Βιβλιογραφία:
[Jeffrey Pearsons, Paul Ralph και ηKatherine Gallagher: Using Viewing Time to Infer User Preference in Recommender Systems]

4.6 Recommendation System βασισμένο στο Referral Web

Το Recommendation systems που βασίζεται στο Referall web στηρίζεται στο ότι η διάδοση των πληροφοριών και της εξειδικευμένης γνώσης πάνω σ' ένα θέμα μπορεί να αναπαρασταθεί με ένα δίκτυο από συνεργάτες, συμφοιτητές και φίλους. Ένα άτυπο κοινωνικό δίκτυο από ανθρώπους που επικοινωνούν, συνεργάζονται και ασχολούνται με παρόμοια θέματα παρουσιάζει ομοιότητες με την δομή ενός επίσημου οργανισμού που αναθέτει και επιμερίζει εργασίες στους εργαζόμενους του.

Από ένα δίκτυο με ανθρώπους που επικοινωνούν, συνεργάζονται και ασχολούνται με παρόμοια θέματα μπορούμε να δημιουργήσουμε ένα recommendation system το οποίο θα προτείνει απαντήσεις σε ερωτήματα που μόνο ένας ιδικός του θέματος θα ήταν σε θέση να απαντήσει. Εδώ έχουμε δύο πλεονεκτήματα. Οι συστάσεις που γίνονται είναι σε ένα εξειδικευμένο θέμα που δεν θα μπορούσε οποιοσδήποτε να απαντήσει. Οι απαντήσεις και τα αποτελέσματα που έχουμε είναι προσαρμοσμένες στην προσωπικότητα του ατόμου που θέτει τα ερωτήματα. Στη συνέχεια θα περιγράψουμε αναλυτικά πως λειτουργεί η μέθοδος του Referral web.

Ένα κοινωνικό δίκτυο όπως γνωρίζουμε μοντελοποιείται από έναν γράφο όπου οι κόμβοι αναπαριστούν μεμονωμένους χρήστες και οι ακμές μεταξύ των κόμβων αναπαριστούν μια σχέση μεταξύ των δυο χρηστών που αναπαρίστανται από τους κόμβους αυτούς.

Υπάρχουν αρκετοί τρόποι για να αποφασίσουμε μεταξύ ποιον κόμβων υπάρχουν ακμές δηλαδή σχέσεις. Ο ποιο απλός είναι αν χρησιμοποιήσουμε ένα είδη υπάρχον γράφο όπως ο γράφος που αναπαριστά το κοινωνικό δίκτυο του Facebook. Τις περισσότερες φορές ένας τέτοιος γράφος δεν θα μας δίνεται όποτε θα πρέπει να τον δημιουργήσουμε Η πιο άβολη τεχνική για να δημιουργήσουμε τον γράφο είναι να ζητάμε από κάθε χρήστη που σχετίζεται με το σύστημα μας να αναφέρει ρητά ποιους συνεργάτες, φίλους και συμφοιτητές έχει, οπότε ο γράφος θα δημιουργείται άμεσα. Εννοείται πως θα προτιμήσουμε τον γράφο να τον φτιάξουμε με πιο

έμμεσους τρόπους όπως στο να ανατρέξουμε στα δεδομένα των mail που έχει ένας χρήστης και να βρούμε με ποιους επικοινωνεί. Αυτή η μέθοδος έχει το αρνητικό ότι μπορεί να υπάρχουν προβλήματα με το προσωπικό απόρρητο των χρηστών.

Ένας καλύτερος τρόπος θα είναι να βασιστούμε στους συνδέσμους που βρίσκονται στα mails και τις παραπομπές που αναφέρει κάθε χρήστης σε οποιαδήποτε προσωπική του σελίδα. Κάθε φορά που ένας χρήστης έχει ένα link προς κάποιον άλλον χρήστη θα σημαίνει πως αυτοί οι δύο συσχετίζονται οπότε θα υπάρχει μια ακμή που θα τους ενώνει στον κοινωνικό γράφο.

Ένας άλλος τρόπος θα ήταν να βασιστούμε σε συνεργασίες για project που ανέλαβαν άνθρωποι από κοινού ή οργανογράμματα Τέλος μπορούμε να κοιτάξουμε για συγγραφείς σε βιβλία και δημοσιεύσεις που υπάρχουν περισσότεροι του ενός ή ακόμη και στις αναφορές που γίνονται στο τέλος κάθε βιβλίου. Η πρακτική είναι όταν δυο ονόματα συναντούνται μαζί, με κάποιον από τους τρόπους που αναφέραμε, μπορούμε τους αντίστοιχους κόμβους να τους ενώσουμε με μία ακμή.

Κάθε φορά που ένας χρήστης εγγράφεται στο σύστημα και δίνει το ονοματεπώνυμο του αμέσως ξεκινά μια διαδικασία όπου γίνεται μια αναζήτηση ατόμων με κάποιους από τους τρόπους που αναφέραμε πιο πάνω. Αυτό αντίστοιχα μπορεί να γίνει αναδρομικά σε πρώτο και δεύτερο επίπεδο για να προστεθούν έμμεσα και άλλα άτομα που μπορεί ακόμη να μην έχουν εγγραφεί στο σύστημα αλλά σχετίζονται έμμεσα με το άτομο που εγγράφεται. Με αυτόν τον τρόπο ο κοινωνικός μας γράφος μεγαλώνει σταδιακά.

Μέσα από αυτό το δίκτυο που έχουμε δημιουργήσει θα γίνεται η αναζήτηση των απαντήσεων στις ερωτήσεις που κάνουμε. Κάθε φορά που ένας εγγεγραμμένος χρήστης ρωτάει κάτι. Το σύστημα θα αναζητά τις απαντήσεις που έχουν δοθεί από τα άτομα που βρίσκονται στους γειτονικούς κόμβους. Σταδιακά θα επεκτείνεται στους πιο απομακρυσμένους κόμβους. Μια απάντηση από ένα άτομο γειτονικού κόμβου αναμένουμε να είναι πιο αξιόπιστη από μια απάντηση ενός ατόμου που αντιστοιχείται σε ένα απομακρυσμένο κόμβο. Μια ερώτηση προς το σύστημα μπορεί να έχει δύο παραμέτρους. Το θέμα για το οποίο θέλουμε να βρούμε πληροφορίες και σε τι βάθος κόμβων από το άτομο που ρωτάει θα αναζητήσουμε την απάντηση.

Στο Recommendation system που είναι βασισμένο στο Referral web μπορούμε να θέσουμε ερωτήσεις όπως "ποια είναι η σχέση μου με τον Tom Eliot" και να δούμε τι απόσταση κόμβων μας χωρίζουν ακόμη και ποια άτομα κοινών γνωστών μεσολαβούν από εμάς προς αυτόν. Μια άλλη ερώτηση είναι ποιοι συνεργάτες μου ή και ακόμη ποιοι συνεργάτες των συνεργατών μου γνωρίζουν για ένα συγκεκριμένο θέμα. Το προτέρημα αυτής της μεθόδου είναι ότι αναμένουμε ότι ο κύκλος γνωστών μας και οι γνωστοί των γνωστών μας αναμένουμε να σχετίζονται καλύτερα με το θέμα που μας απασχολεί από ότι άλλοι που μπορεί μεν να έχουν γράψει κείμενα που τα έχουν τιτλοφορήσει με λέξεις που χρησιμοποιούμε ως λέξεις κλειδιά στην αναζήτηση μας αλλά η σημασιολογική έννοια και η προσέγγιση του θέματος να είναι εντελώς διαφορετική. Η λέξη κλειδί "Windows" για έναν πολιτικό μηχανικό και τους γνωστούς του θα έχει εντελώς διαφορετικά αποτελέσματα από ότι ενός μηχανικού λογισμικού και τους γνωστούς του.

Πρέπει να ξεκαθαρίσουμε ότι το Referral web δεν έχει σκοπό να αντικαταστήσει μια

μηχανή αναζήτησης όπως είναι το google. Αλλά μέσο ενός κοινωνικού δικτύου να γίνουν οι συστάσεις που ταιριάζουν πιο πολύ σε κάθε συγκεκριμένο χρήστη.

Το recommendation system που σχεδιάζεται βάση του referral web μπορεί να χρησιμοποιήσει ένα κοινωνικό δίκτυο μεταξύ ατόμων που υπάρχουν έμμεσα χωρίς να έχουν ρητά εγγραφεί σε μια ιστοσελίδα όπως παράδειγμα γίνεται όταν μπαίνουμε στο facebook. Το κοινωνικό δίκτυο υπάρχει και βασίζεται στους γνωστούς και στις συνεργασίες που έχουν δημοσιευθεί με ποικίλους τρόπους στο διαδίκτυο.

Δεν χρειάζεται κάθε χρήστης να δηλώνει προφίλ, πεδία που τον ενδιαφέρουν και γκρουπ στα οποία ανήκει για να τα πάρουμε υπόψιν μας στην αναζήτηση των απαντήσεων. Όλα τα δεδομένα που χρειαζόμαστε βρίσκονται ήδη στο διαδίκτυο και μπορούμε να τα έχουμε με τεχνικές εξόρυξης δεδομένων (data mining). Με αυτόν τον τρόπο μπορούμε να έχουμε ένα πολύ μεγάλο κοινωνικό γράφο για να αναζητήσουμε δεδομένα. Κατά πολύ μεγαλύτερο από ότι απλούς χρήστες που εγγράφονται σε ένα κοινωνικό δίκτυο. Θεωρητικά ο κοινωνικός μας γράφος θα μπορούσε να είναι τόσο μεγάλος όσοι οι άνθρωποι για τους οποίους έχει δημοσιευθεί κάποιας μορφής συνεργασίας και συνύπαρξης με άλλους ανθρώπους στο διαδίκτυο.

Δεν υπάρχουν περιορισμοί ούτε στα θέματα στα οποία μπορούμε να δώσουμε απαντήσεις γιατί το σύστημα μας δεν εστιάζεται σε θεματικές ενότητες άλλα στις σχέσεις των ανθρώπων.

Οι συστάσεις που κάνουμε δεν είναι ανώνυμες όπως γίνεται συνήθως στα περισσότερα recommendation systems αλλά μπορούμε να χρησιμοποιήσουμε το κύρος ότι αυτή η πληροφορία που παρέχουμε προέρχεται από κάποιον γνωστό ή συνεργάτη του ατόμου που ρωτάει. Έχουμε την δυνατότητα να αποκαλύψουμε την αλυσίδα γνωστών μεταξύ του ατόμου που ρωτάει κάτι με του άτομο που έχει δημοσιεύσει την ενδεχόμενη απάντηση. Αυτό μπορεί να προσφέρει ένα μεγαλύτερο αίσθημα εμπιστοσύνης στην απάντηση που παρέχουμε.

Βιβλιογραφία:
[Henry Kautz, Bart Selman και Mehul Shah: Referal web]

4.7 Το Recommendation System που προτάθηκε και χρησιμοποιήθηκε από το CiteSeer

Το CiteSeer είναι μια μεγάλη εικονική βιβλιοθήκη με μια αποδοτική μηχανή αναζήτησης όπου μπορεί ο καθένας να αναζητήσει και να διαβάσει τίτλους δημοσιεύσεων για διαφορά επιστημονικά και ακαδημαϊκά θέματα. Το Citeseer έχει ένα document recommendation system για να προτείνει στους χρήστες του επιπλέον άρθρα. Το recommendation system που διαθέτει προτείνει άρθρα με την λογική το περιεχόμενο των άρθρων που προτείνονται να είναι ανάλογο με

το περιεχόμενο των άρθρων που έχει διαβάσει ο χρήστης.

Το recommendation system είναι Concept based που συνδυάζει memory-based και model-based χαρακτηριστικά. Ας δούμε τι σημαίνουν αυτοί οι τρεις όροι. Concept based σημαίνει ότι συγκρίνει τις ιδιότητες ενός αντικειμένου με τις ιδιότητες των αντικειμένων που έχει ήδη δει ο χρήστης στο παρελθόν. Memory based σύστημα σημαίνει ότι υπολογίζει τις συστάσεις δυναμικά εκείνη την στιγμή βασισμένο σε προηγούμενες δραστηριότητες του χρήστη. Με τον όρο Model based σύστημα εννοούμε σύστημα το οποίο χρησιμοποιεί ένα σύνολο αντικειμένων για να εκπαιδεύσει, παραμετροποιήσει ένα μοντέλο το οποίο θα κάνει τις συστάσεις.

Για κάθε χρήστη δημιουργείται ένα profile βασισμένο στα αρχεία που έχει δει. Για να δημιουργηθεί αυτό το profile χρησιμοποιήθηκε ένα σύστημα ταξινόμησης το οποίο έχει ταξινομήσει τα άρθρα σε κατηγορίες. Αναλόγως σε ποιες κατηγορίες ανήκουν τα άρθρα που έχει επιλέξει ο χρήστης, το profile του θα προσδιορίσει ότι τον ενδιαφέρουν τα αντίστοιχα θέματα. Τα άρθρα που ανήκουν σε αυτά τα θέματα θα προταθούν. Εννοείτε πως το σύστημα ταξινόμησης έχει κατηγοριοποιήσει όλα τα αρχεία που υπάρχουν στην βάση δεδομένων από πιο πριν. Ενδιαφέρον σημείο είναι ότι κάθε άρθρο μπορεί να ανήκει σε τρεις κατηγορίες και κάθε χρήστης μπορεί να τον ενδιαφέρουν θέματα που ανήκουν σε περισσότερες από μία κατηγορίες.

Το σύστημά μας αποτελείται από τρία βασικά μέρη: τον ταξινομητή (classifier), τον δημιουργό των προφίλ (profiler), και το κομμάτι που κάνει τις συστάσεις (Recommender). Θα κοιτάξουμε ποιο αναλυτικά την αρχιτεκτονική του συστήματος καθώς και το πως λειτουργεί.

4.7.1 Classifier

Όλα τα άρθρα που βρίσκονται στις βάσεις δεδομένων του CiteSeer κατηγοριοποιήθηκαν σε ένα σύνολο θεμάτων. Το σύνολο θεμάτων είχε από πιο πριν οριστεί από το ACM Computer classification system. Χρησιμοποιήθηκε μια τριών επιπέδων βάθους ιεραρχία όπου περιέχει 369 συνολικά θέματα. Η κατηγοριοποίηση έγινε σε δύο στάδια.

Αρχικά έγινε το Training στάδιο. Από τα 1.164.939 άρθρα που υπήρχαν σύνολο τα 31.121 βρέθηκαν να έχουν ήδη tags σύμφωνα με αυτά της ACM. Αυτά τα 31.121 άρθρα χρησιμοποιήθηκαν για να εκπαιδεύσουν έναν kNN classifier. Σε κάθε θέμα του Computer classification system θέλαμε να ανήκουν τουλάχιστον δέκα άρθρα. Υπήρχαν 101 θέματα με λιγότερα των δέκα άρθρων όποτε από τα 369 θέματα μας έμειναν 268.

Έπειτα περάσαμε στο Classification στάδιο. Εδώ όλα τα άρθρα που δεν είχαν tag κατηγοριοποιήθηκαν από τον kNN classifier. Για κάθε άρθρο βρήκαμε τα τρία θέματα που του ταιριάζουν περισσότερο. Παίρνοντας υπόψιν όχι μόνο τα θέματα που υπάρχουν αλλά και τον συντελεστή βαρύτητας με τον οποίο ανήκει σε κάθε θέμα. Τέλος αποθηκεύτηκε στην βάση δεδομένων για κάθε άρθρο τα θέματα που του αντιστοιχούν μαζί με τον συντελεστή όπου

αντιστοιχεί το άρθρο σε κάθε θέμα.

4.7.2 Profiler

Ο ρόλος του profiler είναι να δημιουργεί ένα προφίλ χρήστη για κάθε χρήστη του CiteSeer. Βρίσκουμε τα άρθρα που έχει δει ένας συγκεκριμένος χρήστης με το να αποθηκεύσουμε τα clicks που έχει κάνει. Με αυτό τον τρόπο δημιουργούμε ένα αρχείο με το ιστορικό επισκέψεων του χρήστη. Έπειτα δίνουμε ως είσοδο στον profiler αυτό το αρχείο. Το αρχείο αποτελείτε από μια ακολουθία από IDs όπου το κάθε ID αντιστοιχεί σε ένα άρθρο. Από το κάθε άρθρο που αντιστοιχεί σε ένα ID παίρνουμε τα θέματα στα οποία ανήκει καθώς και τον wt συντελεστή βαρύτητας καθενός θέματος. Ο συντελεστής βαρύτητας wt αναφέρεται στην σχέση μεταξύ ενός άρθρου και του αντίστοιχου θέματος στο οποίο λέμε ότι αντιστοιχεί.

Δημιουργούμε το profile του χρήστη χρησιμοποιώντας τα θέματα καθώς και τους συντελεστές βαρύτητας που αντιστοιχούν τα άρθρα που διάβασε στα θέματα αυτά. Αν το ίδιο θέμα το έχουμε πολλές φορές γιατί προφανώς το ίδιο θέμα συναντήθηκε σε πολλά άρθρα που έχουν διαβαστεί τότε οι συντελεστές βαρύτητας προστίθεται Η τελική έξοδος του profiler είναι μια λίστα από τα θέματα της ACM και των αντίστοιχων συντελεστών βαρύτητας σε φθίνουσα σειρά. Αυτός ο συντελεστής βαρύτητας δηλώνει σε τι βαθμό ο χρήστης έχει διαβάσει συνεπώς και ενδιαφέρεται για το αντίστοιχο θέμα. Ο τύπος υπολογισμού του συντελεστή βαρύτητας που έχει ο profiler είναι

$$wt(cp,j) = \Sigma wt(cp,i) \text{ για όλα τα άρθρα i του χρήστη j}$$

όπου
wt(cp,j) είναι ο συντελεστής βαρύτητας του θέματος cp στο προφίλ του χρήστη j
wt(cp,i) είναι ο συντελεστής βαρύτητας του θέματος cp στο άρθρο i

4.7.3 Recommender

Τελευταίο στάδιο είναι το στάδιο του recommender. Ο recommender παίρνει ως είσοδο το αρχείο που έδωσε ως έξοδο ο Profiler και βγάζει ως έξοδο ένα σύνολο άρθρων που θα ενδιαφέρουν τον χρήστη. Για κάθε θέμα που αναφέρει ο Profiler ο Recommender ανακτά από την βάση δεδομένων όλα τα άρθρα που σύμφωνα με τον classifier έχουν το ίδιο θέμα. Με αυτό τον

τρόπο δημιουργείται μια μακροσκελής λίστα με πολλά υποψήφια άρθρα, τα τρία θέματα στα οποία ανήκουν καθώς και τους αντίστοιχους συντελεστές βαρύτητας. Για κάθε άρθρο υπολογίζουμε ένα τελικό βάρος που προκύπτει συναρτήσει του προφίλ του χρήστη και τους συντελεστές βαρύτητας του άρθρου. Όσα άρθρα έχουν τον μεγαλύτερο τελικό συντελεστή βαρύτητας θα είναι αυτά που τελικά θα προταθούν. Οι τελικοί συντελεστές βαρύτητας υπολογίζονται από τον τύπο

wt(i,j)=wt(cp,j)*wt(cp,i)

όπου
wt(i,j) ο συντελεστής βαρύτητας του άρθρου i για τον χρήστη j
wt(cp,j) ο συντελεστής βαρύτητας του θέματος cp στο προφίλ του χρήστη j
wt(cp,i) ο συντελεστής βαρύτητας του θέματος cp στο άρθρο i

Βιβλιογραφία:
[Conceptual Recommender system for CiteSeer Ajith Kodakateri Susan Gauch Hiep Luong Joshua Eno]

4.8 Recommendation System βασισμένο σε Ανθρώπους και Ετικέτες

Θα αναλύσουμε ένα recommendation system που ανακτά δεδομένα από blogs, bookmarks, κοινωνικά δίκτυα, wikis και αρχεία που διαμοιράζονται στο internet για να πραγματοποιήσει συστάσεις.

Η μέθοδος μας θα βασιστεί στους ανθρώπους που κάνουν χρήση αυτών των μέσων καθώς και στις ετικέτες που έχουν όλα τα αντικείμενα που υπάρχουν. Έχουμε φροντίσει από πριν κάθε αντικείμενο που είναι υποψήφιο για να συσταθεί να συνοδεύεται από μια περιγραφή η οποία εμπερικλείει ετικέτες που το προσδιορίζουν καθώς και κατηγορίες ανθρώπων στους οποίους απευθύνεται. Θα μαζέψουμε από όσες περισσότερες πήγες μπορούμε πληροφορίες για τις σχέσεις που υπάρχουν μεταξύ ανθρώπων, ετικετών και αντικειμένων. Βασισμένοι σε αυτές τις σχέσεις θα προτείνουμε αντικείμενα σε χρήστες όπου το κάθε αντικείμενο χαρακτηρίζεται από μια ετικέτα που είναι η ίδια ετικέτα που δηλώνει τα ενδιαφέροντα του χρήστη.

Σε αυτή την μέθοδο χρησιμοποιούμε πληροφορίες και δεδομένα τα οποία είναι δημόσια διαθέσιμα και δεν χρειάζεται να προτρέψουμε τον χρήστη να εισαγάγει δεδομένα με σκοπό να του γίνουν συστάσεις. Στο άλλο άκρο κάποιες μέθοδοι μπορούν να ζητούν από τους χρήστες να ψηφίσουν για το πόσο ενδιαφέρονται για διάφορα αντικείμενα. Αυτό ευτυχώς αποφεύγεται. Επίσης δεν τίθενται θέμα προσβολής του απόρρητου διότι όπως είπαμε τα δεδομένα βρίσκονται

ήδη διαθέσιμα σε κοινή προβολή στο internet. Το θέμα μας είναι πως θα τα εντοπίσουμε - ανακτήσουμε, να τα συνδυάσουμε, να τα επεξεργαστούμε και να παράγουμε τις σωστές συστάσεις. Δεν τίθεται το πρόβλημα άστοχων συστάσεων στους καινούριους χρήστες γιατί στην αρχή δεν υπάρχουν για αυτούς δεδομένα.

Η καλή απόδοση του συστήματος στηρίζεται στην σωστή επιλογή και χρήση των ετικετών και όχι στην αποδοτικότητα κάποιον αποδοτικών ή μη αποδοτικών υπολογιστικών μεθόδων. Τελευταίο θετικό που θα αναφέρουμε είναι πως άνθρωποι και ετικέτες μπορούν εξίσου να συμβάλουν στο να συσταθούν οποιουδήποτε είδους δεδομένα είτε είναι μουσική είτε βίντεο είτε φωτογραφίες.

Υπάρχουν τρεις τρόποι για να ανακτήσουμε σχέσεις χρηστών - ετικετών που βασίζονται σε πληροφορίες που υπάρχουν δημόσια διαθέσιμες. Η πιο απλή θα είναι να κάνουμε άμεση χρήση ετικετών από οποιαδήποτε πηγή και αν προέρχονται. Δεύτερη επιλογή είναι να υπάρχει έμμεση σύνδεση μεταξύ ενός χρήστη και μιας ετικέτας διαμέσου ενός αντικειμένου. Δηλαδή ετικέτες να συσχετίζονται με τα κείμενα που περιγράφουν τα αντικείμενα και τα αντικείμενα αυτά με την σειρά τους να συσχετίζονται με χρήστες. Τρίτη επιλογή είναι να έχουμε ετικέτες που αντιστοιχούν σε χρήστες οι οποίες έχουν επιλεχθεί από άλλους χρήστες. Αυτό πραγματοποιείται μέσω του ότι οι χρήστες μπορούν να εφαρμόσουν μια ετικέτα ο ένας στον άλλον.

Ο αλγόριθμος που θα παρουσιάσουμε συναθροίζει σχέσεις μεταξύ ανθρώπων, αντικειμένων, και ετικετών. Για κάθε χρήστη θα δημιουργήσουμε λίστες όπου θα έχουμε ανθρώπους και ετικέτες που σχετίζονται με αυτόν καθώς και έναν συντελεστή βαρύτητας που θα υποδηλώνει των βαθμό του συσχετισμού που υπάρχει. Με αυτόν τον τρόπο δημιουργούμε ένα προσωπικό προφίλ για κάθε χρήστη. Επίσης για κάθε αντικείμενο υπάρχουν λίστες με άτομα και ετικέτες που συσχετίζονται αναμεταξύ τους μαζί με τον αντίστοιχο συντελεστή βαρύτητας. Ο αλγόριθμος συστήνει σε έναν χρήστη αντικείμενα τα οποία συσχετίζονται με τους ανθρώπους και τις ετικέτες που υπάρχουν μέσα στο προφίλ του. Το γιατί επιλέχθηκαν αυτά που επιλέχθηκαν να συσταθούν μπορούμε να το κατανοήσουμε μέσω δύο επιπέδων. Στο πρώτο επίπεδο παρουσιάζονται όλοι οι άνθρωποι και οι ετικέτες που παρήγαγαν τα αντικείμενα που συστάθηκαν. Σε ένα δεύτερο επίπεδο για κάθε όνομα ενός συγκεκριμένου ατόμου και κάθε ετικέτα ενός συγκεκριμένου αντικειμένου μπορούμε να δούμε την σχέση του με το αντικείμενο που συστάθηκε.

Στην συνέχεια θα περιγράψουμε αναλυτικά πως λειτουργεί ένα αθροιστικό recommendation system που βασίζεται σε σχέσεις μεταξύ ανθρώπων, αντικειμένων και ετικετών. Στην προσέγγιση που κάνουμε όλες οι οντότητες που υπάρχουν από τις δικτυακές κοινότητες, τα προφίλ, τις δραστηριότητες, τα blogs, τα bookmarks, τα αρχεία, τα wikis είναι αναζητήσιμες και ανακτήσιμες. Σε αυτή την μέθοδο έχει κυρίαρχο λόγο ένας πίνακας που αναπαριστά σχέσεις οντότητας προς οντότητα. Είναι δηλαδή ένας κεντρικός πίνακας που περιέχει όλη την πληροφορία του ποια οντότητα από αυτές που περιγράψαμε πιο πάνω σχετίζεται με κάποιες άλλες οντότητες και σε ποιο βαθμό. Κάθε κελί του πίνακα έχει έναν σταθμισμένο αριθμό βαρύτητας που προσδιορίζει αν δύο οντότητες έχουν σχέση και τον βαθμό της σχέσης που υπάρχει αναμεταξύ

τους.

Ο αριθμός βαρύτητας που προσδιορίζει την σχέση που έχουν δύο οντότητες υπολογίζεται από δύο τύπους σχέσεων. Τις άμεσες σχέσεις και τις έμμεσες σχέσεις. Άμεσες σχέσεις έχουμε στις εξής περιπτώσεις. Για έναν χρήστη όταν σχετίζεται άμεσα με έναν άλλο χρήστη είναι φίλοι, έχει κάνει tag ο ένας τον άλλον ή βρίσκονται μαζί σε ένα οργανόγραμμα (ως συνάδελφοι, προϊστάμενοι ή υπάλληλοι). Για ένα αντικείμενο όπως ένα προϊόν, ένα αρχείο ή ένα group ατόμων αν κάποιος είναι συγγραφέας του, σχολιαστής του, το έχει κάνει tag ή είναι μέλος του. Και για μια ετικέτα υπάρχει άμεση συσχέτιση με έναν χρήστη, αν εφαρμόστηκε σε αυτόν τον χρήστη από τον ίδιο ή κάποιους άλλους χρήστες προς αυτόν . Επίσης αν πρόκειται για αντικείμενο αν στο αντικείμενο αυτό έχει μπει η ετικέτα. Τέλος να αναφέρουμε οτι δεν υπάρχει άμεση συσχέτιση μεταξύ ενός αντικειμένου με ένα αντικείμενο και μιας ετικέτας με μία ετικέτα.

Δύο οντότητες συσχετίζονται έμμεσα στην περίπτωση που και οι δύο είναι άμεσα συσχετισμένες με μιαν άλλη κοινή οντότητα. Για παράδειγμα δύο χρήστες είναι έμμεσα συσχετισμένοι αν και οι δύο είναι άμεσα συσχετισμένοι με έναν κοινό χρήστη όπως το να έχουν τον ίδιο διευθυντή.

Το επόμενο σημαντικό κομμάτι που θα αναλύσουμε έχει να κάνει με το προφίλ των χρηστών. Κάθε φορά που ένας χρήστης u εισέρχεται, το προφίλ του P(u) θα περνάει ως είσοδος στο recommendation system. Τα προφίλ χρειάζονται για να εξατομικεύσουν τα προϊόντα που θα συσταθούν στα ενδιαφέροντα του χρήστη u. Στην εκδοχή που θα περιγράψουμε ένα προφίλ χρήστη u αποτελείται από 30 χρήστες και 30 ετικέτες T(u) με τις οποίες σχετίζεται.

Σε ένα προφίλ χρήστη το ποιοι άλλοι χρήστες υπάρχουν και σχετίζονται μαζί του θα είναι κάτι που θα προσδιορισθεί παίρνοντας υπόψιν τις άμεσες και έμμεσες σχέσεις χρήστη προς χρήστη που αναφέραμε πιο πριν. Για να αποφασίσουμε ποιοι χρήστες θα επιλεγούν να είναι σε αυτούς τους 30 του προφίλ θα ακολουθήσουμε την εξής διαδικασία. Για κάθε άμεση σχέση μεταξύ δύο χρηστών προσθέτουμε τον αριθμό ένα για την σχέση που έχουν αναμεταξύ τους. Ενώ για κάθε έμμεση σχέση μεταξύ δύο χρηστών αθροίζουμε έναν αριθμό που βρίσκεται στο εύρος μεταξύ του μηδενός και του ένα. Το ποια ακριβώς θα είναι η τιμή του αριθμού αυτού εξαρτάται από παραμέτρους όπως του πόσα πολλά κοινά αρχεία έχουν αναμεταξύ τους ή πόσα άρθρα έχουν γράψει μαζί.

Μια ακόμη διάκριση που πρέπει να κάνουμε είναι μεταξύ των σχέσεων οικειότητας (familiarity relationships) και των σχέσεων ομοιότητας (similarity relationships). Σχέσεις οικειότητας είναι οι σχέσεις όπου ο ένας χρήστης γνωρίζει τον άλλον. Περιλαμβάνει όλες τις άμεσες σχέσεις χρήστη προς χρήστη, όπως τις περιγράψαμε πιο πάνω, και δύο περιπτώσεις έμμεσων σχέσεων. Όταν δύο χρήστες είναι συγγραφείς ενός άρθρου και όταν έχουν τον ίδιο διευθυντή.

Σχέσεις ομοιότητας έχουμε όταν δύο χρήστες έχουν παρόμοια κοινωνική δραστηριότητα. Εμπεριέχονται όλες οι υπόλοιπες άμεσες σχέσεις όπως το να έχουν καταχωρήσει και οι δύο σχόλιο στο ίδιο blog, να βρίσκονται ως μέλη στην ίδια κοινωνική ομάδα ή να έχουν κάνει χρήση την ίδια ετικέτα. Οι σχέσεις οικειότητας είναι κατά πολύ πιο σημαντικές από τις σχέσεις ομοιότητας στο να παραχθούν συστάσεις που θα ενδιαφέρουν τον αποδέκτη τους, χωρίς όμως

αυτό να αποκλείει την αξία και των σχέσεων ομοιότητας. Για να δείξουμε αυτήν την προτίμηση των σχέσεων οικειότητας έναντι των σχέσεων ομοιότητας, θα πολλαπλασιάσουμε στις σχέσεων ομοιότητας τον αριθμό που αντιστοιχεί στην βαρύτητα της σχέσεις μεταξύ δυο χρηστών με το 1/3. Τέλος από όλες τις σχέσεις που υπάρχουν χρήστη προς χρήστη θα επιλέξουμε τις 30 με τον μεγαλύτερο βαθμό βαρύτητας και θα τις καταχωρίσουμε στο προφίλ του χρήστη.

Για να εξάγουμε τις ετικέτες που σχετίζονται με κάποιον χρήστη θα κάνουμε τις εξής τρεις διακρίσεις ετικετών. Οι ετικέτες που χρησιμοποίησαν ο ίδιος ο χρήστης προς άλλα αντικείμενα ή άτομα. Ετικέτες που χρησιμοποίησαν άλλοι χρήστες προς αυτόν. Τρίτη κατηγορία είναι οι έμμεσες ετικέτες. Ετικέτες δηλαδή που χρησιμοποιήθηκαν σε αντικείμενα που απλώς σχετίζονται άμεσα με τους χρήστες. Αντίστοιχα θα επιλέξουμε τις τριάντα ετικέτες που σχετίζονται πιο πολύ με τον χρήστη και θα τις καταχωρήσουμε στο προφίλ του.

Από την στιγμή που έχουμε το προφίλ του χρήστη P(u)=(N(u),T(u)) είμαστε σε θέση να κάνουμε συστάσεις στον χρήστη. Οι συστάσεις που θα πραγματοποιηθούν σε έναν χρήστη σχετίζονται με τους χρήστες και τα αντικείμενα που αναφέρονται στο προφίλ του κατά τον εξής τρόπο. Κάθε αντικείμενο i θα έχει έναν βαθμό του κατά πόσο ταιριάζει να συσταθεί στον χρήστη u που προκύπτει από τον τύπο

$$RS(u,i) = e^{-\alpha d(i)} \cdot [\beta \sum_{v \in N(u)} w(u,v) \cdot w(v,i) + (1-\beta) \sum_{t \in T(u)} w(u,t) \cdot w(t,i)]$$

όπου

d(i) ο αριθμός των ημερών που έχουν μεσολαβήσει από την δημιουργία του αντικειμένου i

a ένας συντελεστής απόσβεσης συνήθως τίθεται ίσος με 0.025

β είναι μια παράμετρος για να καθορίζει την βαρύτητα των συστάσεων με βάση τους χρήστες σε σχέση με τις συστάσεις με βάση τις ετικέτες.

w(u,v) είναι ο αριθμός που προσδιορίζει τον βαθμό σχέσης μεταξύ του χρήστη u με τον χρήστη v.

w(u,t) είναι ο αριθμός που προσδιορίζει τον βαθμό σχέσης μεταξύ του χρήστη u με την ετικέτα t.

Το RS(u,i) που υπολογίζουμε από τον παρά πάνω τύπο θα μας δώσει για τα υποψήφια αντικείμενα που είναι να συσταθούν μια τιμή που θα υποδηλώσει το πόσο πολύ είναι κατάλληλο για να επιλεχθεί το αντικείμενο ώστε να συσταθεί. Εννοείται πως εμείς θα επιλέξουμε να συστήσουμε τα αντικείμενα με το μεγαλύτερο RS. Καθώς και ότι δεν συστήσουμε αντικείμενα που είναι άμεσα συσχετισμένα με τον χρήστη. Αντικείμενα που ο χρήστης τα έχει δει, τους έχει κάνει σχόλια, τους έχει εφαρμόσει ετικέτες τα γνωρίσει ήδη και δεν χρειάζεται να του

συσταθούν. Οι παράγοντες που αυξάνουν την πιθανότητα το αντικείμενο να συσταθεί είναι οι εξής:

- Στο προφίλ ενός χρήστη πόσοι πολλοί άνθρωποι και ετικέτες είναι συσχετισμένοι με το αντικείμενο αυτό.

- Πόσο δυνατές είναι οι σχέσεις του χρήστη με τους χρήστες και τις ετικέτες που είναι στο προφίλ του.

- Πόσο δυνατές είναι οι σχέσεις του αντικειμένου που είναι προς σύσταση με τους χρήστες και τις ετικέτες που είναι στο προφίλ του χρήστη.

- Πόσο καινούριο είναι το αντικείμενο.

Βιβλιογραφία:
[Social Media Recommendation based on People and Tags Ido Guy, Naama Zwerdling, Inbal Ronen, David Carmel, Erel Uziel]

4.9 FAB Ένα Recommendation System που συνδυάζει τις αρχές των Content-Based με των Collaborative Συστάσεων.

Το FAB recommendation system είναι το σύστημα συστάσεων που χρησιμοποιείται στην ψηφιακή βιβλιοθήκη του πανεπιστήμιου του Stanford και έχει χρησιμοποιηθεί σε πολλές επιχειρησιακές εκδόσεις από τον Δεκέμβριο του 1994. Αυτό που το κάνει να ξεχωρίζει είναι ότι συνδυάζει την content-based μέθοδο, ότι δηλαδή συστήνει αντικείμενα παρόμοια με αυτά που άρεσαν στον χρήστη στο παρελθόν, με τις collaborative συστάσεις. Δηλαδή για τον χρήστη που μας ενδιαφέρει βρίσκει παρόμοιους χρήστες με αυτόν και έπειτα του συστήνει τα αντικείμενα που τους αρέσουν. Στα θετικά αυτής της μεθόδου μπορούν να προστεθούν η ικανότητα της να αντιμετωπίσει τον συνεχώς αυξανόμενο αριθμό χρηστών και τον συνεχώς αυξανόμενο αριθμό πληροφοριών. Καθώς επίσης και ότι το σύστημα μπορεί να αναγνωρίσει αυτοματοποιημένα τις διάφορες ομάδες ενδιαφερόντων που προκύπτουν από τους χρήστες επιτρέποντας τους να ενημερώνονται και να επικοινωνούν

Στις παρακάτω παραγράφους, αν και τα έχουμε ήδη αναφέρει ξεχωριστά, θα ξαναδώσουμε τα βασικά στοιχεία από την content based και collaborative προσέγγιση, στα οποία

θα βασιστούμε για να οικοδομηθεί η αρχιτεκτονική της μεθόδου Fab. Ταυτόχρονα θα περιγράψουμε τα προβλήματα που υπάρχουν στις δύο μεθόδους και τα οποία θα μπορέσουν να λυθούν συνδυάζοντας τες.

Σε ένα content-based recommendation system οι συστάσεις προέρχονται αποκλειστικά από το προφίλ του χρήστη. Το προφίλ του χρήστη έχει δημιουργηθεί βασιζόμενο σε αντικείμενα τα οποία ο χρήστης έχει επιλέξει και έμμεσα ή άμεσα έχει ψηφίσει ότι τον ενδιαφέρουν. Οι πληροφορίες για ένα αντικείμενο συγκρίνονται στο προφίλ του χρήστη. Όσο πιο σχετικό είναι το προφίλ του χρήστη με το αντικείμενο που είναι υποψήφιο να συσταθεί τόσες πιο πολλές είναι οι πιθανότητες το αντικείμενο να συσταθεί. Χρησιμοποιούνται συντελεστές βαρύτητας σε όρους που χρησιμοποιούνται για να περιγράψουν αντικείμενα ή για κείμενα. Όσο πιο πολύ ενδιαφέρουν οι όροι τον χρήστη τόσο ποιο μεγάλοι θα είναι αυτοί οι συντελεστές βαρύτητας στο προφίλ του.

Δύο αδυναμίες που καλείτε να ξεπεράσει η μέθοδος Fab και έχουν τα content based recommendation systems είναι το πρόβλημα της υπερεξειδίκευσης δηλαδή ότι το προφίλ του χρήστη μπορεί να έχει εξειδικευθεί στα χαρακτηριστικά μιας συγκεκριμένης θεματολογίας και δεν μπορεί να συστήσει διαφορετικά προϊόντα. Θα μπορούσε αυτό να λυθεί εισαγάγοντας ένα ποσοστό τυχαιότητας στις επιλογές που γίνονται. Η δεύτερη αδυναμία που καλείται να ξεπεραστεί είναι όπως αναφέραμε πιο πριν, για να δημιουργηθεί το προφίλ του χρήστη, ο χρήστης θα πρέπει να ψηφίσει για τα αντικείμενα που τον ενδιαφέρουν. Αν αυτό γίνει άμεσα το σύστημα θα φανεί κουραστικό. Για αυτό θα πρέπει να επιλεχθούν μέθοδοι που θα επιτρέπουν αυτή η επιλογή και ψηφοφορία των αντικειμένων να γίνεται έμμεσα.

Σε ένα collaborative Recommendation σύστημα θα προταθούν σε κάποιον αντικείμενα τα οποία άλλοι παρόμοιοι χρήστες τα είδαν και τους άρεσαν. Αν πριν βρίσκαμε την ομοιότητα μεταξύ αντικειμένων, εδώ θα βρούμε την ομοιότητα μεταξύ χρηστών. Το πόσοι πολύ παρόμοιοι χρήστες είδαν τα αντικείμενα κατά πόσο είναι παρόμοιοι αυτοί οι χρήστες με αυτόν που μας ενδιαφέρει και πόσο πολύ ενδιαφέρει αυτό το αντικείμενο καθέναν από αυτούς τους χρήστες είναι κάτι που θα συναθροιστεί για κάθε αντικείμενο. Τα αντικείμενα που συναθροίζουν τον μεγαλύτερο αριθμό θα είναι αυτά που θα συσταθούν. Δεν μας ενδιαφέρουν καθόλου ιδιότητες και χαρακτηριστικά του καθενός αντικειμένου μόνο ένα id είναι αρκετό για να το αναγνωρίσουμε

Τα προβλήματα που έχουν τα collaborative Recommendation συστήματα και καλείται να ξεπεράσει η υβριδική μέθοδος Fab είναι τα εξής. Όταν ένα καινούριο αντικείμενο εμφανίζεται στην βάση δεδομένων δεν μπορεί να συσταθεί γιατί δεν θα υπάρχει χρήστης που να τα έχει επιλέξει ούτως ώστε να συσταθούν και στους παρόμοιους του. Ένα άλλο πρόβλημα συναντιέται αν ο αριθμός των χρηστών είναι μικρός σε σχέση με το πλίθος των αντικειμένων που υπάρχουν. Αυτό σημαίνει πως οι επιλογές και οι ψήφοι για τα αντικείμενα θα είναι λίγες και να μην έχουμε ένα αντιπροσωπευτικό δήγμα για να βγάλουμε συμπεράσματα. Τέλος προβλήματα υπάρχουν όταν υπάρχουν ιδιαίτεροι χρήστες με εξεζητημένες προτιμήσεις. Μπορεί για αυτούς να μην υπάρχουν άλλοι παρόμοιοι χρήστες για να βασιστούμε και να γίνουν οι αντίστοιχες συστάσεις.

Στην υβριδική υλοποίηση του recommendation system Fab θα έχουμε προφίλ χρηστών βασισμένα στην content based προσέγγιση. Θα συγκρίνουμε αυτά τα προφίλ για να βρούμε

παρόμοιους χρήστες και να κάνουνε collaborative συστάσεις. Από την σωστή κατασκευή των προφίλ κρίνεται η επιτυχία του συστήματος. Τα σωστά προφίλ ενισχύουν τον content-based χαρακτήρα της μεθόδου αφού επιβεβαιώνουν ότι οι συστάσεις είναι κατάλληλες. Καθώς επίσης Ενισχύουν και τον Collaborative χαρακτήρα της γιατί επιβεβαιώνουν ότι χρήστες με παρόμοια προφίλ είναι πράγματι παρόμοιοι.

Τα αντικείμενα που θα προταθούν στους χρήστες προκύπτουν με δύο τρόπους. Όταν ένα αντικείμενο παίρνει μια υψηλή βαθμολογία αντιστοίχησης λόγο του ότι η κατηγορία του ταιριάζει πολύ με τα θέματα που υπάρχουν στο προφίλ του χρήστη. Ή πάλι όταν βαθμολογείται ψηλά, άλλα αυτήν την φορά λόγο του ότι άρεσε και χρησιμοποιήθηκε από έναν χρήστη με παρόμοιο προφίλ.

Η διαδικασία των συστάσεων μπορεί να χωριστεί σε δύο στάδια. Στο πρώτο στάδιο θα γίνει μια συλλογή από αντικείμενα για να δημιουργηθεί μια βάση δεδομένων ή ένας πλήρης κατάλογος τον οποίο έπειτα θα μπορούμε να τον διαχειριστούμε. Στο επόμενο στάδιο θα επιλέξουμε τα κατάλληλα αντικείμενα από αυτήν την βάση δεδομένων για τους αντίστοιχους χρήστες.

Το στάδιο της συλλογής αντικειμένων πολλές φορές είναι τετριμμένο και απλό είτε γιατί είναι εύκολη η κατηγοριοποίηση των αντικειμένων ή γιατί μας έχουν είδη δοθεί τα αντικείμενα διαφοροποιημένα από ποιο πριν. Αν τα αντικείμενα που μας ενδιαφέρουν βρίσκονται διάσπαρτα σε κάποια μέρη στον ιστό το πρόβλημα για τον σχεδιαστή του συστήματος είναι αρκετά δύσκολο. Αυτό που θα πρέπει να γίνει είναι να μαζέψουμε σχετικές σελίδες ανά θέματα. Η ποικιλία των θεμάτων θα βρεθεί από ειδικά συστήματα που βρίσκουν τα ενδιαφέροντα των χρηστών και θα πρέπει να είναι σχετικά μικρή.

Αφού γίνει η κατηγοριοποίηση των αντικειμένων σε θέματα θα ξεκινήσει το στάδιο επιλογής όπου βασιζόμενοι στο προφίλ των χρηστών και στο τι ενδιαφέρει τους παρόμοιους χρήστες θα συνδεθούν χρήστες με θέματα. Ένα θέμα μπορεί να ενδιαφέρει πολλούς χρήστες και πολλοί χρήστες να ενδιαφέρονται για ένα θέμα.

Υπάρχουν τρεις βασικές δομικές μονάδες. Οι πράκτορες συλλογής (collection agents), οι πράκτορες επιλογής (selection agents) και ο κεντρικός κατευθυντής (central routers). Καθένας από αυτούς τους δύο πράκτορες έχει ένα προφίλ που χαρακτηρίζεται από συγκεκριμένες λέξεις όπως αυτές βρέθηκαν στον ιστό και επιλέχθηκαν έναντι των άλλων γιατί είχαν την πιο υψηλή σημασία.

Οι πράκτορες συλλογής είναι το κομμάτι που αναζητά μέσα στον ιστό και βρίσκει σελίδες για ένα συγκεκριμένο θέμα. Ο πράκτορας συλλογής αποστέλλει ερωτήματα σε διάφορες εμπορικές μηχανές αναζήτησης που έχουν αναλυτικά ευρετήρια όρων, καθώς επίσης επισκέπτεται τους συνδέσμους που έχει κάθε ιστοσελίδα με σκοπό να βρει περισσότερες σελίδες του ίδιου θέματος. Για κάθε θέμα υπάρχει και ένας πράκτορας συλλογής. Το συγκεκριμένο αυτό θέμα μπορεί με την πάροδο του χρόνου να αλλάζει δυναμικά ομάδες χρηστών στις οποίες απευθύνεται. Αν δημιουργηθεί ένας πράκτορας συλλογής όπου το θέμα του με την πάροδο του χρόνου δεν έχει ευρεία αποδοχή από τους χρήστες θα διαγραφεί.

Οι πράκτορες επιλογής είναι το κομμάτι που που βρίσκει και αντιστοιχεί ποιες σελίδες θα

ενδιέφεραν έναν συγκεκριμένο χρήστη. Κάθε σελίδα θα ανήκει σε κάποιο θέμα οπότε ένας πράκτορας επιλογής μπορεί να συνεργάζεται και να διαχειρίζεται πολλούς πράκτορες συλλογής από τους οποίους θα πάρει τις σελίδες που θα παρουσιάσει. Ο ρόλος των κεντρικών κατευθυντών είναι πιο περίπλοκος και θα τον εξετάσουμε αναλυτικά.

Ο κεντρικός κατευθυντής δέχεται τις σελίδες που βρέθηκαν από τους πράκτορες συλλογής και ανακατευθύνει στους χρήστες αυτές που σύμφωνα με το προφίλ τους θα τους ενδιαφέρουν περισσότερο. Πριν ξεκινήσει η διαδικασία της ανακατεύθυνσης στον κεντρικό κατευθυντή θα λάβουν μέρος οι πράκτορες επιλογής. Είναι ευθύνη του πράκτορα επιλογής να κάνει ένα ξεδιάλεγμα των σελίδων απορρίπτοντας σελίδες που έχουν ήδη δει οι χρήστες και φροντίζοντας από κάθε ιστότοπο να υπάρχουν περισσότερες από μια σελίδες.

Αφού συσταθούν οι σελίδες ζητείται από τον χρήστη να βαθμολογήσει σε μια σκάλα 7 βαθμών το πόσο ενδιαφέρουσες ήταν οι σελίδες που του συστάθηκαν. Αυτό αν και χαλάει την διαφάνεια του συστήματος βελτιώνει πολύ την αποδοτικότητα του. Αυτή η βαθμολογία θα ανανεώσει τα προφίλ των πρακτόρων επιλογής για κάθε χρήστη. Επίσης όλες οι σελίδες στις οποίες ένα χρήστης έβαλε πολύ υψηλή βαθμολογία θα περάσουν κατευθείαν στα προφίλ των πρακτόρων επιλογής όλων των παρόμοιων χρηστών του.

Βιβλιογραφία:
[FAB: content-based, collaborative recommendation από τους Marko Balabanovic και Yoav Shoham]

4.10 PHOAKS ένα Σύστημα Συστάσεων που Διαμοιράζει Συστάσεις μεταξύ των Χρηστών του.

Το Phoaks είναι ένα recommendation system που επαναχρησιμοποιεί συστάσεις που έχουν κάνει χρήστες σε online συζητήσεις τους. Τα αρχικά PHOAKS προέρχονται από τις λέξεις People Helping One Another Know Stuff. Χρησιμοποιεί έναν αυτοματοποιημένο τρόπο να αναγνωρίζει, να αντιστοιχίζει και να διανέμει συστάσεις. Οι εμπειρικές παρατηρήσεις που έγιναν από τους Loren Terveen, Will Hill, Brian Amento, David McDonald και Josh Creter πάνω στο παγκόσμιο δίκτυο συζητήσεων usenet και μας οδήγησαν στην μέθοδο PHOAKS είναι οι εξής:

- Ένα ποσοστό 90% των συστάσεων που γίνονται είναι υπολογιστικά αναγνωρίσιμες.
- Ένα ποσοστό 23% των μηνυμάτων που γράφονται στο usenet είναι πηγές στο Internet.

Από αυτές ένα 30% είναι συστάσεις.

- Κάποιες από τις πηγές στο Internet συστήνονται από περισσότερους των δύο χρηστών.

Όταν μια πηγή συστήνεται αρκετές φορές αναμένουμε ότι θα αναφέρει κάτι σημαντικό για το θέμα που διαπραγματεύεται. Γενικώς μπορούμε να υιοθετήσουμε την άποψη ότι το πόσο συχνά συστήνεται μια πηγή είναι κριτήριο για τόσο πόσο σημαντική είναι. Επίσης μια αντιστοίχηση και σύγκριση των πηγών που βρίσκουμε στο usenet με τις πηγές που βρίσκουμε στα διάφορα FAQs (Λίστες που έχουν απαντήσεις από ειδικούς σε συχνά ερωτήματα) στο Internet μπορεί να μας δώσει ένα ακόμη ισχυρό κριτήριο για το ποιες πηγές είναι σημαντικές.

Ας δούμε αναλυτικά πως λειτουργεί η μέθοδος. Οι αναγνώστες των ειδήσεων του Usenet συχνά κάνουν post με με τις εντυπώσεις του και της απόψεις τους γύρο από το θέμα που διαβάζουν. Πολύ συχνά τα σχόλια τους συνοδεύοντα και με έναν σύνδεσμο URL που παραπέμπει σε κάποια άλλη σελίδα που συσχετίζεται με το θέμα που διαβάζουν και πολύ πιθανόν να ενδιαφέρει και άλλους χρήστες. Αυτός ο σύνδεσμος είναι υποψήφιος να χρησιμοποιηθεί ως σύσταση. Για να θεωρηθεί και να χρησιμοποιηθεί ως σύσταση πρέπει να ικανοποιεί κάποια κριτήρια. Παρακάτω παρατίθενται τα κριτήρια.

- Το ίδιο μήνυμα δεν πρέπει να είναι δημοσιευμένο σε πολλά διαφορετικά θέματα. Αν συμβαίνει αυτό κατά πάσα πιθανότητα θα είναι κάτι αρκετά γενικό και δεν θα αντιστοιχεί σε μια συγκεκριμένη θεματική ενότητα.

- Το URL δεν πρέπει να είναι κομμάτι της ψηφιακής υπογραφή ενός χρήστη ή ενός αρχείου που συνοδεύουν ένα μήνυμα.

- Το URL δεν πρέπει να είναι μέρος μιας παράθεσης που έρχεται ως απάντηση σε ένα προηγούμενο σχόλιο.

- Το μήνυμα που έχει το URL δεν πρέπει να έχει σημάδια (markers) ότι είναι διαφήμιση ή ανακοίνωση.

Από όσους συνδέσμους ικανοποιούν τα προηγούμενα κριτήρια αυτοί που θα επιλεχθούν να συσταθούν θα είναι αυτοί που έχουν συσταθεί περισσότερες φορές από διαφορετικούς χρήστες. Όσο ποιο πολλές φορές έχει αναφερθεί ένας σύνδεσμος σε ένα θέμα τόσο πιο πολύ αξιόλογος και κατάλληλος θα θεωρηθεί ούτως ώστε να συσταθεί.

Το σύστημα συστάσεων PHOAKS αποτελείται από τρία στάδια. Αρχικά αναζητά σε όλα

τα μηνύματα που έχουν δημοσιοποιήσει οι χρήστες κάποιες συμβολοακολουθίες οι οποίες υποδηλώνουν ότι υπάρχει κάποιος σύνδεσμος. Τέτοιες συμβολοακολουθίες θα μπορούσαν να είναι οι "www" και "http://". Αν ικανοποιεί τα προηγούμενα κριτήρια εξάγουμε τον σύνδεσμο αυτό μαζί με το κείμενο που το περιβάλει. Από το κείμενο μπορούμε να έχουμε κάποιες περισσότερες πληροφορίες για τον συγκεκριμένο σύνδεσμο.

Το επόμενο στάδιο είναι να κατατάξουμε τους συνδέσμους αυτούς σε κατηγορίες. Αυτό μπορεί να γίνει με το να επεξεργαστούμε το URL. Μπορούμε να βρούμε αν είναι προσωπική ιστοσελίδα, από πια χώρα προέρχεται, αν παραπέμπει σε ένα forum ή ένα blog. Ακόμη πιο πολλά δεδομένα μπορούμε να έχουμε από το κείμενο που περιβάλει την σελίδα. Επίσης πολύ σημαντικό σε αυτήν την κατηγοριοποίηση είναι ότι το θέμα στο οποίο έγινε ανάρτηση του μηνύματος και του URL είναι το πιο πιθανό να είναι και το θέμα του URL.

Τελευταίο στάδιο είναι να αποθηκευθούν με κάποιον τρόπο τα δεδομένα ούτως ώστε να μπορούν να ανακτηθούν και έπειτα να επεξεργαστούν αν χρειαστεί. Μια βάση δεδομένων θα μπορούσε να είναι ο ιδανικός τρόπος.

Από την στιγμή που όλοι μας οι σύνδεσμοι βρίσκονται στην βάση δεδομένων μαζί με περιγραφές και τα θέματα στα οποία ανήκουν είμαστε σε θέση ανά πάσα στιγμή να προσφέρουμε συστάσεις όταν χρειαστούν. Κάθε φορά που ένας χρήστης διαβάζει δεδομένα σε έναν ιστότοπο ενός θέματος μπορούμε να θέσουμε στην βάση δεδομένων ένα querie με το θέμα που ενδιαφέρει τον χρήστη και να του παρουσιάσουμε ως συστάσεις τους αντίστοιχους συνδέσμους.

Βιβλιογραφία:
[PHOAKS: a system for sharing recommendations. Απο τους Loren Terveen, Will Hill, Brian Amento, David McDonald και Josh Creter]

4.11 Yoda Ένα recommendation system που συνδυάζει τις αρχές των Content-based με των Collaborative συστάσεων.

Όπως έχουμε πει οι δύο πιο βασικές τεχνικές σχεδιασμού Recommendation συστημάτων είναι αυτών που βασίζονται στις σχεδιαστικές αρχιτεκτονικές των content-based συστημάτων και των collaborative filtering. Και οι δύο τεχνικές έχουν θετικά και αρνητικά που τα έχουμε αναπτύξει αναλυτικά πιο πριν. Φυσικό είναι σε δεύτερο επίπεδο να αναπτυχθούν υβριδικές μέθοδοι που θα τις συνδυάζουν για να μειώσουν τα μειονεκτήματα κάθε μίας και να βελτιώσουν όσο μπορούν τα προτερήματα τους. Πριν παρουσιάσαμε το σύστημα συστάσεων FAB το οποίο καταφέρνει με έναν πολύ πετυχημένο τρόπο να τις συνδυάζει. Φυσικά η μέθοδος FAB δεν είναι η μόνη που μπορεί να συνδυάσει την content-based με την collaborative filtering αρχιτεκτονική σχεδίαση

recommendation συστημάτων. Μια άλλη προσέγγιση του θέματος έρχεται από το σύστημα συστάσεων Yoda. Αξίζει να μελετήσουμε παρακάτω πως η μέθοδος Yoda συνδυάζοντας με τον δικό της τρόπο τις δύο τεχνικές και παράγει επιτυχημένες συστάσεις.

Η μέθοδος Yoda εφαρμόζεται στον αχανή ιστό του διαδικτύου και μπορεί να προβάλει συστάσεις σε πραγματικό χρόνο καθώς ο χρήστης περιηγείται σε ιστοσελίδες ενώ έχει επεξεργαστεί και προετοιμάσει τα δεδομένα της όσο ο χρήστης ήταν offline.

Κατά την διάρκεια που οι χρήστες είναι online και περιηγούνται στον ιστό το σύστημα παρατηρεί την συμπεριφορά τους και κατατάσσει τον κάθε ένα σε ομάδες χρηστών που τους αντιπροσωπεύουν κάποιες κατηγορίες ενδιαφερόντων. Αναλόγως σε ποιες ομάδες ανήκει κάθε χρήστης και με τι συντελεστή βαρύτητας για την κάθε μία, του γίνονται οι αντίστοιχες συστάσεις. Κάθε ομάδα χρηστών έχει ένα ανάλογο σύνολο συστάσεων. Όση ώρα οι χρήστες είναι offline το σύστημα απασχολείται με το να βρει ποιες συστάσεις αρμόζουν καλύτερα σε κάθε ομάδα χρηστών με τα αντίστοιχα ενδιαφέροντα. Θα μπορούσαμε να πούμε ότι το σύστημα λειτουργεί σε δύο φάσεις. Οι οποίες είναι οι παρακάτω.

- Ταξινόμηση των χρηστών. Κατά την διάρκεια αυτής της φάσης μαζεύονται δεδομένα για την συμπεριφορά των χρηστών μέσα στον ιστό και γίνεται η κατάταξη των χρηστών σε ομάδες ενδιαφερόντων.

- Βαθμολόγηση των αντικειμένων. Κατά την διάρκεια αυτής της φάσης όλα τα ενδιαφέροντα που έχουμε βρει για κάθε χρήστη εφαρμόζονται για να βαθμονομήσουν και να κατατάξουν τα αντικείμενα. Για κάθε χρήστη δημιουργείται μια ταξινομημένη λίστα αντικειμένων. Η ταξινόμηση στην λίστα είναι σύμφωνη με το πόσο ενδιαφέρει κάθε αντικείμενο τον συγκεκριμένο χρήστη.

Ας δούμε αναλυτικά πως δομείται και λειτουργεί το σύστημα αυτό. Το recommendation σύστημα Yoda αρχικά συλλέγει δεδομένα όπως τον χρόνο που είδε ο κάθε χρήστης μια σελίδα (view time), πόσες φορές έκανε κλικ για να την επισκεφτεί (hit-count), την αλληλουχία των σελίδων που επισκέφτηκε και παράγει πληροφορίες γύρω από τα αντικείμενα που παρουσιάστηκαν. Αυτές οι πληροφορίες είναι η βάση με την οποία θα συμπεράνουμε ποια αντικείμενα ενδιαφέρουν τον κάθε χρήστη.

Για να αποθηκεύσει, να επεξεργαστεί και να αναλύσει η μέθοδος Yoda τα παραπάνω δεδομένα χρησιμοποιεί ένα ακριβές και ευέλικτο μοντέλο δομών δεδομένων. Σε αυτό το μοντέλο έχουμε ένα σύνολο από δεδομένα που αναπαριστούν ποιοι χρήστες ή ομάδες χρηστών είχαν πρόσβαση σε κάθε ιστοσελίδα. Έπειτα χρησιμοποιείται ένα μέτρο ομοιότητας για να αξιολογήσουμε κατά πόσο είναι παρόμοια τα ενδιαφέροντα και η συμπεριφορά ένας συγκεκριμένου χρήστη με τα ενδιαφέροντα και την συμπεριφορά μιας ομάδας χρηστών. Για

αυτόν τον λόγο χρησιμοποιούμε μια εκδοχή της ευκλείδειας απόστασης (Projected Pure Euclidian Distance). Για να ποσοτικοποιήσουμε το κατά πόσο διαφέρει ένας χρήστης u με τον μέσω χρήστη μιας ομάδα k ορίζουμε την συνάρτηση.

Suk = MaxDistance – TPPED(u,k) $0 \leq S \leq$ MaxDistance

όπου

MaxDistance είναι μια σταθερά που ορίζεται ως το άνω φράγμα της απόστασης

$$TPPED(u,k) = \begin{cases} PPED(u,k) & if\ PPED(u,k) \leq MaxDistance \\ MaxDistance & if\ PPED(u,k) > MaxDistance \end{cases}$$

Αφού υπολογίσουμε το S_{uk} είμαστε σε θέση να ξέρουμε κατά πόσο τα ενδιαφέροντα ενός χρήστη ταιριάζουν με τα ενδιαφέροντα μιας ομάδας για να τον κατατάξουμε σε αυτήν. Έπειτα το S_{uk} μπορεί να χρησιμοποιηθεί ως συντελεστής βαρύτητας για να δούμε πόσο σημαντικές είναι οι συστάσεις που προκύπτουν από μια ομάδα σε σχέση με τις συστάσεις που προκύπτουν από άλλες ομάδες και να προταθούν οι σημαντικότερες

Στην επόμενη φάση για κάθε χρήστη θα βαθμονομήσουμε τα αντικείμενα ούτως ώστε να τα κατατάξουμε σε μια σειρά και να συστήσουμε στον καθένα αυτά τα οποία τον ενδιαφέρουν ποιο πολύ. Αυτό θα πραγματοποιηθεί σε τρία βασικά μέρη. Μια offline διαδικασία στην οποία για κάθε ομάδα χρηστών θα βρούμε μια λίστα με τα αντικείμενα που τους ενδιαφέρουν πιο πολύ. Μια online διαδικασία όπου για κάθε χρήστη παράγονται τα πιθανά αντικείμενα που θα του συσταθούν. Τέλος έναν μηχανισμό φιλτραρίσματος ο οποίος βελτιστοποιεί από πλευράς χρονικής πολυπλοκότητας τις λίστες συστάσεων που παράγουν οι διάφορες ομάδες. Αλλά ας τα δούμε όλα αυτά λίγο ποιο λεπτομερειακά.

Η μέθοδος Yoda αναπαριστά τα ενδιαφέροντα των χρηστών σε κάθε ομάδα ενδιαφερόντων ως ένα σύνολο ιδιοτήτων που η κάθε ιδιότητα αναλόγως με το πόσο σημαντική είναι έχει την αντίστοιχη βαρύτητα. Οι διάφοροι χρήστες που συνιστούν μια ομάδα χρηστών έχουν επιλέξει και έχουν επισκεφτεί ένα σύνολο από αντικείμενα. Τα αντικείμενα που είναι ποιο δημοφιλή από τους χρήστες θα είναι αυτά που θα καθορίσουν τις ιδιότητες της ομάδας.

Για κάθε χρήστη υπάρχει μια λίστα με τα αντικείμενα που επισκέφτηκε b_u. Για κάθε ομάδα χρηστών υπάρχει μια λίστα με τα αντικείμενα που επισκέφτηκαν όλοι οι χρήστες μέλη της B_k. Από αυτά ακολουθούμε μια διαδικασία να επιλέξουμε τα πιο δημοφιλή και αντιπροσωπευτικά για να τα ορίσουμε ως τα αντικείμενα που θα καθορίσουν τις ιδιότητες της ομάδας $F_p(k)$. Παρακάτω παραθέτουμε τους τύπους όπου γίνεται ο υπολογισμός των ιδιοτήτων της ομάδας χρηστών.

Με την μέθοδο Yoda μπορούμε να αξιολογήσουμε την προτίμηση $u_k(i)$ του αντικειμένου i από την ομάδα χρηστών k. Αυτό γίνεται με μια συνάρτηση που μετράει και ποσοτικοποιεί την

ομοιότητα μεταξύ των υψηλότερων προτιμήσεων της ομάδας k με ένα συγκεκριμένο αντικείμενο. Αυτό προκύπτει από τους τύπους:

Definition *User browse-list*, b_u, and *cluster browse-list*, B_k, are defined as:

$$b_u = \{i \mid i \in I, \text{“}i\text{” is visited by } u \in U\}$$
$$B_k = \bigcup_{u \in k} b_u$$

where U is the training set of users. The voting procedure extracts the favorite PV, $F_p(k)$, corresponding to property p for cluster k as follows[1]:

$$C_{p,f}(k) = \|\{i \mid i \in B_k, \tilde{p}_i = f\}\|$$
$$F_p(k) = fmax\{f \mid f \in \varphi, C_{p,f}(k) = \max_{\forall f' \in \varphi} \{C_{p,f'}(k)\}\}$$

Definition First, properties are grouped based on their corresponding values in favorite PVs of the cluster k:

$$G_f(k) = \{p \mid f \in \varphi,\ p \in P,\ F_p(k) = f\}$$

then, the preference value $v_k(i)$ for item i is computed as:

$$E_{k,f}(i) = f \times fmax\{\tilde{p}_i \mid p \in G_f(k)\}$$
$$v_k(i) = fmax\{E_{k,f}(i) \mid \forall f \in \varphi\}$$

Όπου ||φ|| ο αριθμός των διαφορετικών συνόλων των καλύτερων προτιμήσεων της ομάδας k.

Και P={p|p είναι είναι μια ιδιότητα της ομάδας χρηστών}

Όλη αυτή η διαδικασία μπορεί να γίνει την ώρα που οι χρήστες είναι offline και να έχουμε την λίστα με τα αντικείμενα τα οποία ενδιαφέρουν περισσότερο την ομάδα. Έπειτα θα βρούμε για τον κάθε χρήστη μεμονωμένα ποια είναι τα αντικείμενα που τον ενδιαφέρουν. Αυτό θα γίνει με το να συναθροίσουμε τις λίστες με τα αντικείμενα για κάθε ομάδα που ανήκει ένας χρήστης και να συνθέσουμε μια λίστα για έναν συγκεκριμένο χρήστη. Για να γίνει αυτό θα πρέπει πρώτα να

έχουμε υπολογίσει την ομοιότητα του χρήστη με την κάθε ομάδα S_{uk} όπως έχουμε περιγράψει παραπάνω.

Αρχικά οι ομάδες ομαδοποιούνται βασισμένες στην ομοιότητα με τον χρήστη u.

$$G_f(u)=\{k|f\in\varphi,S_{uk}=f\}$$

Έπειτα η προτίμηση $u_i(u)$ για το αντικείμενο i υπολογίζεται ως

$$E_{u,f}(i) = f \times fmax\{v_i(k) \mid k \in G_f(u)\}$$
$$v_i(u) = fmax\{E_{u,f}(i) \mid \forall f \in \varphi\}$$

Λόγο του ότι τα αντικείμενα που επιλέχθηκαν μπορεί να είναι πολλά θα κάνουμε ένα φιλτράρισμα των N ποιο σημαντικών. Αυτή η διαδικασία μπορεί να γίνει αρκετά περίπλοκη. Θα την περιγράψουμε απλοποιημένα.

Αρχικά κατακερματίζουμε τα αντικείμενα που έχουν επιλεχθεί σε σύνολα αντικειμένων σύμφωνα με κάποιον αλγόριθμο όπως ο αλγόριθμος LSH (Locality-sensitive hashing). Έπειτα παίρνουμε τα N πιο κοντινά αντικείμενα στο Vk με το να επισκεφτούμε κάθε κερματισμένο σύνολο ένα προς ένα μέχρι να βρεθούν τα N αυτά αντικείμενα

όπου

$$V_k = \{(p, F_p(k)) \mid p\epsilon P\}$$

Βιβλιογραφία:
[Yoda: An Accurate and Scalable Web-based Recommendation Syste από τους Cyrus Shahabi, Farnoush Banaei-Kashani, Yi-Shin Chen και Dennis McLeod]

4.12 Ένα Recommendation System που κάνει χρήση Συσχετιστικών Κανόνων

Θα παρουσιάσουμε ένα Recommendation system που αναλόγως με τις ενέργειες ενός χρήστη εφαρμόζει κάποιους κανόνες οι οποίοι παράγουν συστάσεις. Η γενική λογική είναι ότι αν ο χρήστης κάνει την ενέργεια Α τότε κάνε του την σύσταση Β.

Η ενέργεια Α στην απλή της περίπτωση μπορεί να είναι ότι απλώς είδε μια ιστοσελίδα. Σε μια πιο σύνθετη εκδοχή αν είδε έναν συνδυασμό από ιστοσελίδες. Σε μια ακόμη ποιο σύνθετη περίπτωση μπορεί να μας ενδιαφέρει ποιες σελίδες είδε με πια σειρά και πόση ώρα κάθισε σε κάθε σελίδα.

Οι κανόνες που εφαρμόζονται μπορούν να είναι διάφορων μορφών. Μια μορφή κανόνων είναι ότι αν ο χρήστης επισκέφτηκε ή αγόρασε ένα προϊόν να του συσταθεί έπειτα να δει και την συνέχεια του προϊόντος πχ ταινίες ή βιβλία που έχουν δεύτερο και τρίτο μέρος. Μια δεύτερη ομάδα κανόνων μπορεί να είναι ότι όταν επισκέπτεται ή αγοράζει ένα προϊόν να του γίνονται συστάσεις για όλα τα συνοδευτικά προϊόντα. Παράδειγμα αν ένας χρήστης αγοράσει ένα laptop θα ήταν μια καλή επιλογή να του συσταθεί μια τσάντα για laptop.

Κανόνες μπορούν να παραχθούν όχι μόνο συναρτήσει του τι έχει δει και αγοράσει ο χρήστης αλλά και με βάση άλλων πληροφοριών. Όπως η εποχή του χρόνου, η μέρα και η ώρα της εβδομάδας. Αν το σύστημα εντοπίσει έναν χρήστη να περιηγείται τις ιστοσελίδες του το Σάββατο απόγευμα ανεξαρτήτως με το τι βλέπει εκείνη την ώρα ο χρήστης είναι πολύ πιθανόν να τον ενδιαφέρει μια σύσταση για μια έξοδο το βράδυ.

Κανόνες σαν τους προηγούμενους θα πρέπει να τους ορίσει ρητά ο προγραμματιστής του συστήματος. Επίσης ρητά θα πρέπει να έχουν καταγραφεί οι σχέσεις μεταξύ των αντικειμένων. Ότι το αντικείμενο y είναι η συνέχεια του αντικειμένου x ή ότι κάποια αντικείμενα ταιριάζουν να συσταθούν το καλοκαίρι όπως ένα κλιματιστικό.

Από την άλλη μπορεί το ίδιο το σύστημα να παρακολουθεί την δραστηριότητα των χρηστών και να παράγει κανόνες τους οποίους θα είναι σε θέση να τους εφαρμόσει στους επόμενους χρήστες που θα το επισκεφτούν. Το σύστημα με αυτόν τον τρόπο θα είναι κάπως αυτοεκπαιδευόμενο και δυναμικό. Μια τέτοια ποιο σύνθετη υλοποίηση θα εξετάσουμε αναλυτικότερα.

Η αρχιτεκτονική του συστήματος αναλύεται και υλοποιείται σε τρία στάδια. Αρχικά θα πρέπει να γίνει η συλλογή των δεδομένων έπειτα θα πρέπει να αναγνωριστούν τα πρότυπα συμπεριφοράς των χρηστών και έπειτα να γίνουν οι συστάσεις. Τα πρώτα δύο στάδια λόγο του πλήθους των log files και του ότι χρειάζεται αρκετή επεξεργασία μπορούν να γίνουν offline.

Οι κανόνες θα παραχθούν βασιζόμενοι στις αρχές της ομοιότητας των δράσεων που έχουν κάνει οι χρήστες. Όλες οι ενέργειες των χρηστών κατά την διάρκεια της περιήγησης στο σύστημα έχουν αποθηκευθεί σε log files τα οποία θα επεξεργαστούμε για να παράγουμε τους κανόνες. Οι κανόνες που θα παραχθούν μπορούν να βασιστούν σε διάφορες τεχνικές όπως της κατηγοριοποίησης, της ομαδοποίησης και του να ανακαλυφθούν πρότυπα συμπεριφοράς χρηστών τα οποία θα συσχετίζουν αντικείμενα αναμεταξύ τους.

Υπάρχουν διάφοροι αλγόριθμοι που είναι κομμάτια της υλοποίησης ενός τέτοιου

συστήματος. Ενδεικτικά αναφέρουμε τους PageGather, WebPersonalizer, SUGGEST και Smart-Miner. Ο PageGather δημιουργεί ομάδες και έπειτα χρησιμοποιεί την ομαδοποίηση που δημιούργησε για να βρει συλλογές από παρόμοιες σελίδες. Ο WebPersonalizer έχει σκοπό να κάνει συστάσεις στους χρήστες σύμφωνα με παρόμοιες συμπεριφορές που έχουν οι χρήστες. Επεξεργάζεται log files και παράγει τα πιο σημαντικά πρότυπα περιήγησης μέσα σε ένα site. Ο SUGGEST και ο Smart-Miner μπορούν αποδοτικά να επεξεργαστούν μεγάλο μέγεθος από log files και να παράγουν χρήσιμα συμπεράσματα.

Ένα πολύ σημαντικό θέμα για το οποίο θα πρέπει να προνοήσουμε είναι η ανάλυση και το φιλτράρισμα των log files. Τα προβλήματα που καλούμαστε να αντιμετωπίσουμε σχετικά με τα log files είναι:

- Πώς θα φορμαριστούν το πλήθος των εγγραφών ούτως ώστε να εφαρμοστούν οι τεχνικές εξόρυξης δεδομένων.

- Η διαδικασία ταυτοποίησης μεταξύ των χρηστών και του πλήθους των συνεδριών που είχε ο καθένας.

- Η έλλειψη πληροφοριών γύρω από το περιεχόμενο κάποιον σελίδων.

- Η επεξεργασία των log files λόγο του πλήθος τους, χρειάζεται πολλούς επεξεργαστικούς πόρους και χρόνο.

Μια σίγουρη επιλογή που πρέπει να γίνει είναι κάθε χρήστης να τακτοποιηθεί με ένα μοναδικό ID ούτως ώστε από όποιον υπολογιστή και αν συνδέεται να αναγνωρίζονται όλες οι συνεδρίες που κάνει ως δικές του. Αυτό το πρόβλημα μπορεί να λυθεί σε ένα πρώτο επίπεδο με το να αναγνωρίζεται ο χρήστης μέσω της IP διεύθυνσης του. Αλλά γιατί όπως αναφέραμε μπορεί ο χρήστης να συνδέεται και από άλλους υπολογιστές ή γιατί η σύνδεση που χρησιμοποιεί του προσφέρει δυναμική IP είτε και ακόμη γιατί χρησιμοποιεί κάποιον Proxy server υπάρχουν ποιο περίπλοκες μέθοδοι. Ένας τρόπος είναι να ζητείται κάποιο συνθηματικό για να συνδεθεί στο σύστημα.

Για κάθε χρήστη θα αποθηκεύουμε σε έναν πίνακα το μοναδικό του ID καθώς επίσης και όλες τις πληροφορίες που κρίνουμε ότι θα πρέπει να τον συνοδεύουν. Από αυτές τις πληροφορίες θα είμαστε σε θέση να δημιουργήσουμε τους συσχετιστικούς κανόνες. Πληροφορία που είναι βασική και θα πρέπει να χρησιμοποιήσουμε είναι, ποιους συνδέσμους επέλεξε να δει ο χρήστης. Ιδικά αν κάποιους συνδέσμους επέλεξε να τους δει περισσότερες φορές από άλλους ή για περισσότερη ώρα μπορεί να αυξήσει την σημασία αυτού του συνδέσμου για τον συγκεκριμένο χρήστη.

Έχοντας έναν πίνακα όπου για κάθε χρήστη έχουμε την ακολουθία των αντικειμένων που είδε είμαστε σε θέση να παράγουμε συσχετιστικούς κανόνες. Αν συναντάμε συχνά το μοτίβο κάποιοι χρήστες να βλέπουν πρώτα έναν αριθμό αντικειμένων k και έπειτα έναν δεύτερο αριθμό αντικειμένων l. Ο κανόνας που μπορεί να παραχθεί θα είναι για όλους τους επόμενους χρήστες αν επιλέξουν να δουν κάποια από τα k αντικείμενα τότε να τους συστήσουμε τα l αντικείμενα.

Μπορούν να φτιαχτούν αρκετοί κανόνες για διάφορους συνδυασμούς των k αντικειμένων Το σύστημα μπορεί να μην περιμένει ακριβώς αυτά τα k αντικείμενα για να εφαρμόσει τον κανόνα. Μπορούμε να το παραμετροποιήσουμε, να εφαρμόζει τον κανόνα αν υπάρχει ένας βαθμός ομοιότητας άνω του 75% των k αντικειμένων με αυτά που είδε ο χρήστης. Ή κάθε φορά να εφαρμόζει τον κανόνα που ταιριάζει πιο πολύ στα αντικείμενα που έχει δει προς το παρόν ο χρήστης.

Ακόμη αν ο χρήστης δεν επιλέξει κάποιο από τα l αντικείμενα που του έχουν συσταθεί το σύστημα μπορεί να έχει την δυνατότητα να εφαρμόσει τον επόμενο κανόνα που ταιριάζει περισσότερο στον χρήστη.

Θετικό αυτής της μεθόδου είναι ότι η παραγωγή κανόνων μπορεί να γίνεται offline και οι κανόνες να αλλάζουν δυναμικά κάθε φορά ανάλογα με τις τελευταίες επιλογές που έκαναν οι χρήστες.

Βιβλιογραφία:

[Association-Rules-Based Recommender System for Personalization in Adaptive Web-Based Applications απο τους Daniel Mican και Nicolae Tomai]

5 Recommendation Systems προσαρμοσμένα στις ανάγκες του Twitter

Το Twitter είναι ένα κοινωνικό δίκτυο που προσφέρει microblogging υπηρεσίες. Κάθε χρήστης μπορεί να αναρτά μηνύματα (tweets) έως 140 χαρακτήρων τα οποία μπορούν να τα διαβάζουν οι φίλοι του. Ποιο συγκεκριμένα όταν αναρτά ένα μήνυμα όσοι χρήστες έχουν επιλέξει να ενημερώνονται για το τι γράφει (followers) θα το διαβάσουν. Ο ίδιος όμως ο χρήστης θα μπορέσει να διαβάσει μόνο τα μηνύματα από τους χρήστες που ο ίδιος έχει επιλέξει ή αλλιώς στην γλώσσα του Twitter "ακολουθεί" (following). Παράδειγμα ο πρόεδρος Barack Obama μπορεί να έχει εκατομμύρια χρήστες για followers αφού φυσικό είναι μεγάλο πλήθος του αμερικάνικου λαού να θέλει να ενημερώνεται για το τι λέει ο πρόεδρος τους. Αλλά από την άλλη ο ίδιος να έχει following μόνο λίγους χρήστες όπως τα κυβερνητικά του στελέχη και μέλη της οικογένειας του.

Ένα recommendation system στο Twitter μπορεί να υλοποιηθεί για να συστήνει tweets που έγραψαν άλλοι χρήστες, συνδέσμους (URLS) που εσωκλείονται μέσα σε ένα tweet, συγκεκριμένες πληροφορίες που έχουν αναγνωριστεί μέσα από λέξεις κλειδιά ή ακόμη και άτομα για να ακολουθεί ένας χρήστης. Για να γίνουν οι συστάσεις σε έναν χρήστη δεν θα χρησιμοποιηθούν μόνο οι γείτονες του χρήστη άλλα μπορεί να χρησιμοποιηθεί και ολόκληρο το κοινωνικό δίκτυο του Twitter. Οι μέθοδοι που θα ακολουθήσουμε είτε θέλουμε να συστήσουμε ολόκληρα tweets, είτε συνδέσμους, είτε συγκεκριμένες πληροφορίες διέπονται από την ίδια φιλοσοφία.

5.1 Σημεία που πρέπει να λάβουμε υπόψιν μας στο Twitter για να ενσωματώσουμε ένα Recommendation System

Η μορφή και ο τρόπος που λειτουργεί το Twitter μας καλεί να προσαρμόσουμε τις γνώσεις μας από τα recommendation systems ούτως ώστε να ταιριάζουν στις ιδιαιτερότητες του. Κάθε φορά αναλόγως με το τι είδους αντικείμενα έχουμε να συστήσουμε καθώς και σε τι σύστημα καλούμαστε να ενσωματώσουμε το recommendation σύστημα μας, πρέπει να επιλέξουμε τις αντίστοιχες μεθόδους και να τις προσαρμόσουμε ανάλογα.

- Τα κείμενα (tweets) που συγγράφουν οι χρήστες έχουν μέγιστο αριθμό 140 χαρακτήρες.

- Αν ένας χρήστης x διαβάσει τα tweets από έναν χρήστη y αυτό δεν σημαίνει πως και ο y διαβάζει τα tweets του χρήστη x.

- Ένας χρήστης του Twitter συνήθως δεν είναι ένας παθητικός αποδέκτης μηνυμάτων. Μπορεί να παράγει tweets με την ίδια άνεση που μπορεί και να διαβάσει. Από αυτά που διαβάζει μπορούν να βγουν συμπεράσματα για αυτά που παράγει και από αυτά που παράγει για αυτά που διαβάζει.

- Οι χρήστες αλληλεπιδρούν ρητά και με σαφή τρόπο αναμεταξύ τους. Ο καθένας μπορεί να καθορίσει ξεκάθαρα από ποιους θα διαβάζει tweets και μπορεί να αποτρέψει κάποιους από το να τον διαβάζουν.

- Το Twitter συνδυάζει πληροφορίες που βρίσκονται σε μορφή κειμένου (από τα tweets) και πληροφορίες που απορρέουν από την κοινωνική σχέση των χρηστών. Ένα Recommendation system μπορεί να βασιστεί σε ένα από τα δύο ή να τα συνδυάσει.

- Ο χρόνος που διαμεσολάβησε από την στιγμή που ένας χρήστης έγραψε ένα tweet έως την στιγμή που ένας δεύτερος το διάβασε είναι κρίσιμος και θα πρέπει το σύστημα να μην επιφέρει πρόσθετες καθυστερήσεις

- Τα tweets πολλές φορές εμπεριέχουν συνδέσμους (urls) που προσφέρουν την πραγματική πληροφορία που θέλει να μεταδώσει ο χρήστης και το συνοδευτικό κείμενο είναι απλός για να επεξηγήσει το θέμα του url.

- Το ίδιο tweet μπορεί να έχει αναρτηθεί από πολλούς χρήστες. Κατ επέκταση και ο ίδιος σύνδεσμος.

- Το Twitter αρκετά συχνά χρησιμοποιείται από κινητά τηλέφωνα με συνέπεια να αυξάνεται η αμεσότητα και η συχνότητα χρήσης του.

- Το Twitter διαθέτει ένα σύνολο από δημόσια API που μας επιτρέπουν να αναπτύξουμε τις δικές μας εφαρμογές. Ένα recommendation system μπορεί να υλοποιηθεί ως μια τέτοια εφαρμογή.

Το recommendation σύστημα που καλούμαστε να αναπτύξουμε στο κοινωνικό δίκτυο του Twitter καλείται να κάνει δύο βασικές εργασίες. Να φιλτράρει και να ανακαλύπτει της χρήσιμες πληροφορίες. Ένας χρήστης μπορεί να έχει μπροστά του εκατοντάδες tweets ακόμη και από αυτούς που ακολουθεί. Το σύστημα μας θα είναι πετυχημένο αν είναι σε θέση να τα φιλτράρει και να παρουσιάζει αυτά που τον ενδιαφέρουν.

Από την άλλη πλευρά είναι επιθυμητό το recommendation σύστημα να είναι σε θέση να ανακαλύπτει πληροφορίες από tweets άλλων χρηστών, τους οποίους μπορεί να μην ακολουθεί ο ίδιος αλλά να σχετίζονται έμμεσα μαζί του. Είτε γιατί είναι κοντά στον κοινωνικό γράφο είτε γιατί είχαν παρόμοια tweets στο παρελθόν.

Η χρήση της αρχιτεκτονικής Collaborative filtering δεν ενδείκνυται στην περίπτωση του Twitter. Ο χρόνος που θα χρειαστεί για να γίνει η ένταξη ενός tweet στα rating patterns των χρηστών και να γίνει η σύσταση θα είναι μεγάλος. Είναι καλύτερο να προσανατολιστούμε σε content based προσεγγίσεις. Παρακάτω θα παρουσιάσουμε μια σειρά από μεθόδους και τεχνικές που μπορούν να υλοποιήσουν ένα recommendation system που να ταιριάζει στις ανάγκες του twitter.

5.2 Επιλογή και Βαθμονόμηση του Υποψήφιου Συνόλου

Σε πρώτο επίπεδο συνειδητοποιούμε ότι το αρχικό πρόβλημα είναι πολύ μεγάλο και θα πρέπει με κάποιο τρόπο να μειωθεί. Κάθε ώρα μεταδίδονται 400.000 με 1.400.000 tweets. Αυτοί οι αριθμοί είναι πολύ μεγάλοι. Οπότε καθίσταται αδύνατο να γίνει αναλυτική επεξεργασία για όλα τα tweets σε πραγματικό χρόνο. Θα πρέπει να εφαρμοστεί μια μέθοδος φιλτραρίσματος όπου θα πρέπει να μειώσει το αρχικό πλήθος των tweets που θα εξετάσουμε.

Το φιλτράρισμα σε συνδυασμό με μια βαθμονόμηση των tweets μπορεί να είναι τόσο διεξοδικό που στο τέλος να μας έχουν μείνει ελάχιστα tweets τα οποία και θα τα συστήσουμε. Είναι δική μας επιλογή αν θέλουμε να κρατήσουμε ένα υποσύνολο από tweets τα οποία θα είναι υποψήφια και έπειτα εφαρμόζοντας μια από τις επόμενες μεθόδους να εξάγουμε τα τελικά που θα συσταθούν ή θα φιλτράρουμε και θα βαθμονομήσουμε διεξοδικά με αποτέλεσμα να καταλήξουμε σε αυτά τα tweets που έχουμε να συστήσουμε χωρίς άλλη επεξεργασία

Ένα τρόπος για να πραγματοποιηθεί το φιλτράρισμα είναι να χρησιμοποιήσουμε την αρχή της τοπικότητας. Μέσα στον κοινωνικό γράφο οι γείτονες που είναι πιο κοντά στον χρήστη που

εξετάζουμε, αναμένουμε να έχουν πολύ ποιο αυξημένες πιθανότητες να γράφουν tweets που θα τον ενδιαφέρουν. Όταν λέμε γείτονες εννοούμε :

- Τους χρήστες που ακολουθεί.
- Τους χρήστες από τους οποίους ακολουθείται.
- Τους χρήστες που ακολουθούν αυτούς που ακολουθεί.
- Τους χρήστες από τους οποίους ακολουθούνται αυτοί που ακολουθεί.
- Τους χρήστες που ακολουθούν αυτούς που τον ακολουθούν.
- Τους χρήστες από τους οποίους ακολουθούνται αυτοί που των ακολουθούν.

Αυτή την στιγμή χρησιμοποιούμε τον κοινωνικό γράφο και αναζητούμε τα tweets από χρήστες με βάθος δύο. Θα μπορούσαμε να συλλέξουμε ακόμη περισσότερα tweets αν αναζητούσαμε χρήστες σε μεγαλύτερο βάθος. Συνήθως όμως ένα βάθος δύο είναι αρκετό.

Όλες οι περιπτώσεις που αναφέραμε δεν έχουν την ίδια αξία. Ένα tweet από ένα χρήστη τον οποίο ακολουθεί ο χρήστης που εξετάζουμε αναμένουμε να έχει πιο μεγάλη σημασία από ένα tweet ενός χρήστη που ακολουθείται από κάποιον χρήστη που ακολουθεί. Οπότε θα μπορούσαμε αναλόγως την σχέση μεταξύ γείτονα και χρήστη να έχουμε έναν συντελεστή βαρύτητας στα tweets. Ιδικά αν κάποιοι χρήστες αλληλοακολουθούνται αναμεταξύ τους ή αν κάποιος συναντιέται περισσότερες από δύο φορές ως γείτονας θα περιμένουμε έναν ακόμη μεγαλύτερο συντελεστή βαρύτητας ο οποίος δημιουργήθηκε αθροιστικά.

Ένας δεύτερος τρόπος για να φιλτράρουμε τα tweets που κυκλοφορούν είναι πόσες φορές αναρτήθηκε αυτό το tweet είτε αυτούσιο, είτε τον σύνδεσμο (URL) που περικλείει, είτε ένα κομμάτι του πχ μια διεύθυνση ή ένα όνομα ξενοδοχείου. Αναμένουμε ότι όσο πιο σημαντική είναι μια πληροφορία τόσο πιο πολλές φορές θα έχει αναρτηθεί σε διάφορα tweets.

Ένας τρόπος για να επιτευχθεί αυτό είναι να γίνεται μια διαδικασία ταξινόμησης όλων των tweets. Θα ταξινομήσουμε όλα τα αλφαριθμητικά των 140 χαρακτήρων που έχουν γραφτεί αλφαβητικά. Κάθε φορά που βρίσκουμε ένα tweet να επαναλαμβάνεται αυτούσιο θα αυξάνουμε τον αριθμό που δηλώνει το πόσες φορές συναντήθηκε. Από εκεί και πέρα μπορούμε να επιλέγουμε ένα ποσοστό των tweets που συναντήθηκαν πιο πολλές φορές.

Αντίστοιχα μπορούμε να λειτουργήσουμε και για τους συνδέσμους ή για λέξεις κλειδιά. Θα ταξινομούμε αλφαριθμητικά κάθε καινούρια λέξη κλειδί που έρχεται και θα την προσθέτουμε στην ταξινομημένη μας λίστα με συχνότητα εμφάνισης ένα. Αν έρχεται ένας σύνδεσμος

αντίστοιχα μια λέξη κλειδί που είδη υπάρχει θα αυξάνουμε την συχνότητα εμφάνισης της.

5.3 Συστάσεις που βασίζονται στην Θεματολογία που έχουν τα Tweets.

Μια μέθοδος για να υλοποιηθεί ένα recommendation system στο Twitter είναι να αναγνωριστεί η θεματολογία στην οποία αναφέρεται κάθε tweet και έπειτα να συσταθούν tweets με παρόμοια θεματολογία. Κάθε καινούριο tweet που θα προκύπτει θα ελέγχουμε κατά πόσο ταιριάζει με τα θέματα που ενδιαφέρουν τον χρήστη και αναλόγως θα συστήνεται ή όχι.

Για κάθε χρήστη θα δημιουργήσουμε ένα προφίλ που θα περιέχει τις λέξεις που έχει γράψει με μεγαλύτερη συχνότητα στα tweets του. Μια λίστα που θα περιλαμβάνει την κάθε λέξη μαζί με τον αριθμό του πόσες φορές έχει γραφτεί. Αντίστοιχα η μέθοδος θα μπορούσε να περιλαμβάνει και τις λέξεις από τα tweets που έχει επιλέξει να διαβάσει. Προαιρετικά λέξεις που χρησιμοποιούν όλοι οι χρήστες όπως "είναι", "και", "αλλά" μπορούν να παραληφθούν. Αλλά ακόμη και αν δεν παραληφθούν από την αρχή η μέθοδος που θα περιγράψουμε θα τις αγνοήσει γιατί δεν είναι λέξεις που μπορούν να διακρίνουν τα ενδιαφέροντα του ενός χρήστη από του άλλου.

Θα περιγράψουμε πως ακριβώς λειτουργεί αυτή η μέθοδος. Αρχικά δεχόμαστε όλα τα tweets που έχει γράψει ο χρήστης και αποκόβουμε τις πιο σημαντικές λέξεις. Το πιες λέξεις θεωρούμε σημαντικές μπορούμε να το έχουμε καθορίσει από πιο πριν έχοντας μια μακροσκελή λίστα με τις σημαντικές λέξεις. Έπειτα για κάθε χρήστη δημιουργούμε στο προφίλ του ένα διάνυσμα V_u που είναι όλες οι λέξεις που έχει γράψει και τον χαρακτηρίζουν

$$V_u = (v_u(w_1), V_u(w_2), V_u(w_3), ..., V_u(W_m))$$

Όπου m είναι ο συνολικός αριθμός των ξεχωριστών λέξεων που έχουν γραφεί σε όλα τα tweets.

Και Το $v_u(w_i)$ περιγράφει το κατά πόσο ο χρήστης u χρησιμοποιεί και ενδιαφέρεται για την λέξη w_i.

Η τιμή του vu(wi) υπολογίζεται από τους ακόλουθους τύπους:

$TF_u (w_i)$ = Η συχνότητα που ο χρήστης u έγραψε την λέξη w_i στα tweets του.

$IDF_u (w_i)$ = log [(# Όλοι οι χρήστες) / (# Όλων τους χρήστες που χρησιμοποιούν την λέξη w_i τουλάχιστον μια φορά)]

$v_u(w_i) = TF_u (w_i) \cdot IDF_u (w_i)$, Έπειτα κανονικοποιούμε τις τιμές ούτως ώστε η κανονική τιμή του v_u να είναι το ένα.

Ένας μεγάλος αριθμός για το $TF_u(w_i)$ μιας λέξης σημαίνει ότι ο χρήστης u χρησιμοποιεί την λέξη w_i συχνά όποτε και ενδιαφέρεται για tweets που θα την περιλαμβάνουν.

Ένας μεγάλος αριθμός για το IDF_u (w_i) μιας λέξης σημαίνει ότι είναι λίγοι συνολικά οι χρήστες που την χρησιμοποιούν. Οπότε μπορεί να λειτουργήσει ως ένας παράγοντας για να διακρίνει τον έναν χρήστη από τους άλλους.

5.4 Συστάσεις που βασίζονται στα Tweets που Διάβασε ο Χρήστης

Η μέθοδος που περιγράψαμε δεν παίρνει υπόψιν της ότι μπορεί ένας χρήστης να γράφει tweets για ένα θέμα άλλα μπορεί να υπάρχουν και άλλα θέματα που των ενδιαφέρουν για τα οποία όμως να μην γράφει. Πολλοί χρήστες μπορεί να ενημερώνονται περιστασιακά ή συστηματικά για διάφορα θέματα που τους ενδιαφέρουν αλλά για διάφορους λόγους να μην έχουν γράψει πότε κάτι για αυτά. Το σύστημα μας θα πρέπει να είναι σε θέση να καταλαβαίνει τα ενδιαφέροντα ενός χρήστη όχι μόνο από το τι γράφει αλλά και από το τι διαβάζει.

Θα δημιουργήσουμε μια δεύτερη λίστα V_f στο προφίλ ενός χρήστη που θα βασίζεται στα tweets που γράφουν οι χρήστες που ακολουθεί. Για κάθε έναν από τους χρήστες f που ακολουθεί ο χρήστης που εξετάζουμε θα έχουμε ένα διάνυσμα από λέξεις που θα επιλεγούν με την ακόλουθη μέθοδο.

Θα επιλέξουμε όλες τις λέξεις που έχει γράψει ο χρήστης f και θα τις κατατάξουμε με φθίνουσα σειρά σύμφωνα με το πόσες φορές έχει χρησιμοποιήσει ο χρήστης f την κάθε λέξη. Στη συνέχεια επιλέγουμε από αυτές το 20% των ποιο συχνά χρησιμοποιημένων λέξεων. Τελευταίο στάδιο είναι να αφαιρέσουμε από αυτό το 20% των λέξεων που έχουν περισσέψει, όλες τις λέξεις που χρησιμοποιεί ο χρήστης f αλλά δεν χρησιμοποιούν οι υπόλοιποι χρήστες που ακολουθεί ο χρήστης που εξετάζουμε. Οι λέξεις που θα απομείνουν είναι οι λέξεις που θα αντικατοπτρίζουν περισσότερο τα θέματα για τα οποία ενδιαφέρεται να γράφει ο χρήστης f.

Η δεύτερη λίστα V_f θα δημιουργηθεί με παρόμοιο τρόπο όπως και η V_u με τις λίγες διαφορές. Οι λέξεις w_i που θα χρησιμοποιήσουμε είναι οι λέξεις που θα απομείνουν από την προηγούμενη διαδικασία έστω αυτό το σύνολο λέγεται l_f

FTF_f (w_i) = (# Ο αριθμός των χρηστών f που ακολουθεί ο χρήστης u που εξετάζουμε και έχουν την λέξη w_i στο σύνολο l_f)

IDF_f (w_i) = log [(# Όλοι οι χρήστες f) / (# Όλους τους χρήστες f που χρησιμοποιούν την λέξη w_i τουλάχιστον μια φορά)]

$V_f(w_i) = FTF_f$ $(w_i) \cdot IDF_f$ (w_i), Έπειτα κανονικοποιούμε τις τιμές ούτως ώστε η κανονική τιμή του v_u να είναι το ένα.

Βλέπουμε ότι αντικαταστήσαμε το αντίστοιχο TF_u (w_i) με το FTF_f (w_i). Το FTF_f (w_i) δηλώνει το πόσοι πολλοί από τους χρήστες που ακολουθεί ο u χρησιμοποιούν συχνά την λέξη w_i

5.5 Συστάσεις Συνδέσμων από Tweets

Η ανάλυση που κάναμε είναι για την σύσταση tweet που έχουν γράψει άλλοι σε έναν χρήστη. Αυτές οι μέθοδοι μπορούν να προσαρμοστούν και για να συστήνουμε συνδέσμους (URLS). Μια πρώτη σκέψη θα ήταν να ανιχνεύουμε τα tweets που περιέχουν συνδέσμους και σε αυτά να εφαρμόζουμε τις παραπάνω μεθόδους στο κείμενο που περιβάλει και επεξηγεί τον σύνδεσμο. Στο τέλος θα συστήνουμε μόνο τον σύνδεσμο ανεξαρτήτως των tweet που των περιέλαβαν.

Για την σύσταση συνδέσμων υπάρχουν πιο εξειδικευμένες μέθοδοι. Μια από αυτές προσπαθεί να μοντελοποιήσει το θέμα ενός συνδέσμου ως ένα διάνυσμα. Αντίστοιχα θα υπάρχει και το διάνυσμα ενδιαφερόντων του χρήστη. Έπειτα υπολογίζουμε το συνημίτονο μεταξύ των δύο διανυσμάτων και έχουμε μια ένδειξη για το κατά πόσο ταιριάζει αυτός ο σύνδεσμος με τα ενδιαφέροντα του χρήστη. Στο τέλος θα συστήσουμε τους σύνδεσμος με τον μεγαλύτερο βαθμό ομοιότητας.

Γενικώς η μέθοδος του να βρίσκουμε την ομοιότητα μεταξύ ενός αντικειμένου με το προφίλ ενός χρήστη είναι μια μέθοδος που την ακολουθήσαμε σε πολλά recommendation systems. Αυτό που έχει πιο πολύ σημασία είναι το πως δημιουργούμε το διάνυσμα που χαρακτηρίζει τον συγκεκριμένο σύνδεσμο.

Μια μέθοδος είναι ότι καταμετράμε κάθε λέξη που συνοδεύει τον σύνδεσμο μέσα στο tweet. Όσες πιο πολλές φορές βρέθηκε η ίδια λέξη να συνοδεύει τον σύνδεσμο τόσο πιο μεγάλο συντελεστή θα έχει. Στο τέλος ως διάνυσμα που χαρακτηρίζει τον σύνδεσμο έχουμε τις πιο συχνές λέξεις που βρέθηκαν να συνοδεύουν στον σύνδεσμο μαζί με μια τιμή που προσδιορίζει το πόσες φορές τις συναντήσαμε. Το καλό με αυτή την μέθοδο είναι ότι δεν χρειάζεται να ανατρέξουμε μέσα στα περιεχόμενα του κάθε συνδέσμου για να ανακαλύψουμε το περιεχόμενο του.

Μια άλλη μέθοδος για να συστήνουμε συνδέσμους από tweets βασίζεται στις εξής αρχές ότι όσο περισσότεροι χρήστες έχουν περιλάβει έναν σύνδεσμο στα tweets τους τόσο πιο πιθανό είναι ο σύνδεσμος να είναι άξιος να συσταθεί. Επίσης δεν αναφερόμαστε γενικώς σε όλους τους χρήστες του Twitter αλλά στους χρήστες που είναι πιο κοντά στον κοινωνικό γράφο με τον χρήστη που εξετάζουμε. Ακόμη θα πρέπει να λάβουμε υπόψιν μας την αξιοπιστία που έχει κάθε ένας από αυτούς τους χρήστες. Αλλά ας μιλήσουμε για όλα αυτά αναλυτικότερα

Το πρώτο μας μέλημα είναι να επιλέξουμε το σύνολο των χρηστών που θεωρούμε ότι θα αναρτούν συνδέσμους που θα ενδιαφέρουν τον χρήστη που εξετάζουμε. Η επιλογή αυτών μπορεί

να γίνει αν ακολουθήσουμε σε κάποιο βάθος όλους τους χρήστες που ακολουθεί, των ακολουθούν και αντίστοιχα αυτούς που ακολουθούν και ακολουθούνται από αυτούς που ακολουθεί και τον ακολουθούν ως ένα βάθος που θα επιλέξουμε εμείς.

Έπειτα θα προσπαθήσουμε με κάποια κριτήρια να βρούμε πόσο αξιόπιστος και πόσο σημαντικοί είναι οι σύνδεσμοι που αναρτά ο κάθε χρήστης. Ένα κριτήριο είναι ότι η εμπιστοσύνη που έχει ο χρήστης Α σε έναν χρήστη Β θα αυξηθεί αν υπάρχουν πολλοί χρήστες που ακολουθούν τον Β και ταυτόχρονα ο Α τους ακολουθεί.

Ένα άλλο κριτήριο είναι η συχνότητα με την οποία ένας χρήστης αναρτά tweets. Είναι πολύ πιθανόν χρήστες που χρησιμοποιούν το Twitter κυρίως για φλυαρία και οτιδήποτε βρίσκουν μπροστά τους να το αναρτούν να μην προσφέρουν πληροφορίες και συνδέσμους που έχουν μεγάλη αξία. Αυτή η κατηγορία χρηστών συνήθως αναρτούν πληροφορίες γιατί αρέσκονται στο να χρησιμοποιούν το Twitter και όχι γιατί έχουν κάτι σημαντικό να δημοσιοποιήσουν. Από την άλλη χρήστες που αναρτούν πιο σπάνια συνδέσμους στα tweets είναι πιο πιθανόν να πρέπει κάτι να τους έχει κάνει μεγάλη εντύπωση και να είναι πολύ σημαντικό για να το αναφέρουν. Αυτό μπορούμε να το περιλάβουμε με το να προσθέσουμε έναν συντελεστή αξιοπιστίας για τον κάθε χρήστη που θα είναι αντιστρόφως ανάλογος του πλήθους των tweets που δημοσιεύει.

Από τους χρήστες που έχουμε επιλέξει να εξετάζουμε τα tweets τους, αναζητάμε τους συνδέσμους που έχουν αναρτήσει και εφαρμόζομαι τον εξής κανόνα. Όσο πιο σημαντικός είναι ένας σύνδεσμος τόσο πιο πολύ αναμένουμε να τον έχουν αναφέρει στα tweets τους. Οπότε κάθε σύνδεσμος που αναφέρεται από το σύνολο των χρηστών που έχουμε επιλέξει τον ταξινομούμε σε μια λίστα. Η ταξινόμηση μπορεί να γίνει αλφαριθμητικά μαζί με έναν αριθμό που υποδηλώνει το πόσες φορές αναφέρθηκε και ένα συνάθροισμα της αξιοπιστίας όλων των χρηστών που ανέφεραν τον σύνδεσμο. Από αυτούς τους παράγοντες τελικά υπολογίζουμε τους συνδέσμους που θα συστήσουμε.

Ο Κανόνας του ότι όσο πιο σημαντικός είναι ένας σύνδεσμος τόσο πιο πολλοί αναμένουμε να τον έχουν αναφέρει στα tweets τους δεν ισχύει πάντα, αλλά αναμένουμε με ένα αρκετά καλό ποσοστό επιτυχίας να μας προσφέρει τα πιο δημοφιλή tweets που ενδιαφέρουν έναν χρήστη.

Βιβλιογραφία:

[Short and Tweet: Experiments on Recommending Content from Information Streams απο τούς Jilin Chen , Rowan Nairn , Les Nelson , Michael Bernstein , Ed H. Chi]

6 Recommendation System στο Twitter που κάνει χρήση Multi Document Summarization μεθόδων

Ένας πολύ ενδιαφέρον αλλά σύνθετος τρόπος για να γίνονται συστάσεις είναι να χρησιμοποιήσουμε την μέθοδο του summarization text. Θα είχε πολύ ενδιαφέρον ένα recommendation system όπου θα μπορεί να διαβάζει όλα τα tweets ενός χρήστη και έπειτα να μπορεί να έχει μια περίληψη όλων αυτών που έχει γράψει.

Έχοντας μια περίληψη από τα tweets που έχουν γράψει διάφοροι χρήστες και συγκρίνοντας τες αναμεταξύ τους θα είμαστε σε θέση να βρίσκουμε χρήστες που γράφουν και ενδιαφέρονται για ίδια θέματα. Επίσης κάθε tweet που προκύπτει μπορούμε να το συγκρίνουμε με την περίληψη που αντιστοιχεί σε έναν χρήστη και να ξέρουμε κατά πόσο διαπραγματεύεται ένα θέμα που απασχολεί τον χρήστη. Βέβαια υπάρχει μια πιθανότητα αν η ομοιότητα του tweet με την περίληψη που αντιστοιχεί στον χρήστη είναι από κάποιο βαθμό και πάνω μεγάλη ίσως το tweet να μην έχει να πει κάτι καινούριο στον χρήστη.

Σκοπός του multi-document summarization είναι αλγοριθμικά να εξάγει από πολλές πηγές (στην συγκεκριμένη περίπτωση tweets) μια αμερόληπτη περίληψη χωρίς καμία επέμβαση από κάποιον εξωτερικό χρήστη. Η περίληψη πρέπει να είναι συνοπτική και περιεκτική.

Το μεγάλο πρόβλημα στην μέθοδο αυτή είναι ότι τα tweets μεταξύ τους μπορεί να έχουν μια διακύμανση στα θέματα που διαπραγματεύονται. Σκοπός μας είναι να μπορέσουμε να τα συνδυάσουμε και να έχουμε ως αποτέλεσμα ένα κείμενο που να είναι πλήρες, αναγνώσιμο και περιεκτικό. Τα κριτήρια που πρέπει να ικανοποιεί η τελική μας περίληψη είναι.

- Να υπάρχει καθαρή δομή στο κείμενο.
- Το κείμενο να είναι διαιρεμένο σε παραγράφους που η κάθε μια να αναπτύσσει ένα θέμα.
- Να μην υπάρχει πλεονασμός στις πληροφορίες που δύνονται.

- Να υπάρχει μια σταδιακή μετάβαση από τα γενικά θέματα σε ποιο συγκεκριμένα
- Να υπάρχει καλή αναγνωσιμότητα είτε από ένα άνθρωπο είτε από ένα πρόγραμμα.

Για να είναι μια περίληψη σωστή πρέπει να παράγει ένα κείμενο που να είναι ευχάριστο να διαβαστεί από τον αναγνώστη. Εδώ πέρα όμως ο σκοπός μας δεν είναι η παραγωγή της περίληψης. Η περίληψη είναι το μέσο για να γίνουν οι συστάσεις. Οπότε ακόμη και αν η περίληψη μας δεν έχει μια ευχάριστη, διηγηματική μορφή δεν θα είναι μεγάλο πρόβλημα. Το θέμα μας είναι να δοθεί ως είσοδος στον συγκριτή κειμένων για να βρει με βάση την ομοιότητα τα tweets που θα συστήσει. Παρακάτω θα περιγράψουμε τις βασικές μεθόδους που μπορούμε να κάνουμε μια περίληψη.

6.1 Περιλήψεις που παράγονται με βάση την Αποκοπή Κομματιών από τα Αρχικά Κείμενα

Η πιο απλή μορφή περιλήψεων που μπορούμε να κάνουμε είναι με το να αποσπάσουμε τις πιο σημαντικές προτάσεις από τα αρχικά κείμενα (tweets) και να τις συρράψουμε στο κείμενο της περίληψης μας. Αν όλα μας τα tweets διαπραγματεύονται το ίδιο θέμα η διαδικασία της συνένωσης είναι εύκολη. Αν όμως τα tweets διαπραγματεύονται διαφορετικά θέματα θα πρέπει να δώσουμε περισσότερη σημασία ούτως ώστε κάθε παράγραφος να ανταποκρίνεται σε μια θεματική ενότητα και όλες οι προτάσεις που διαπραγματεύονται το ίδιο θέμα να είναι μαζί.

Συνήθως τα κομμάτια που αποσπούνται είναι προτάσεις ή ακόμη και παράγραφοι. Τώρα λόγο του περιορισμένου μεγέθους των κειμένων του Twitter μπορεί να αποσπαστούν και μικρότερα κομμάτια όπως λέξεις. Υπάρχουν διάφορα κριτήρια σύμφωνα με τα οποία μπορούμε να επιλέξουμε ποια κομμάτια να αποσπάσουμε αυτά είναι:

- Η συχνότητα με την οποία μια λέξη ή μια φράση μπορεί να επαναλαμβάνεται.
- Η δομή του κειμένου και η θέση που έχει μια λέξη μέσα στο αρχικό κείμενο.
- Αν υπάρχουν κρίσιμες λέξεις τις οποίες τις έχουμε προκαθορίσει από πριν ότι είναι

σημαντικές.

- Οι σχέσεις μεταξύ των λέξεων που υπάρχουν.
- Χρήση συντακτικών ή στατιστικών κανόνων.
- Χρήση τεχνικών μηχανικής μάθησης (machine learning).

Ένας ή συνδυασμός των παραπάνω κριτηρίων μπορούν να εφαρμοστούν. Αφού επιλεγούν τα ποιο σημαντικά αποσπάσματα προσπαθούμε να τα ομαδοποιήσουμε ούτως ώστε αυτά που είναι πιο κοντά θεματολογικά να σχηματίσουν παραγράφους. Μπορεί να χρειαστεί να προστεθούν και βοηθητικές λέξεις για να ενώσουν τα αποσπάσματα αναμεταξύ τους.

6.2 Αφηρημένες Μέθοδοι για την Παραγωγή Περιλήψεων

Υπάρχουν περιπτώσεις που η δομή μιας περίληψης είναι συγκεκριμένη και απλώς περιμένουμε τις πληροφορίες που θα εξάγουμε από τα αρχικά κείμενα για να της βάλουμε. Θα μπορούσαμε να το παρομοιάσουμε με μια φόρμα που είναι έτοιμη να συμπληρωθεί και αναλόγως τις εκάστοτε συνθήκες την συμπληρώνουμε

Για να καταλάβουμε καλύτερα τη ευνοούμε έστω ότι έχουμε πολλά tweets γύρω από έναν μουσικό. Θα μπορούσαμε να έχουμε μια ασυμπλήρωτη φόρμα που μπορεί να περιγράφει οποιονδήποτε μουσικό με κάποια κενά. Από τα στοιχεία που θα εξάγουμε από τα tweets του θα αρχίσουμε να συμπληρώνουμε τα κενά: το μουσικό είδος που παίζει, τι μουσικούς δίσκους έχει κυκλοφορήσει, τι συναυλίες προετοιμάζεται να δώσει, με ποιους μουσικούς έχει συνεργαστεί και πόσο δημοφιλής είναι στο κοινό. Στο τέλος σίγουρα θα έχουμε μια πολύ καλή περίληψη του μουσικού.

Το σημαντικό κομμάτι της δουλειάς είναι τώρα το σύστημα μας να εντοπίσει τις πληροφορίες που χρειαζόμαστε από τα αρχικά tweets. Αυτό μπορεί να γίνει με το να θεωρήσουμε ότι τις πληροφορίες που χρειαζόμαστε τις περιβάλουν συγκεκριμένες λέξεις κλειδιά και να τις αναζητήσουμε.

Μια άλλη αφηρημένη μέθοδος για την παραγωγή περιλήψεων είναι η διαδικασία κατά την οποία μπορούμε να δεχθούμε δύο ή τρεις ή ακόμη και περισσότερες προτάσεις και να προσπαθήσουμε να τις συντάξουμε σε μία αφαιρώντας λέξεις που κρίνουμε ότι δεν έχουν μεγάλη σημασία.

Πολύ ενδιαφέρουσες αφηρημένες περιλήψεις θα μπορούσαμε να έχουμε αν

αναγνωρίζαμε ότι τα πιο σημαντικά αποσπάσματα που έχουμε πάρει από τα αρχικά tweets συνιστούν μια καινούρια γνώση η οποία δεν είναι απαραίτητα αντιγραφή και συμπύκνωση του αρχικού κειμένου. Η καινούρια γνώση μπορεί να είναι συνεπαγωγή και λογικό συμπέρασμα όλων αυτών που έχουν γραφεί. Αυτή η μέθοδος είναι ιδιαίτερα περίπλοκη και για να υλοποιηθεί το σύστημα θα πρέπει να έχει πρόσβαση σε εξωτερικές πληροφορίες όπως οντολογικές ή γνωσιολογικές βάσεις.

Για να αντιληφθούμε καλύτερα την μέθοδο αυτή ας υποθέσουμε το παράδειγμα να υπάρχουν μια μέρα πολλά αποσπάσματα από tweets που να έχουν εκφράσεις και λέξεις όπως, απόδραση στις παραλίες, όλη μέρα αναμμένα τα κλιματιστικά, λιποθυμίες, υδράργυρος είναι επόμενο να καταλάβουμε πως εκείνη την ημέρα είχε καύσωνα. Οπότε μπορούμε να αντικαταστήσουμε όλα τα αποσπάσματα που δίνουν σχετικές πληροφορίες με το ότι εκείνη την ημέρα είχε καύσωνα.

6.3 Σύνθεση της Τελικής Περίληψης από τα Επιμέρους Κομμάτια

Με όποια από τις παραπάνω μεθόδους και αν ξεκινήσουμε την περίληψη θα καταλήξουμε στο πρόβλημα του πώς θα συνθέσουμε τις προτάσεις και τις λέξεις που εμπεριέχουν όλο το νόημα από τα αρχικά tweets. Ένας τρόπος είναι να μετρήσουμε την ομοιότητα μεταξύ όλων των αποσπασμάτων και να τα ομαδοποιήσουμε αναμεταξύ τους.

Ένας άλλος τρόπος είναι να αρχίσουμε να συντάσσουμε το κείμενο της περίληψης από ένα τυχαίο. Έπειτα να ανατρέξουμε επαναληπτικά σε όλα τα αποσπάσματα που δεν έχουμε περιλάβει ακόμη και να προσθέσουμε όσα ταιριάζουν. Αυτό που προσθέσαμε δεν θα το ξανά προσθέσουμε. Αν μια φορά ανατρέξουμε όλα τα αποσπάσματα και δεν βρεθεί κάποιο να είναι όμοιο τότε ξεκινάμε καινούρια παράγραφο με ένα άλλο τυχαίο απόσπασμα.

Μια πιο απλή μέθοδος είναι να παραθέσουμε ότι έχει αποσπαστεί με την σειρά της χρονικής ακολουθίας που αναρτήθηκε στο Twitter. Αυτό βέβαια θα μπορούσε να εφαρμοστεί αν οι χρήστες έχουν μια λογική ακολουθία σε αυτά που γράφουν και δεν μεταβαίνουν άστατα από το ένα θέμα στο άλλο.

Γενικώς το πρόβλημα του να παραχθούν περιλήψεις από πολλά κείμενα είναι ένα σύνθετο πρόβλημα το οποίο είναι ανοιχτό. Παρακάτω θα παρουσιάσουμε μια ποιο σύνθετη αλλά και ολοκληρωμένη μέθοδο δημιουργίας περιλήψεων που βασίζεται στην χρήση γράφων.

Βιβλιογραφία:

[Introduction to the Special Issue on Summarization από τους Dragomir R. Radev και Kathleen McKeown]

6.4 Μέθοδοι για Σύγκριση και Εύρεση Ομοιότητας μεταξύ Περιλήψεων.

Αφού δημιουργηθεί μια περίληψη με μια από τις προηγούμενες μεθόδους θα πρέπει να είμαστε σε θέση να μπορούμε να συγκρίνουμε τις περιλήψεις αναμεταξύ τους για να έχουμε έναν βαθμό που να προσδιορίζει την σημασιολογική τους ομοιότητα. Δηλαδή κατά πόσο τα δύο κείμενα έχουν κάποιο κοινό νόημα και διαπραγματεύονται το ίδιο θέμα.

6.5 Δημιουργία και Σύγκριση Περιλήψεων μέσω Γράφων

Θα αναπτύξουμε μια μέθοδο για την αυτόματη επιλογή και αποκοπή προτάσεων από ένα κείμενο χρησιμοποιώντας αλγορίθμους που βασίζονται στην βαθμονόμηση γράφων με σκοπό την παραγωγή περιλήψεων.

Σε έναν γράφο οι κόμβοι θα αντιστοιχούν σε λέξεις ή προτάσεις. Κυρίως θα χρησιμοποιήσουμε προτάσεις. Οι ακμές θα είναι οι σχέσεις αναμεταξύ τους. Θα αποφασίζουμε την σημασία ενός κόμβου μέσα στον γράφο βασιζόμενε στην θέση που έχει και στην σχέση του με τους άλλους κόμβους. Αρχικά θα δημιουργήσουμε τον γράφο που αναπαριστά το κείμενο και θα διασυνδέσουμε τις λέξεις ή τα αποσπάσματα που έχουμε. Οι διασυνδέσεις μεταξύ των κόμβων θα γίνουν με τέτοιο τρόπο ούτως ώστε να υπάρχει κάποια νοηματική σχέση μεταξύ τους.

Ποιο συγκεκριμένα για την παραγωγή των συνδέσεων θα αναζητήσουμε μια σχέση ομοιότητας. Σχέση ομοιότητας θα έχουμε όταν δύο προτάσεις έχουν νοηματική επικάλυψη. Δηλαδή όταν δύο προτάσεις έχουν κάποιο κοινό νόημα. Όταν υπάρχει μια σχέση ομοιότητας μεταξύ δύο προτάσεων μπορούμε να το αναπαραστήσουμε αυτό στον γράφο με μια ακμή που να συνδέει τους δύο αντιστοίχους κόμβους.

Κάθε ακμή μπορεί να είναι ισοσταθμισμένη με έναν συντελεστή βαρύτητας που αντιστοιχεί στο κατά πόσο μεγάλη είναι η ομοιότητα μεταξύ των δύο αυτών ακμών. Θα περιγράψουμε μια μέθοδο που θα υπολογίζει τον βαθμό ομοιότητας μεταξύ δύο προτάσεων βασισμένη στο πλήθος των κοινών στοιχείων που έχουν οι λεκτικές τους αναπαραστάσεις. Εδώ μπορεί να χρησιμοποιηθεί ένα συντακτικό φίλτρο το οποίο μετράει λέξεις μιας συγκεκριμένης κατηγορίας.

Έστω ότι έχουμε δύο προτάσεις S_i και S_j και κάθε πρόταση αναπαρίσταται από N_i λέξεις που εμφανίζονται στην πρόταση ως $S_i = w_1^i, w_2^i, w_1^i, ..., w_{Ni}^i$. Η ομοιότητα μεταξύ των προτάσεων S_i και S_j ορίζεται ως:

$$Similarity(S_i, S_j) = \frac{|W_k|W_k \in S_i \& W_k \in S_j|}{log(|S_i|)+log(|S_j|)}$$

Με την εφαρμογή του τύπου ομοιότητας θα έχουμε έναν σταθμισμένο συνδεδεμένο γράφο με συντελεστές βαρύτητας σε κάθε ακμή που να υποδηλώνουν την σύνδεση μεταξύ των προτάσεων που υπάρχουν στο κείμενο.

Μετά την κατασκευή του γράφου όπως περιγράψαμε πιο πριν θα αναφέρουμε μια σειρά από μεθόδους που θα εντοπίζουν τις πιο σημαντικές προτάσεις ή αντίστοιχα τους πιο σημαντικούς κόμβους.

Μια μέθοδος για να εντοπίσουμε τους πιο σημαντικούς κόμβους είναι να καταμετρήσουμε τον αριθμό των κόμβων που απορρέουν ως διάδοχοι από έναν άλλον κόμβο καθώς και τον βαθμό τους. Δηλαδή από έναν κόμβο πόσοι άλλοι κόμβοι υπάρχουν που τον διαδέχονται. Ο τύπος που το υπολογίζει αυτό είναι:

$$POS_P(V_i) = \frac{1}{|V|} \sum_{V_j \in Out(V_i)} (1 + POS_P(V_j))$$

Ένας άλλος τρόπος υπολογισμού των σημαντικών κόμβων προκύπτει από τον τύπο

$$PR(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{PR(V_j)}{|Out(V_j)|}$$

Όπου το d είναι μια παράμετρος μεταξύ 0 και 1.
$In(V_i)$ είναι το σύνολο των προηγούμενων (προκάτοχων) κόμβων .
$out(V_i)$ είναι το σύνολο των επόμενων (διάδοχων) κόμβων.

Για τους δύο παραπάνω τύπους μπορούμε να ξεκινήσουμε από μια αυθαίρετη τιμή για κάθε κόμβο. Έπειτα από κάποιες επαναλήψεις θα συγκλίνουν στης τιμές που για κάθε κόμβο θα δείχνουν την πραγματική του αξία μέσα στον γράφο.

Αφού βαθμολογήσουμε όλους τους κόμβους τους διατάσσουμε σε μια αντίστροφη σειρά και κρατάμε τους ποιο υψηλόβαθμους Αυτοί οι πιο υψηλόβαθμοι θα είναι αυτοί που θα αποτελούν την περίληψη μας. Αν θέλουμε μπορούμε να τους ξανά διατάξουμε σύμφωνα με την

σειρά που βρίσκονταν στο αρχικό κείμενο.

Βιβλιογραφία:
[Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization απο την Rada Mihalcea]

7 Μια Ανάλυση στο Recommendation System που βασίζεται στο N-Gram Graphs

Μια πολύ ενδιαφέρουσα μέθοδος για να κατηγοριοποιούμε κείμενα και έπειτα να κάνουμε συστάσεις είναι μέσω των N-Gram γράφων. Με αυτή την μέθοδο θα ασχοληθούμε αναλυτικότατα, θα την περιγράψουμε, θα την υλοποιήσουμε προγραμματιστικά και θα την εφαρμόσουμε σε διάφορα tweets όπου και θα αξιολογήσουμε τα αποτελέσματα της. Η ιδέα της μεθόδου αυτής με μια πρώτη ματιά είναι να δημιουργεί σταθμισμένους γράφους από κείμενα. Κάθε κόμβος του γράφου είναι μια σειρά γραμμάτων από το αρχικό κείμενο. Οι ακμές του γράφου δημιουργούνται με τον εξής τρόπο. Κάθε φορά που η ακολουθία γραμμάτων που αντιστοιχεί σε έναν κόμβο βρίσκεται στο κείμενο κοντά σε μιαν άλλη ακολουθία γραμμάτων που αντιστοιχεί σε κάποιον άλλον κόμβο του γράφου αυξάνεται η αξία της ακμής που τους ενώνει. Συγκρίνοντας τους αντίστοιχους σταθμισμένους γράφους που έχουν παραχθεί από τα διάφορα κείμενα μπορούμε να συμπεράνουμε κατά πόσο δύο οι περισσότερα κείμενα έχουν κοινά στοιχεία αναμεταξύ τους.

Αυτή η μέθοδος έχει πολλά προτερήματα όπως του ότι είναι ανεξάρτητη από την γλώσσα που είναι γραμμένο το κείμενο, είναι αυτοματοποιημένη, βασίζεται σε στατιστικούς κανόνες καθώς και πολλά άλλα στοιχειά που θα τα δούμε στην συνέχεια της παρουσίασης της.

Ας εξετάσουμε τις αρχές αυτής της μεθόδου αναλυτικότερα. Σκοπός μας είναι από ένα κείμενο να εξάγουμε στατιστικές πληροφορίες, να τις αναπαριστούμε, να τις συγκρίνουμε με τις αντίστοιχες από άλλα κείμενα και να βρούμε μέσω ενός μοντελοποιημένου τρόπου τις ομοιότητες τους. Τα τρία θέματα που κατευθείαν προκύπτουν είναι:

- Ο τύπος των στατιστικών δεδομένων που θα εξαχθούν.
- Ο τρόπος που θα αναπαρασταθούν τα δεδομένα από τις πληροφορίες που έχουν εξαχθεί.
- Η μέθοδος που θα υπολογίζει την ομοιότητα τους.

Ο τύπος των στατιστικών δεδομένων που θα εξαχθούν από το αρχικό κείμενο είναι η σχέση ανάμεσα σε ακολουθίες γραμμάτων (όχι απαραίτητα ρίζες λέξεων) με άλλες αντίστοιχες ακολουθίες γραμμάτων. Αν για παράδειγμα έχουμε την πρόταση. Ο Νίκος έφαγε μήλο. Η

ακολουθία γραμμάτων Νίκος συσχετίζεται με την ακολουθία γραμμάτων έφαγε και αντίστοιχα η ακολουθία γραμμάτων έφαγε συσχετίζεται και με τις δύο ακολουθίες γραμμάτων έφαγε και μήλο. Στο παράδειγμα μας προς χάριν απλότητας επιλέξαμε οι ακολουθίες γραμμάτων μας να είναι όλες των πέντε γραμμάτων - σημείων στίξεως και να αντιστοιχούν σε μία λέξη η κάθε μια. Το σημαντικό είναι ότι αυτές οι ακολουθίες γραμμάτων που από εδώ και πέρα θα τις λέμε ν-γράμματα (μετάφραση του όρου n-grams) βρέθηκαν κοντά αναμεταξύ τους, όποτε αναμένουμε να έχουν κάποια νοηματική συνάφεια. Αυτή η νοηματική συνάφεια μας ενδιαφέρει. Μας ενδιαφέρει η γειτονικότητα των διάφορων ν-γραμμάτων που μπορεί να υπάρχει σ' ένα κείμενο.

Επόμενο μέλημα μας είναι πως θα μπορέσουμε να αναπαραστήσουμε τα στατιστικά δεδομένα που εξάγουμε από ένα κείμενο. Η καλύτερη μέθοδος είναι ένας σταθμισμένος γράφος όπου θα έχουμε ως κορυφές ν-γράμματα και ως ακμές τις σχέσεις μεταξύ όλων αυτών των ν-γραμμάτων. Οι ακμές μπορούν να αντιπροσωπεύουν την μέση απόσταση ανάμεσα στα γειτονικά ν-γράμματα. Μια άλλη εκδοχή θα ήταν να αντιπροσωπεύουν την στατιστική πιθανότητα που θα υποδηλώνει ότι έχοντας ένα ν-γραμμα πόσες φορές συναντήσαμε ένα άλλο ν-γραμμα να γειτονεύει μαζί του. Πχ αν είχαμε τις προτάσεις Ο Νίκος έφαγε μήλο. Ο Ηλίας έφαγε μήλο. Ο Νίκος έδωσε μήλο. παρατηρούμε ότι σε ένα ποσοστό 66% των περιπτώσεων μας το ν-γράμμα έφαγε συνορεύει με το ν-γραμμα μήλο. Η ακμή που ενώνει τους κόμβους έφαγε και μήλο θα μπορούσε να σταθμιστεί με την τιμή 0.66. Για το τι θα αναπαριστούν οι ακμές έχουν προταθεί διάφορες εκδοχές όπως το να υπάρχει μια λεπτομερή διανομή αποστάσεων μεταξύ κάθε ζεύγους κορυφών του γράφου ή να συνυπολογίσουμε έναν συντελεστή βάρους συνύπαρξης μεταξύ καθενός ζευγαριού ν-γραμμάτων που συνορεύουν.

Τέλος χρειαζόμαστε έναν βαθμό ομοιότητας μεταξύ των γράφων που έχουμε εξάγει. Για να καταλάβουμε την ομοιότητα μεταξύ των κειμένων από τα οποία έχουν προκύψει οι γράφοι. Οι δύο βασικές μεθοδολογίες είναι. Να συγκρίνουμε απλώς τους κόμβους μεταξύ των γράφων. Ή να συγκρίνουμε ολόκληρους τους γράφους δηλαδή τους κόμβους καθώς και τις ακμές που τους ενώνουν. Σίγουρα η εργασία να βγάζουμε συμπεράσματα από γράφους είναι δουλειά πιο συστηματοποιημένη και προσανατολισμένη σε πληροφοριακά συστήματα από ότι το να βγάζουμε συμπεράσματα από απλά κείμενα.

7.1 Πώς κατασκευάζεται ο Γράφος

Η κατασκευή του γράφου θα μπορούσε να χωριστεί σε δύο μεγάλα έργα. Το πρώτο έργο μας είναι να κατασκευάσουμε τους κόμβους του γράφου που θα είναι ουσιαστικά τα ν-γράμματα που έχουμε εξάγει από το αρχικό κείμενο. Έπειτα οι ακμές που θα συνδέουν τους κόμβους

αναμεταξύ τους καθώς και τα βάρη που θα έχει η κάθε μία. Και για τα δύο αυτά θέματα έχουν προταθεί διάφορες προσεγγίσεις που θα τις δούμε αναλυτικά.

Έχουμε ήδη πει ότι κάθε κόμβος του γράφου μας αντιστοιχεί σ' ένα ν-γραμμα. Επόμενο στάδιο είναι να πούμε το τι ορίζουμε ως ν-γραμμα. Ν-γραμμα ορίζεται μια υπακολουθία ν αντικειμένων από μια δοσμένη αρχικά ακολουθία. Τα αντικείμενα τις αρχικής ακολουθίας καθώς και της ανακολουθίας μπορεί να είναι φωνήματα, συλλαβές, γράμματα ή ακόμη και ολόκληρες λέξεις. Κάθε φορά το τι θα είναι εξαρτάται από την εφαρμογή. Ένα ν- γραμμα ενός στοιχείου αναφέρεται ως μονόγραμμα (unigram), δύο στοιχείων ως δίγραμμα (bigram), τριών στοιχείων ως τρίγραμμα (trigram) και από τέσσερα και πάνω πιο απλά ως ν-γράμματα (n-grams).

Για την πληρότητα του ορισμού να αναφέρουμε ότι ως ακολουθία ορίζουμε μια διατεταγμένη λίστα από αντικείμενα (ή γεγονότα). Υποακολουθία ορίζουμε μια ακολουθία που μπορεί μα παραχθεί από μια άλλη ακολουθία με το να διαγράψουμε κάποια στοιχεία χωρίς να αλλάξουμε την σειρά των στοιχείων που έχουν παραμείνει.

Ο κανονικοποιημένος ορισμός είναι:
Αν $n > 0$, $n \in Z$, και c_i είναι ο i-ος χαρακτήρας από μία l-μήκους σειρά χαρακτήρων $T^l = \{c1,c2,... ,cl\}$ (το κείμενο μας), τότε ένα ν- γράμμα χαρακτήρων $S^n = (s_1 , s_2 , . . . , s_n)$ είναι μια υποακολουθία μήκους n της $T^l \Leftrightarrow \exists i \in [1, l - n +1]$: $\forall j \in [1, n]$: $s_j = c_{i+j-1}$. θα αναφέρουμε ότι να ν-γράμματα εκτίονται από c_i έως το c_k , $k > i$, ως $S_{i,k}$, ενώ ν-γράμματα μήκους n θα δηλώνονται ως S_n .

Στην περίπτωση μας μπορούμε να θεωρήσουμε ένα κείμενο ως ένα σύνολο γραμμάτων ή συλλαβών ή λέξεων. Ένα ν-γραμμα θα μπορούσε να είναι μια σειρά λέξεων ή σειρά γραμμάτων. Σε κάθε περίπτωση είναι σημαντικό να διατηρούν την αρχική διάταξη σύμφωνα με το κείμενο.

Ν-γράμματα έχουν χρησιμοποιηθεί σε πολλές εφαρμογές όπως σε μηχανές μεταφράσεων, αναγνώριση συγγραφέων, αυτοματοποιημένες κατασκευές περιλήψεων, ακόμη και σε αξιολογήσεις περιλήψεων.

Στις υλοποιήσεις που θα παρουσιάσουμε παρακάτω θα χρησιμοποιήσουμε Ν-γράμματα που θα είναι υποακολουθίες γραμμάτων από το αρχικό κείμενο και όχι λέξεων. Αντίστοιχα θα μπορούσαμε να ενεργήσουμε και για λέξεις. Αυτό το κάνουμε γιατί τα συμπεράσματα που μπορούμε να βγάλουμε από την συνύπαρξη και την γειτονικότητα λέξεων είναι λιγότερα από ότι αν δουλέψουμε με ακολουθίες γραμμάτων Ο λόγος είναι ότι έτσι μπορούμε να δουλέψουμε καλύτερα με παραλλαγές των λέξεων και με τις ρίζες των λέξεων. Επίσης αν δουλεύαμε με ν-γράμματα λέξεων ο αριθμός των διαφορετικών κόμβων που θα είχαμε στον γράφο μας θα ήταν πολύ μεγαλύτερος με έμμεσες συνέπειες σε όλες τις εργασίες που θα θέλαμε να κάνουμε έπειτα στον γράφο μας.

7.2 Υλοποίηση των Κόμβων

Ας μπούμε πιο βαθιά στην σχεδίαση του αλγόριθμου. Ένα από τα χαρακτηριστικά που θέλουμε να έχει ο αλγόριθμος μας είναι να μην υπάρχει προ- επεξεργασία του αρχικού κειμένου. Δεν θέλουμε να μπούμε στην διαδικασία να αφαιρούμε σημεία στίξης και τα κενά μεταξύ των λέξεων.

Θα χρησιμοποιήσουμε τον όρο του παραθύρου για να ξεκαθαρίσουμε πια γράμματα θα αποτελούν ένα ν-γραμμα. Ένα παράθυρο είναι ένα σύνολο ν συνεχόμενων χαρακτήρων. Το παράθυρο αυτό θα διατρέχει το κείμενο μας μετατοπιζόμενο από την αρχή προς το τέλος ένα - ένα γράμμα κάθε φορά. Δεν θέλουμε κάθε ν-γράμμα μας να ξεκίνα από εκεί που τελείωσε το προηγούμενο άλλα να είναι τα ν-1 τελευταία γράμματα του προηγούμενου συν ένα επόμενο. Για να το πούμε με διαφορετικά λόγια τα ν-γράμματα σχηματίζονται από την ολίσθηση του παραθύρου πάνω στο αρχικό κείμενο κατά ένα γράμμα την φορά.

Είναι φανερό πως αυτή η μέθοδος έχει μια μορφή επικάλυψης γραμμάτων που μπορεί να παρομοιαστεί με την πράξη της συνέλιξης. Με αυτό τον τρόπο αποφεύγουμε την πράξη της κατάτμησης και ταυτόχρονα υπολογίζουμε ομοιότητες μεταξύ κειμένων χωρίς να έχουμε ορίσει συγκεκριμένα σημεία μεταξύ των κειμένων που να ταιριάζουν.

Η μέθοδος της κατάτμησης του κειμένου και η κατασκευή γράφου από αυτό δεν μπορεί να κρατήσει πληροφορίες σχετικά με την θέση των ν-γραμμάτων μέσα στο αρχικό κείμενο αλλά μόνο της γειτονικότητας που έχουν αναμεταξύ τους. Αυτό μας κάνει αδύνατη την ανασύσταση του αρχικού κειμένου από τον γράφο και δεν έχουμε καμιά πληροφορία αν τα δύο γειτονικά ν-γράμματα που έχουμε ποιο έπεται και ποιο προηγείται. Απλώς ξέρουμε με πια συχνότητα αυτά τα ν-γράμματα βρίσκονται κοντά στο κείμενο. Και σε αυτό έγκειται η δύναμη της μεθόδου αυτής. Δύο κείμενα με ν-γράμματα που γειτονεύουν με μεγάλη συχνότητα, θα έχουν παρόμοιους γράφους και ως αποτέλεσμα θα έχουν παρόμοια εννοιολογική σημασία, παρόμοιο νόημα, θα διαπραγματεύεται το ίδιο θέμα.

Ας χρησιμοποιήσουμε ένα αντιπαράδειγμα για να αποσαφηνίσουμε αυτή την μέθοδο. Έστω ότι έχουμε την πρόταση

“Does he write this example?”

Η μέθοδος που δεν θα ακολουθήσουμε στον αλγόριθμό μας είναι να κομματιάσουμε την πρόταση σε τριγράμματα που το ένα ξεκινά να δημιουργείται έπειτα από το τελευταίο γράμμα του προηγούμενου. Σε αυτήν την περίπτωση θα είχαμε τα εξής ν-γράμματα

“Doe” “s h” “e w” “rit” “e t” “his” “ ex” “amp” “le?”

Η μέθοδος που θα ακολουθήσουμε και περιγράψαμε εκτενώς πιο πάνω από την ίδια

πρόταση θα έβγαζε τα εξής ν- γράμματα

“Doe” “oes” “s h” “ he” “he ” “e w” “ wr” “wri” “rit” “ite” “te ” “e t” “ th” “thi” “his” “is ” “s e” “ ex” “exa” “xam” “amp” “mpl” “ple” “le?”

που από ότι βλέπουμε είναι κάτι εντελώς διαφορετικό από αυτά που εξάγαμε πριν.

Ό αλγόριθμος σε ψευδογλώσσα που υλοποιεί αυτήν την μέθοδο εξαγωγής ν- γραμμάτων από ένα κείμενο είναι ο ακόλουθος

Algorithm 1. Extraction of n-grams
Input: text
Output: n-gram set
// T is the text we analyse
1 for all i in [1,length(T)-n+1] do
2 get substring of T from index i to i+n-1
3 end

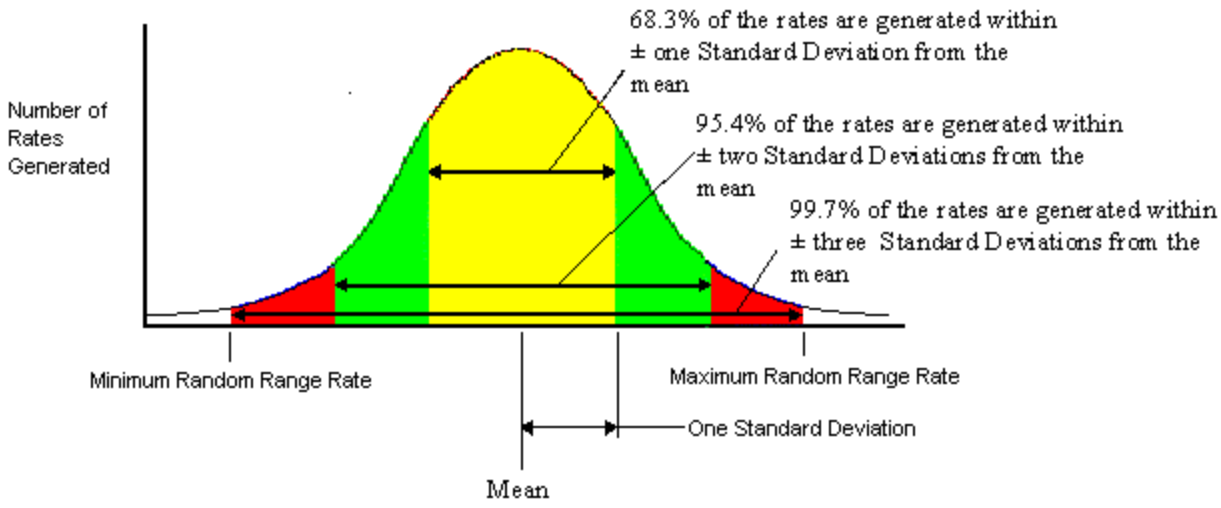
7.3 Υλοποίηση των Ακμών

Ο τρόπος που μπορούν να κατασκευαστούν οι ακμές των ν- γραμμάτων βασίζεται στο πώς η γειτονικότητα των κοντινών ν-γραμμάτων υπολογίζεται σε ένα κείμενο. Πάλι θα χρησιμοποιήσουμε τον όρο του παραθύρου αυτή την φορά όχι για να μιλήσομε για γειτονικούς χαρακτήρες αλλά για να ξεκαθαρίσουμε ποια είναι τα γειτονικά ν-γράμματα. Σε γενικές γραμμές μπορούμε να πούμε πως έχουμε ένα συγκεκριμένο παράθυρο ν-γραμμάτων γύρω από ένα δοσμένο Ν-γράμμα Ν0. Όλα τα ν-γράμματα που υπάρχουν μέσα σε αυτό το παράθυρο μπορούν να θεωρηθούν ως γείτονες αυτού του αρχικού Ν-γράμματος Ν0. Η γειτονικότητα των Ν-γράμματων αυτών μπορεί να αναπαρασταθεί σαν συνδεδεμένες κορυφές στον γράφο μας. Λέγοντας το διαφορετικά η γειτονικότητα δύο ν-γραμμάτων αναπαρίσταται με μια ακμή που συνδέει τις δύο αντίστοιχες κορυφές. Οι ακμές που θα συνδέουν τις κορυφές των ν-γράμματων θα έχουν ένα βάρος που θα υποδηλώνει ή την απόσταση μεταξύ γειτονικών κορυφών ή έναν αριθμό ανάλογο με το πόσες φορές τα ν-γράμματα που βρίσκονται στις άκρες της ακμής βρέθηκαν να γειτονεύουν στο αρχικό κείμενο. Οι τρεις υλοποιήσεις που έχουμε για τις ακμές των κόμβων βασίζονται στους διάφορους τύπους παραθύρων που μπορούμε να χρησιμοποιήσουμε. Παρακάτω παραθέτουμε τις κυριότερες.

- Μη συμμετρική προσέγγιση: Σε αυτήν την περίπτωση έχουμε ένα παράθυρο μήκους n όπου διατρέχει το κείμενό μας. Αν το N_0 ν-γράμμα μας βρίσκεται στην θέση p_0 τότε το παράθυρο θα καλύψει ένα εύρος από p_0-1 έως p_0-n. Σε αυτήν την περίπτωση παίρνουμε υπόψιν μας μόνο προηγούμενους χαρακτήρες. Την κάθε γειτονική ακμή του κόμβου που αντιστοιχεί στο N_0 την βαθμονομούμε με ένα αριθμό που εξαρτάται από τα γειτονικά ν-γράμματα που βρίσκονται μέσα στο παράθυρο αυτό.

- Συμμετρική προσέγγιση: Και εδώ έχουμε ένα παράθυρο μήκους n που διατρέχει το αρχικό μας κείμενο αλλά αυτή την φορά κεντραρισμένο στην αρχή του ν-γράμματος μας N_0. Υποθέτοντας ότι το ν-γράμα μας N_0 βρίσκεται στην θέση p_0 τότε το παράθυρο μας θα εκτείνεται από την θέση p_0 - [n/2] έως την θέση p_0 + [n/2] (το [x] δίνει την ακέραια ποσότητα του αριθμού x) . Με αυτόν τον τρόπο παίρνουμε υπόψιν μας και τους προηγούμενους αλλά και τους χαρακτήρες που έπονται από την θέση p_0 που βρίσκεται το N_0. Κάθε γειτονική ακμή της κορυφής που αντιστοιχεί στο ν-γράμμα την βαθμονομούμε όπως και πριν με βάση τα γειτονικά ν-γράμματα που βρίσκονται μέσα στο παράθυρο αυτό.

- Κανονικοποιημένη Γκαουσιανή συμμετρική προσέγγιση: Η προσέγγιση αυτή είναι πιο περίπλοκη από τις προηγούμενες δύο. Εδώ έχουμε ένα παράθυρο μήκους 3 x n το οποίο διατρέχει όλο το κείμενο. Το παράθυρο αυτό είναι κεντραρισμένο στη αρχή του ν-γράμματος N_0 στην θέση p_0. Το παράθυρο εκτείνεται από την θέση $p_0 – [(3 \text{ x } n)/2]$ έως την θέση $p_0 + [(3 \text{ x } n)/2]$. Πάλι παίρνουμε υπόψιν μας προηγούμενους και επόμενους χαρακτήρες από την θέση p_0 που βρίσκεται το N_0. Η διαφορά εδώ είναι ότι παίρνουμε υπόψιν μας τις αποστάσεις των γειτονικών ν-γραμμάτων από το τρέχων N_0 ν-γράμμα. Όσο πιο κοντά είναι ένα ν-γράμμα N_1 στο N_0 με τόσο μεγαλύτερο αριθμό θα βαθμονομηθεί η ακμή που θα συνδέει τον κόμβο που αντιστοιχεί στο N_0 με τον κόμβο που αντιστοιχεί στο N_1. Αντίστοιχα όσο πιο μακριά είναι με τόσο μικρότερο βαθμό θα γίνει η βαθμονόμιση. Είναι άξιο παρατήρησης ότι στην μέθοδο αυτήν έχουμε πάρει χαρακτήρες έξω από το προκαθορισμένο μέγεθος παράθυρου D_{win}. Το εύρος από όπου παίρνουμε χαρακτήρες είναι $3xD_{win}$. Αυτή την επιλογή την κάναμε γιατί παίρνοντας ένα παράθυρο τριπλάσιο του Dwin μπορούμε να έχουμε το 99.70% των τιμών της Γκαουσιανής κατανομής (όπως βλέπουμε και στο σχήμα 1). Σε αυτή την περίπτωση θεωρούμε το D_{win} ίσο με μία τυπική απόκλιση. Η Κανονικοποιημένη Γκαουσιανή συμμετρική προσέγγιση μπορεί να αποκλείσει ν-γράμματα που δεν συσχετίζονται με ν-γράμμα μας N_0 και να πάρει υπόψιν της κατά ένα ποσοστό 99.70% όλα τα ν-γράμματα που σχετίζονται με μία τιμή βαρύτητας ανάλογη με το πόσο γειτονεύουν με το N_0.



Σχήμα 2: Τρεις τυπικές αποκλίσεις επί της Γκαουσιανής Κατανομής

Ας πάρουμε για παράδειγμα ότι έχουμε το κείμενο abcdefghi από αυτό μπορούν να δημιουργηθούν τα εξής 3-γράμματα N_1=abc N_2=bcd N_3=cde N_4=def N_5=efg N_6=fgh N_7=ghi
η σειρά ν-γράμματων μαζί με τις αντίστοιχες θέσεις τους φαίνεται παρακάτω.

$$N_1\ N_2\ N_3\ N_4\ \underline{N_5}\ N_6\ N_7$$
$$p_1\ \ p_2\ p_3\ \ p_4\ p_5\ p_6\ \ p_7$$

θεωρούμε παράθυρο n=4, το 3-γραμμα που μας ενδιαφέρει είναι το N_5 που βρίσκεται στην πέμπτη θέση p_5. Στην Μη συμμετρική προσέγγιση η γειτονικότητα μπορεί να υπολογιστεί ως p_5-1 = η θέση p_4 έως p_5-4 = η θέση p_1 οπότε με έντονα γράμματα έχουμε τα 3-γράμματα που βρίσκονται στο παράθυρο

$$\mathbf{N_1\ N_2\ N_3\ N_4}\ \underline{N_5}\ N_6\ N_7$$
$$p_1\ p_2\ p_3\ p_4\ p_5\ p_6\ p_7$$

Στην συμμετρική προσέγγιση η γειτονικότητα μπορεί να υπολογιστεί από p_5 - [4/2]= θέση p_3 έως p_5 + [4/2] = θέση p_7. Οπότε βλέπουμε τα ν-γράμματα που βρίσκονται στο παράθυρο.

$$N_1\ N_2\ \mathbf{N_3}\ \mathbf{N_4}\ \underline{\mathbf{N_5}}\ \mathbf{N_6}\ \mathbf{N_7}$$
$$p_1\ p_2\ p_3\ p_4\ p_5\ p_6\ p_7$$

Στην Κανονικοποιημένη Γκαουσιανή συμμετρική προσέγγιση έχουμε ένα μεγαλύτερο παράθυρο να εκτείνεται από την θέση $p_5 - [(3 \text{ x } 4)/2] = p_{-1}$ έως $p_5 + [(3 \text{ x } 4)/2] = p_{11}$. Όσο πιο μεγάλα γράμματα έχουμε στα τα ν-γράμματα μας τόσο πιο μεγάλο συντελεστή βαρύτητας έχουμε πάρει. Οπότε πάλι παρατηρούμε ότι τα ν γράμματα που είναι πιο κοντά στο N_5 υπολογίζονται με μεγαλύτερο συντελεστή βαρύτητας από ότι τα πιο απομακρυσμένα παρότι όλα βρίσκονται στο ίδιο παράθυρο.

$$N_{-2}\ \mathbf{N_{-1}}\ \mathbf{N_0}\ \mathbf{N_1}\ \mathbf{N_2}\ \mathbf{N_3}\ \mathbf{N_4}\ \underline{\mathbf{N_5}}\ \mathbf{N_6}\ \mathbf{N_7}\ \mathbf{N_8}\ \mathbf{N_9}\ \mathbf{N_{10}}\ \mathbf{N_{11}}\ N_{12}$$
$$p_{-2}\ p_{-1}\ p_0\ p_1\ p_2\ p_3\ p_4\ p_5\ p_6\ p_7\ p_8\ p_9\ p_{10}\ p_{11}\ p_{12}$$

Έπειτα από ένα σύνολο πειραμάτων που έχει γίνει βρέθηκε πως η καλύτερη προσέγγιση είναι η συμμετρική και αυτή θα χρησιμοποιηθεί στην μετέπειτα υλοποίηση του αλγορίθμου μας.

Ας μπούμε όμως πιο βαθιά στην διαδικασία βαθμονόμησης των ακμών που έχουμε επιλέξει και σύμφωνα με αυτό θα υλοποιήσουμε τον αλγόριθμο μας. Κάθε ακμή που συνδέει δυο κορυφές N_i N_j έχει ένα βάρος c_{ij} που υποδηλώνει των αριθμό των φορών που τα αντίστοιχα δύο ν-γράμματα τυχαίνει να είναι γειτονικά αναμεταξύ τους μέσα σε κάποια απόσταση D_{win}. Το D_{win} είναι ένας συγκεκριμένος αριθμός ν-γραμμάτων που βάζει ένα όριο σε πια ν-γράμματα θεωρούμε γειτονικά και πια τα θεωρούμε τόσο απομακρυσμένα που να μην τα λαμβάνουμε υπόψη μας. Το D_{win} είναι το μέγεθος του παραθύρου για το οποίο μιλούσαμε πιο πριν

Σε αυτό το σημείο καλούμαστε να αναλογιστούμε τι μέγεθος παραθύρου να έχουμε καθώς και τι μέγεθος ν-γράμματος να επιλέξουμε ανεξάρτητα από κάποιο συγκεκριμένο κείμενο ή γλώσσα. Αν επιλέξουμε ένα μικρό ν για το ν-γράμμα μας, θα πρέπει έπειτα να έχουμε ένα μεγαλύτερο μέγεθος παραθύρου. Το μέγεθος του παραθύρου όμως δεν μπορεί να είναι αρκετά μεγάλο γιατί τότε χάνεται το πραγματικό νόημα τις γειτονικότητας δύο ν-γραμμάτων. Το μειονέκτημα του πολύ μεγάλου μεγέθους παραθύρου είναι ότι είτε δύο ν-γράμματα είναι διπλά είτε απέχουν εκατό θέσεις ο αλγόριθμος και η αναπαράσταση μας στον γράφο θα τα αντιμετωπίζει με τον ίδιο τρόπο.

Επιλέγοντας ένα μεγάλο ν για το ν-γράμμα μας κινδυνεύουμε να συσσωρεύομαι πολύ άχρηστη πληροφορία η οποία θα μας επιβαρύνει και από πλευράς μνήμης και από πλευράς

επεξεργαστικής ισχύς στο μετέπειτα πληροφοριακό μας σύστημα που θα κληθούμε να υλοποιήσουμε. Αυτό οφείλεται στο ότι αυξάνοντας το ν αυξάνεται κατά πολύ ο αριθμός των διαφορετικών ν-γράμματων και κατ' αντιστοιχία του γράφου που θα τα αναπαραστήσει. Ένας τρόπος που έχει προταθεί για να αντιμετωπίσουμε αυτό το πρόβλημα είναι αν επιλέξουμε να έχουμε ν-γράμματα μεταβλητού μήκους και ένα παράθυρο D_{win} αρκετά μικρό περίπου στο μέσο μέγεθος μιας λέξης.

7.4 Μέθοδοι Σύγκρισης μεταξύ Γράφων

Παρακάτω θα παρουσιάσουμε δύο μεθόδους όπου μπορούμε να συγκρίνουμε γράφους αναμεταξύ τους και να βρίσκουμε κατά πόσο είναι όμοιοι Αυτές οι μέθοδοι έχουν μεγάλες πρακτικές εφαρμογές. Μέσω αυτών μπορούμε να καταλάβουμε κατά πόσο δύο κείμενα διαπραγματεύονται παρόμοια θέματα. Κείμενα με παρόμοια θεματολογία αναμένουμε να έχουνε και παρόμοιους γράφους ν-γραμμάτων.

Σε ένα ακόμη πιο προχωρημένο επίπεδο πιθανόν να μπορούσαμε να έχουμε μια λίστα γράφων όπου για κάθε θεματολογία να έχουμε έναν γράφο που να την αντιπροσωπεύει. Έπειτα δημιουργώντας τον γράφο από ένα τυχαίο κείμενο και συγκρίνοντας τον με αυτή την λίστα να μπορούμε με έναν αυτοματοποιημένο τρόπο να καταλάβουμε ποιο θέμα διαπραγματεύεται το κείμενο.

Οι δύο μέθοδοι είναι η Ομοιότητα συνύπαρξης (Coocurrence Similarity) και η ομοιότητα κατά αξία (value similarity) και οι δύο μπορούν να εφαρμοστούν σε οποιοδήποτε γράφο. Παρακάτω τις παρουσιάζουμε αναλυτικά.

7.4.1 Ομοιότητα Συνύπαρξης (Coocurence Similarity) CS

Έχοντας δύο γράφους η μέθοδος της ομοιότητας συνύπαρξης ή αλλιώς ομοιότητα περιεχομένου (Containment Similarity) μετρά πόσες από τις ακμές που υπάρχουν στον ένα γράφο υπάρχουν επίσης και στον άλλον. Η μέθοδος αυτή δεν παίρνει καθόλου υπόψιν της την βαρύτητα που μπορεί να έχει κάθε ακμή. Αν δύο ισοσταθμισμένοι γράφοι έχουν ακριβώς τους ίδιους κόμβους και ακμές αλλά διαφορετικά βάρη στις ακμές θα φανούν ότι είναι ακριβώς ίδιοι. Η μέθοδος αυτή ορίζεται με την συνάρτηση 1 που έχει ως σύνολο τιμών έναν αριθμό μεταξύ του μηδέν και του ένα. Το μηδέν υποδηλώνει ότι είναι εντελώς διαφορετικοί οι δυο γράφοι. Δεν υπάρχει ακμή και στους δύο γράφους που να συνδέει τους δύο αντίστοιχους κόμβους. Το ένα

υποδηλώνει ότι οι δύο γράφοι είναι ίδιοι δηλαδή όλες οι ακμές συνδέουν τους αντίστοιχους κόμβους. Η συνάρτηση αυτή είναι η

$$\mathrm{CS}(G^i, G^j) = \frac{\sum_{e \in G^i} \mu(e, G^j)}{\max(|G^i|, |G^j|)}.$$

ισχύει ότι e Є $E^G \equiv$ e ЄG
μ = 1 αν e ανήκει στο G
μ = 0 αν e δεν ανήκει στο G
όπου $|G^i|$ και$|G^j|$ είναι ο αριθμός των ακμών που ανήκουν στον γράφο G^i και G^j αντίστοιχα
Κάθε ακμή που ταιριάζει μεταξύ των δύο γράφων προσφέρει στο συνολικό άθροισμα κατά

$$\frac{1}{\max(|H_1|,|H_2|)}$$

Από τον ορισμό της CS φαίνεται ότι η CS είναι συμμετρική συνάρτηση.

$$CS(G^i, G^j) = CS(G^j, G^i).$$

7.4.2 Ομοιότητα κατά Αξία (Value Similarity) VS

Η ομοιότητα κατά αξία παίρνει υπόψιν της και τα βάρη που έχουν οι ακμές στους ισοσταθμισμένους γράφους. Εδώ μας ενδιαφέρει όλες οι ακμές να συνδέουν τους αντίστοιχους κόμβους και στους δύο γράφους αλλά και κατα πόσο οι ακμές έχουν τα ίδια βάρη. Πλήρης ομοιότητα μεταξύ κόμβων, ακμών και βαρών μας δίνει αποτέλεσμα ένα. Η μέθοδος αυτή είναι συμμετρική και η συνάρτηση είναι η

$$\mathrm{VS}(G^i, G^j) = \frac{\sum_{e \in G^i} \left(\mu(e, G^j) \times \frac{\min(w_e^i, w_e^j)}{\max(w_e^i, w_e^j)} \right)}{\max(|G^i|, |G^j|)}.$$

Κάθε ακμή e που ταιριάζει και έχει ένα βάρος w^i_e στον γράφος G^i συνεισφέρει κατά ένα ποσοστό στο συνολικό άθροισμα

$$\frac{VR(e)}{max(|G^i|,|G^j|)}$$

Το VR(e) ορίζεται ως

$$\mathrm{VR}(e) = \frac{\min(w_e^i, w_e^j)}{\max(w_e^i, w_e^j)}.$$

Κάθε φορά που η ακμή δεν ταιριάζει μεταξύ των δύο γράφων δεν την παίρνουμε υπόψιν μας στο άθροισμα.

Από έρευνες που έγιναν σε ν-γράμματα χαρακτήρων η μέθοδος σύγκρισης κατά αξία βρέθηκε ότι υπερτερεί σε σχέση με την μέθοδο συνύπαρξης. Αυτό βεβαία το αναμέναμε αφού η μέθοδος κατά αξία παίρνει υπόψιν της και τα βάρη των ακμών. Βεβαία η μέθοδος κατά αξία είναι ποιο περίπλοκη στην υλοποίηση και πιο χρονοβόρα στης πράξεις για να μας βγάλει το αποτέλεσμα.

Ο συνδυασμός ν-γράμματα χαρακτήρων με μέθοδο αναπαράστασης γράφων και σύγκριση κατά αξία έχει τα καλύτερα αποτελέσματα σε σύγκριση με το να χρησιμοποιούσαμε λέξεις ή ιστογράμματα που θα περιγράψουμε παρακάτω ή την ομοιότητα συνύπαρξης

7.4.3 Ομοιότητα κατά Αξία σε N-Gram Histograms

Μια παραπλήσια μέθοδος είναι η μέθοδος των N-Gram ιστογραμμάτων. Τα ν-γράμματα μπορούν να κατασκευαστούν ακριβώς όπως και στην περίπτωση του N-Gram Graph. Εδώ θα χρησιμοποιήσουμε ένα απλό ιστόγραμμα συχνοτήτων όπου απλά αναπαριστούμε την συχνότητα όπου βρέθηκε το κάθε ν-γραμμα. Με αυτή την μέθοδο δεν μπορούμε να έχουμε πληροφορία για το ποια ν-γράμματα βρίσκονται κοντά αναμεταξύ τους. Αυτό κάνει πιο αδύναμη αυτή την μέθοδο. Άλλα ταυτόχρονα πολύ πιο απλή στην υλοποίηση, στον σχεδιασμό και στις απαιτήσεις πόρων σε ένα υπολογιστικό σύστημα.

Η μέθοδος των N-gram histograms υλοποιείται ως εξής. Ένα παράθυρο μήκους ν διατρέχει όλο το αρχικό μας κείμενο σχηματίζοντας ν-γράμματα με έναν από τους τρόπους που

περιγράψαμε πιο πάνω. Για κάθε καινούριο Ν-γράμμα που συναντάμε βάζουμε μια καινούρια στήλη στο ιστόγραμμα μας που το αντιπροσωπεύει και τιμής ένα. Αν το έχουμε ξανά συναντήσει αυτό το ν-γραμμα απλός αυξάνουμε κατά ένα την τιμή της αντίστοιχης στήλης του ν-γράμματος. Στο τέλος μπορούμε να διαιρέσουμε όλες τις τιμές που έχουμε στα ν-γράμματα με το άθροισμα των τιμών των ν-γραμμάτων για να έχουμε μια πιο κανονικοποιημένη τιμή.

7.5 Μέθοδοι Σύγκρισης μεταξύ Ιστογραμμάτων

Όπως έχουμε μεθόδους σύγκρισης μεταξύ γράφων όπου τις παρουσιάσαμε πιο πριν έτσι έχουμε και τις αντίστοιχες μεθόδους σύγκρισης μεταξύ ιστογραμμάτων.

7.5.1 Ομοιότητα Συνύπαρξης (Coocurence Similarity) CSH σε Ιστογράμματα

Η μέθοδος αυτή βασίζεται στο κατά πόσο ένα στοιχείο h1 από το ιστόγραμμα H1 υπάρχει και στο ιστόγραμμα H2 ανεξάρτητα με τις τιμές που μπορούν να έχουν τα h1 και h2. Κάθε φορά που υπάρχουν τα αντίστοιχα στοιχειά στα δύο ιστογράμματα συμβάλουν στην συνάρτηση μας κατά ένα ποσοστό

$$\frac{1}{\max(|H_1|,|H_2|)}$$

Η συνάρτηση είναι η

$$\mathrm{CS}_H(H_1, H_2) = \sum_{h \in H_1} \frac{\mu(h, H_2)}{\max(|H_1|, |H_2|)},$$

όπου
$\mu(h,H_2) = 1$ αν το h υπάρχει και στα δύο ιστογράμματα H_1 H_2

$\mu(h,H_2) = 0$ αν το h δεν υπάρχει και στα δύο ιστογράμματα H_1 H_2
και $|H_1|$ $|H_2|$ είναι το πλίθος των στοιχείων που έχει το H_1 και το H_2

7.5.2 Ομοιότητα κατά Αξία (Value Similarity) VS_H σε Ιστογράμματα

Με παρόμοιο τρόπο όπως και πριν μόνο που εδώ συνυπολογίζουμε την αξία $v(h,H_2)$ που έχει ένα στοιχείο στο ιστόγραμμα μας παραθέτουμε την συνάρτηση

$$\mathrm{VS}_H(H_1, H_2) = \sum_{h \in H_1} \frac{\mu(h, H_2) \times v(h, H_2)}{\max(|H_1|, |H_2|)}.$$

7.6 Σύγκριση N-Gram Graphs με N-Gram Histograms σε ν-γράμματα Χαρακτήρων

Η σύγκριση των N-Grams γράφων έναντι των N-Grams ιστογραμμάτων για χαρακτήρες έχει γίνει χρησιμοποιώντας τους δυο αλγόριθμους σύγκρισης γραφημάτων και ιστογραμμάτων. Από τα πειράματα και την έρευνα που έχει γίνει έχουμε καταλήξει ότι η χρίση των N-Grams γράφων υπερτερεί έναντι των ιστογράμμων έχοντας ως απαραίτητη προϋπόθεση ότι δεν έχουμε επιλέξει υπερβολικά μεγάλες τιμές για τα D_{win}, L_{min} και L_{max}.

Μια μεγάλη τιμή στο D_{win} δεν επηρεάζει καθόλου την μέθοδο των ιστογράμμων. Αφού το Dwin δεν χρησιμοποιείται και χρησιμοποιούνται μόνο οι παράμετροι L_{min} και L_{max}. Από την άλλη πλευρά βλέπουμε ότι στην μέθοδο των γράφων μια μεγάλη τιμή στο Dwin θα έχει ως αποτέλεσμα να θεωρούμε γειτονικά ν-γράμματα που δεν είναι στην πραγματικότητα κοντά και να αλλοιώνεται η ποιότητα των αποτελεσμάτων με αυτή την μέθοδο. Μια μεγάλη τιμή στο D_{win} μπορεί να αναδείξει την μέθοδο των N-Grams γράφων χειρότερη από των N-Grams ιστογραμμάτων.

7.7 Χρίση Λέξεων αντί για Χαρακτήρες στις μεθόδους N-Gram Graphs και N-Gram Histograms

Στις μεθόδους Ν-γραμμάτων είτε με γράφους είτε με ιστογράμματα μπορούμε αντί να χρησιμοποιήσουμε χαρακτήρες να χρησιμοποιήσουμε λέξεις. Αυτό το έχουμε αναφέρει και πιο πριν καθώς και το ότι η μέθοδος των ν-γράμματων με χαρακτήρες υπερτερεί στις περισσότερες περιπτώσεις έναντι της μεθόδου των γραμμάτων με λέξεις. Αυτή την άποψη την στηρίξαμε και την αιτιολογήσαμε όταν δώσαμε τον ορισμό των γραμμάτων

Στην περίπτωση που επιλέξουμε να υλοποιήσουμε την μέθοδο των ν- γραμμάτων με λέξεις η μέθοδος αναπαράστασης και χρίσης ιστογραμμάτων έχει καλύτερα αποτελέσματα όπως έδειξαν πειράματα και είναι κατά πολύ πιο εύκολη στην υλοποίηση.

Η μέθοδος ν-γραμμάτων με λέξεις στην αναπαράσταση με ιστογράμματα έχει και αυτή τις δύο παραμέτρους από όπου εξαρτάται η επίδραση της (L_{min}, L_{max}) όπως και όταν χρησιμοποιήσαμε χαρακτήρες. Η μέθοδος αυτή είναι κατά πολύ πιο εξαρτημένη από το αντίστοιχο L_{min} ενώ το L_{max} φαίνεται να μην επηρεάζει καθόλου τις επιδόσεις του συστήματος. Παρόλα αυτά πάντα προτιμάμε να μην έχουμε πολύ μεγάλο L_{max} για να μην αυξάνεται κατά πολύ ο αριθμός των υπολογισμών.

Τέλος η καλύτερη μέθοδος σύγκρισης ιστογράμμων όταν χρησιμοποιούμε ν-γράμματα λέξεων είναι η ομοιότητα κατά αξία. Η ομοιότητα συνύπαρξης έχει λίγο χειρότερες επιδόσεις.

8 Προγραμματιστική Υλοποίηση του Recommendation System που βασίζεται στο 3-Gram Graph.

Έχουμε προγραμματιστικά υλοποιήσει ένα Recommendation σύστημα στην γλώσσα Java. Παρακάτω θα περιγράψουμε την γενική δομή του, τις κλάσεις του, τις μεθόδους του, τα δεδομένα που δέχεται ως είσοδο, τον τρόπο που τα επεξεργάζεται και τις εξόδους που παράγει.

Η γενική ιδέα του προγράμματος είναι να δέχεται δύο αρχεία τα οποία θα έχουν tweets από δύο χρήστες. Για κάθε αρχείο θα παράγει έναν γράφο. Ο γράφος αυτός θα έχει ως κόμβους τριγράμματα από το αρχικό αρχείο και ακμές που θα αντιστοιχούν στην γειτονικότητα που βρίσκονται τα τριγράμματα αυτά στα αρχικά tweets. Έπειτα θα συγκρίνονται οι δύο γράφοι και έτσι θα έχουμε μια πολύ καλή προσέγγιση του κατά πόσο αυτά τα δύο αρχεία ταιριάζουν νοηματικά. Για την παράσταση του γράφου χρησιμοποιήσαμε δύο αρχεία, ένα για τους κόμβους και ένα για τις ακμές. Για την σύγκριση των γράφων υλοποιήσαμε δύο μεθόδους την value similarity και την containment similarity. Παρακάτω θα παρουσιάσουμε αναλυτικά κάθε μέρος του προγράμματος.

8.1 public class Thesis

Η κλάση από όπου ξεκινά το πρόγραμμα μας είναι η κλάση Thesis. Η κλάση αυτή έχει μόνο μία μέθοδο την main. Σκοπός της είναι να καλωσορίσει των χρήστη στο πρόγραμμα, να κάνει μια απεικόνιση των γράφων που έχουν παραχθεί από τα tweets και να παροτρύνει τον χρήστη να επιλέξει ποια μέθοδο σύγκρισης γράφων επιθυμεί.

8.1.1 public static void main(String[] args)

Η μέθοδος main έχει αρχικά μια σειρά String μεταβλητών που περιέχουν τα ονόματα των

αρχείων που θα χρησιμοποιηθούν. Αυτό μας προσφέρει την ευκολία άμα χριαστεί να αλλάξουμε το όνομα ενός αρχείου να μην χρειάζεται να ανατρέχουμε και να το αλλάζουμε πολλές φορές στον κώδικά αλλά μόνο μια φορά στην αντίστοιχη μεταβλητή. Έπειτα καλούμε την μέθοδο Input.main_Input(String,String,String) μια φορά για κάθε αρχείο με tweets που έχουμε. Με αυτή την κλήση παίρνουμε τα δύο αρχεία κόμβων και ακμών που χρειαζόμαστε για την αναπαράσταση των γράφων. Στη συνέχια δίνουμε στον χρήστη μια αναπαράσταση των γράφων όπου με χρήση αριθμών αναφέρουμε κάθε κόμβος του γράφου με ποιον άλλον κόμβο συνδέεται. Αυτό το κάνουμε και για τους δύο γράφους. Τέλος προτρέπουμε τον χρήστη να επιλέξει ποια μέθοδο σύγκρισης επιθυμεί εισαγάγωντας απο το πληκτρολόγιο το γράμμα 'C' για την containment similarity και το γράμμα 'V' για την Value similarity.

Ενδιαφέρον παρουσιάζει ότι καλούμε την μέθοδο simpleGraph(etring,String) από την κλάση Compare2Graphs ούτος ώστε τον γράφο που έχουμε στης μεταβλητές G1 και G2 από εκει που έχει παράλληλες ακμές να τον κάνουμε απλό. Απλό γράφο χρειαζόμαστε στην Containment similarity γιατί η μέθοδος σύγκρισης αυτή χειρίζεται απλούς γράφους. Στην value similarity αν και επεξεργαζόμαστε τον γράφο με βάρη χρειάζεται και ο απλός γιατί με την χρήση του απλού θα γίνει η διάσχιση από όλες τις ακμές και από τον γράφο με τις παράλληλες ακμές θα βρίσκουμε τα βάρη των ακμών.

Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
public static void main(String[] args) throws Exception
{ // Nodes1.txt contains all the Nodes-3grams and Edges1.txt contains all the edges of //Nodes.
        String tweet1="Tweet1.txt",nodes1="Nodes1.txt",edges1="Edges1.tx
tweet2="Tweet2.txt", nodes2="Nodes2.txt", edges2="Edges2.txt";
        String simpleEdges1= "SimpleEdges1.txt", simpleEdges2= "SimpleEdges2.txt";
        Input.main_Input(tweet1,nodes1,edges1);
// It creates Nodes1.txt and Edges1.txt from Tweet1.txt
        Input.main_Input(tweet2,nodes2,edges2);
// It creates Nodes2.txt and Edges2.txt from Tweet2.txt

        System.out.println("\n\n\n---Tweet1--- \nPresentation of Nodes and Edges: ");
        In in1 = new In(edges1);
// The constructor of In takes the argument "Edges1.txt"
    Graph G1 = new Graph(in1);
// The constructor of G takes the object in1 of class In.
    StdOut.println(G1);
// It represents all the nodes and the edges which adjoin each Node.

    System.out.println("\n\n\n---Tweet2--- \nPresentation of Nodes and Edges: ");
    In in2 = new In(edges2);// Same as above
    Graph G2 = new Graph(in2);
```

```
                StdOut.println(G2.toString());
                System.out.println("\n\n\n");

                char c='a'; // It chooses the Similarity method I prefer.
                do{
                if((c>='a'&&c<='z')||(c>='A'&&c<='Z')){
                   System.out.print("If you want to use Value       Similarity press V\nIf you want t
Containment Similarity press C\ninput>_");
                  c = (char) System.in.read();
                }else c = (char) System.in.read();

                if(c=='V'||c=='v'){ // Value Similarity
                  Compare2Graphs.simpleGraph(edges1,simpleEdges1);
                  In in3 = new In(simpleEdges1);
   // It makes simpleEdges from Edges and puts them in a string. So we have the simple //g
without weights.
                  Graph SimpleG1 = new Graph(in3);
 // SimpleG1.toString is the Simple Graph without weights.
                  Compare2Graphs.simpleGraph(edges2,simpleEdges2);
                  In in4 = new In(simpleEdges2);// Same as above
                  Graph SimpleG2 = new Graph(in4);

System.out.println(Compare2Graphs.valueSimilarity(G1.toString(),G2.toString(),SimpleG1.toS
, SimpleG2.toString(), nodes1, nodes2));
                  break;
 // It takes the weighted Graphs, the simple Graphs, the files with the nodes, It compares //then
takes the Value Similarity.
                }
                if(c=='C'||c=='c'){ // Containment Similarity
                  Compare2Graphs.simpleGraph(edges1,simpleEdges1);
                  Compare2Graphs.simpleGraph(edges2,simpleEdges2);
                  Compare2Graphs.contSimilarity(nodes1,simpleEdges1,nodes2,simpleEdges2);
                  break;
 // In the case of Containment Similarity is needed only the Simple Graphs and the Nodes.
                }
                }while(true);
            }
```

8.2 public class Input

Η κλάση Input είναι η κλάση με την οποία διαβάζονται τα tweets από ένα αρχείο και σχηματίζεται ο γράφος. Τα tweets βρίσκονται σε ένα αρχείο χαρακτήρων και τα χωρίζει μια αλλαγή γραμμής. Είναι σημαντικό να σημειώσουμε ότι από το πρόγραμμα μας δεν γίνονται δεκτά tweets μικρότερου μεγέθους από πέντε χαρακτήρες. Για κάθε αρχείο με tweets που θα διαβάσει παράγει δύο αρχεία για να αναπαραστήσει τον γράφο. Ένα αρχείο με τους κόμβους που αντιστοιχούν στα τριγράμματα που παρήχθησαν και ένα αρχείο με της ακμές που δείχνουν την γειτονικότητα στα τριγράμματα. Η κλάση Input αποτελείται από τέσσερις μεθόδους την main_Input, την compare, την prepend, και την initialize

8.2.1 public static void main_Input (String tweetFile, String nodesFile, String edgesFile)

Στην μέθοδο main_Input γίνεται η ανάγνωση των δεδομένων από το αρχείο που περιέχει τα tweets. Η μέθοδος δέχεται ως είσοδο το όνομα του αρχείου από όπου θα διαβαστούν τα tweets και τα ονόματα των αρχείων που θα γραφτούν οι κόμβοι και οι ακμές αντίστοιχα.

Στην αρχή ανοίγει το αρχείο από το οποίο θα γίνει η ανάγνωσή των tweets. Έπειτα διαβάζει το πρώτο τρίγραμμα. Η μέθοδος compare() που ακολουθεί ελέγχει αν αυτό το τρίγραμμα υπάρχει είδη. Αν δεν υπάρχει το βάζει στο τέλος του αρχείου των κόμβων. Αν υπάρχει δεν γράφει κάτι στο αρχείο των κόμβων. Αυτό που κάνει και στις δύο περιπτώσεις είναι να επιστρέφει και να καταχωρείται στην μεταβλητή positionNowcbfNode την θέση, τον αριθμό που αντιστοιχείται στο τρίγραμμα μέσα στο αρχείο των κόμβων.

Αφού διαβάσουμε το πρώτο τρίγραμμα συνεχίζουμε διαβάζοντας το δεύτερο τρίγραμμα. Αυτό γίνεται ολισθαίνοντας κατά μία θέση τους χαρακτήρες του τριγράμματος που έχουμε και προσθέτοντας έναν ακόμη χαρακτήρα που διαβάζουμε από το αρχείο των tweets. Τώρα έχουμε το δεύτερο τρίγραμμα. Ελέγχουμε πάλι μέσω της μεθόδου compare() αν υπάρχει ή όχι μέσα στο αρχείο των κόμβων για να προστεθεί καθώς και να μας δώσει τον αριθμό της θέσεις που χαρακτηρίζει το τρίγραμμα.

Το δεύτερο και το τρίτο τρίγραμμα βρίσκονται κοντά, βρίσκονται μέσα στο παράθυρο των τριών τριγραμμάτων που έχουμε επιλέξει για να σχηματίζουμε ακμές αναμεταξύ τους. Οπότε ανοίγουμε το αρχείο των κόμβων για να γράψουμε αυτή την ακμή. Η ακμή καταγράφεται στο αρχείο ως ο αριθμός που αντιστοιχεί στον πρώτο κόμβο, κενό, ο αριθμός που αντιστοιχεί στον δεύτερο κόμβο. Παράλληλες ακμές και self loops επιτρέπονται.

Το πρόγραμμα συνεχίζει με το να σχηματίσει το τρίτο τρίγραμμα πάλι με την ίδια λογική ολισθαίνει κατά ένα χαρακτήρα το ήδη υπάρχον τρίγραμμα και προσθέτει ένα ακόμη χαρακτήρα. Ελέγχει αν υπάρχει ή όχι στο αρχείο των κόμβων για να το καταγράψει. Βρίσκει τον αριθμό του κόμβου του και σχηματίζει δύο ακόμα ακμές του τρίτου τριγράμματος με τα δύο προηγούμενα. Αυτές οι δύο ακμές καταγράφονται στο αρχείο των ακμών.

Στο πρόγραμμα μας έχουμε επιλέξει ένα παράθυρο τριών τριγραμμάτων. Όταν διαβάσαμε το πρώτο τρίγραμμα δεν το συνδέσαμε με κάποια ακμή εκείνη την στιγμή. Όταν διαβάσαμε το δεύτερο τρίγραμμα το συνδέσαμε μόνο με το πρώτο. Όταν διαβάσαμε το τρίτο το συνδέσαμε με το δεύτερο και το πρώτο. Όσα θα διαβάσουμε από το Τέταρτο και μετά θα τα συνδέουμε ακριβώς με τα τρία προηγούμενα. Για κάθε τρίγραμμα που θα σχηματίζουμε θα σχηματίζουμε ταυτόχρονα τρεις ακμές με τα τρία προηγούμενα του τριγράμματα. Αυτό πραγματοποιείται με το ότι μπαίνουμε σε μια επαναληπτική διαδικασία η οποία διαβάζει χαρακτήρες μέχρι το τέλος του κάθε tweet

Ο επαναληπτικός βρόχος που είναι ο κύριος πυρήνας της μεθόδου. Ξεκινά με το να μετακινεί το παράθυρο των τριγραμμάτων κατά μία θέση. Έπειτα σχηματίζει το καινούριο τρίγραμμα ολισθαίνοντας τους δύο από τους τρεις χαρακτήρες του κατά μία θέση και προσθέτοντας ως τρίτο χαρακτήρα τον επόμενο χαρακτήρα που διαβάζεται από το αρχείο των tweets. Στην συνέχεια αναζητείται αυτό το καινούριο τρίγραμμα στο αρχείο των κόμβων και προστίθεται αν δεν υπάρχει. Στην μεταβλητή positionNowcbfNode καταχωρούμε την θέση που έχει αυτό το τρίγραμμα στο αρχείο των κόμβων.

Στην μεταβλητή positionNowcbfNode έχουμε την θέση του τρέχοντος τριγράμματος, στις μεταβλητές position1stNode, position2ndNode, position3rdNode έχουμε τις θέσεις των προηγούμενων τριών τριγραμμάτων. Με αυτά τα δεδομένα σχηματίζουμε τρεις ακμές. Κάθε ακμή θα έχει στο ένα άκρο της τον τρέχον κόμβο και στο άλλο άκρο της ένα από τους τρεις προηγούμενους κόμβους. Τις τρεις αυτές ακμές τις καταγράφουμε στο αρχείο των ακμών. Επιτρέπεται να έχουμε καταγράψει πολλές φορές την ίδια ακμή γιατί επιτρέπονται οι παράλληλες ακμές. Αυτό που δεν επιτρέπεται είναι να καταγράψουμε ένα τρίγραμμα περισσότερες από μία φορές στο αρχείο των κόμβων. Κάθε τρίγραμμα θα υπάρχει μια και μοναδική φορά

Ο επαναληπτικός βρόχος θα τελειώσει όταν βρεθούμε στο τέλος του tweet. Ο τελευταίος χαρακτήρας του tweet θα σημάνει ότι βγαίνουμε από τον βρόχο που διαβάζουμε χαρακτήρες για αυτό το tweet. Περιμένουμε ως πολύ πιθανόν να υπάρχουν και άλλα tweets για αυτό υπάρχει ένας ακόμη επαναληπτικός βρόχος εξωτερικός ο οποίος επαναλαμβάνει την προηγούμενη διαδικασία για κάθε tweet και τελειώνει μόνο όταν διαβάσουμε τον τελευταίο χαρακτήρα από το τελευταίο tweet.

Είναι σημαντικό να διασαφηνίσουμε ότι για κάθε tweet που υπάρχει στο αρχείο των tweets, τα διαβάζουμε, σχηματίζουμε κόμβους - τριγράμματα και ακμές με την ίδια διαδικασία σαν να είναι μοναδικά και όχι σαν να είναι ένα συνεχόμενο κείμενο. Δηλαδή για κάθε tweet που έχουμε διαβάζουμε πρώτα το πρώτο τρίγραμμα έπειτα το δεύτερο και σχηματίζουμε την πρώτη ακμή έπειτα το τρίτο τρίγραμμα και σχηματίζουμε δύο ακμές και έπειτα όλα τα άλλα τριγράμματα που για το καθένα σχηματίζουμε τρεις ακμές με τα τρία προηγούμενα του τριγράματα. Δίνουμε προσοχή στο ότι δεν σχηματίζουμε ακμές μεταξύ του πρώτου τριγράμματος ενός tweet με τα τρία τελευταία τριγράμματα του προηγούμενου tweet. Κάθε tweet το επεξεργαζόμαστε σαν αυτοτελής ομάδα αλλά τα δεδομένα από όλα τα tweet καταχωρούνται σε ένα κοινό αρχείο.

Καθ' όλη την διάρκεια εκτέλεσης της μεθόδου ενημερώνουμε και προβάλουμε στον χρήστη τα τριγράμματα και τις ακμές που σχηματίστηκαν.

Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
 public static void main_Input (String tweetFile, String nodesFile, String edgesFile )throws Exce
{

System.out.println("-----***The processing of file "+ tweetFile +" has begun.***-----");
try {
BufferedReader in = new BufferedReader (new FileReader (tweetFile)) ;
// It accepts the file. It makes 3-Grams / nodes, It finds which 3-Grams are inside the //frame to r
the edges.
int positionNowcbfNode, position1stNode, position2ndNode, position3rdNode;
char [] cbuf= new char[3];  // The 3-Gram / node.
char [] relocation = new char[1]; // The new char which will be added in our new //3-Gram
relocation[0]=' ';
initialize(tweetFile,nodesFile, edgesFile);

  do{
          /** It reads the 1st 3-Gram / node. **/
  if (relocation[0]=='\r'||relocation[0]=='\n'){in.read(relocation);}  //***SOS ***
  in.read(cbuf);  // It reads the first 3 chars, these chars will be my 1st 3-Gram / node.
  positionNowcbfNode=compare(cbuf,nodesFile); // It added the 1st 3-Gram in nodes.txt

  /** It reads the 2nd 3-Gram / node. **/
  in.read(relocation); // It reads only one char, this char will be the new char in my new //3-Gram
node.
  position1stNode=positionNowcbfNode;
  cbuf[0]=cbuf[1]; // It shifts two chars of the 3-Gram. It also add the last char I read.
  cbuf[1]=cbuf[2];
  cbuf[2]=relocation[0];
  positionNowcbfNode=compare(cbuf,nodesFile); // It added the 2nd 3-Gram in //nodes.txt

  try { // It open the file to write the Edges.
  File file = new File(edgesFile);
  // if file doesn't exists, then create it
  if (!file.exists()) {file.createNewFile();}

  FileWriter fw = new FileWriter(edgesFile,true);
  BufferedWriter bw = new BufferedWriter(fw);

  // It writes only ONE new edge in the file of Edges. It is written in the form number of //1st noc
```

```
number of 2nd node.
  bw.write(Integer.toString(positionNowcbfNode)+" "+Integer.toString(position1stNode) + "\n")
  System.out.print("The ");
  System.out.print(++numberOfEdges);
  System.out.print(" edge which adjoins the nodes ");
  System.out.print(Integer.toString(positionNowcbfNode)+" - "+Integer.toString(position1stNod
" was added.\n");

  bw.close();  // It closes the file of Edges.
  } catch (IOException e) {e.printStackTrace();}

  /** It reads the 3rd 3-Gram / node. **/
  in.read(relocation); // It reads only one char, this char will be the new char in my new //3-Gram
node.
  position2ndNode=position1stNode; // position1stNode, position2ndNode are the positions of
previous nodes. They are needed to make new Edges.
  position1stNode=positionNowcbfNode;
  cbuf[0]=cbuf[1]; // It shifts two chars of the 3-Gram. It also add the last char it reads.
  cbuf[1]=cbuf[2];
  cbuf[2]=relocation[0];
  positionNowcbfNode=compare(cbuf,nodesFile);// positionNowcbfNode has the //number of th
node of the current 3-Gram is read. I is useful to make the new Edges.

  try{ // It open the file to write the Edges.
  File file = new File(edgesFile);
  // if file doesn't exists, then create it
  if (!file.exists()) {file.createNewFile();}

  FileWriter fw = new FileWriter(edgesFile,true);
  BufferedWriter bw = new BufferedWriter(fw);

  // It writes TWO new edges in the file of Edges. They are written in the form number of //1st n
number of 2nd node.
  bw.write(Integer.toString(positionNowcbfNode)+" "+Integer.toString(position2ndNode) + "\n"
  System.out.print("The ");
  System.out.print(++numberOfEdges);
  System.out.print(" edge which adjoins the nodes ");
  System.out.print(Integer.toString(positionNowcbfNode)+" - "+Integer.toString(position2ndNo
" was added.\n");

  bw.write(Integer.toString(positionNowcbfNode)+" "+Integer.toString(position1stNode) + "\n")
  System.out.print("The ");
  System.out.print(++numberOfEdges);
  System.out.print(" edge which adjoins the nodes ");
```

```
  System.out.print(Integer.toString(positionNowcbfNode)+" - "+Integer.toString(position1stNod
" was added.\n");

  bw.close(); // It closes the file of Edges.
  } catch (IOException e) {e.printStackTrace();}

  /** It reads all the 3-Grams / nodes from 3rd onwards. **/
    do{
    position3rdNode=position2ndNode; // //position3rdNode,position2ndNode,position1stNode a
the positions of the three //previous 3Grams / nodes
    position2ndNode=position1stNode; // They are needed to make three new edges. //There are
three Edges cause the frame includes three 3-Grams / Nodes
    position1stNode=positionNowcbfNode;
    if(in.read(relocation)==-1)break; // relocation[0] gets the new char which is read.
      if(relocation[0]=='\r'||relocation[0]=='\n'){
      System.out.println("\n\n\n-----The processing of a new tweet from "+tweetFile + " will
start-----");
      break;}

    cbuf[0]=cbuf[1]; // It makes the shift to have  new 3-Gram / node.
    cbuf[1]=cbuf[2];
    cbuf[2]=relocation[0];
    positionNowcbfNode=compare(cbuf,nodesFile); // positionNowcbfNode keeps the position o
the new node. It is useful to have the new edges.

    try{
    File file = new File(edgesFile);
    // If file doesn't exists, then create it.
    if (!file.exists()) {file.createNewFile();}

    FileWriter fw = new FileWriter(edgesFile,true);
    BufferedWriter bw = new BufferedWriter(fw);

    // 3 edges are added because we use a 3 grams window.
    bw.write(Integer.toString(positionNowcbfNode)+" "+Integer.toString(position3rdNode) + "\n
1st edge
    System.out.print("The ");
    System.out.print(++numberOfEdges);
    System.out.print(" edge which adjoins the nodes ");
    System.out.print(Integer.toString(positionNowcbfNode)+" -
"+Integer.toString(position3rdNode)+" was added.\n");

    bw.write(Integer.toString(positionNowcbfNode)+" "+Integer.toString(position2ndNode) + "\r
2nd edge
```

```
   System.out.print("The ");
   System.out.print(++numberOfEdges);
   System.out.print(" edge which adjoins the nodes ");
   System.out.print(Integer.toString(positionNowcbfNode)+" - "+Integer.toString(position2ndN
+" was added.\n");

   bw.write(Integer.toString(positionNowcbfNode)+" "+Integer.toString(position1stNode) + "\n
3rd edge
   System.out.print("The ");
   System.out.print(++numberOfEdges);
   System.out.print(" edge which adjoins the nodes ");
   System.out.print(Integer.toString(positionNowcbfNode)+" - "+Integer.toString(position1stNo
+ " was added.\n");

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

   }while(true);
  if(relocation[0]!='\r'){break;} // This is the end and there is no more tweets so it goes //out.
}while(true); // It is always true cause I want to read many tweets from the same file.
in.close(); // It closes the input file, the file from which  the tweets are read.
}
catch( IOException e ) {e . printStackTrace () ;}

System.out.print("Through "+tweetFile + " ");
System.out.print(numberOfEdges+1); // It adds one more cause It also exists the zero //node.
System.out.print(" edges and ");
System.out.print(numberOfNodes);
System.out.println(" nodes were made.\n\n");

String numbers=Integer.toString(numberOfNodes) + "\n"+Integer.toString(numberOfEdges+1) +
"\n";
prepend(edgesFile,numbers ); // It adds in the beginning of the file of edges the total //number of
nodes and edges.

}
```

8.2.2 public static int compare (char cbuf[], String nodesFile)

Η μέθοδος compare την οποία είδαμε να χρησιμοποιούμε αρκετές φορές στην main_Input

έχει σκοπό να γράφει στο αρχείο των κόμβων ένα τρίγραμμα στην περίπτωση που αυτό το τρίγραμμα δεν υπάρχει ήδη και επιστρέφει την θέση όπου βρίσκεται το τρίγραμμα μέσα στο αρχείο.

Ως είσοδο δέχεται ένα τρίγραμμα και το όνομα του αρχείου των κόμβων όπου θα γραφτεί αν δεν υπάρχει το τρίγραμμα.

Υπάρχουν οι εξής μεταβλητές η position που είναι ένας ακέρεος δηλώνει την θέση του τρέχοντος τριγράμματος που διαβάζουμε. Η existedNode που είναι ένας πίνακας τριών χαρακτήρων δηλώνει το ποιο είναι αυτό το τρέχον τρίγραμμα και η found πού είναι μια μεταβλητή boolean δηλώνει αν βρέθηκε ή όχι το τρίγραμμα που ψάχνουμε μέσα στο αρχείο.

Στην αρχή ανοίγει το αρχείο όπου θα γραφτούν οι κόμβοι. Έπειτα διατρέχουμε το αρχείο κατά τρεις χαρακτήρες την φορά που αντιστοιχούν σε ένα τρίγραμμα και συγκρίνουμε το τρέχον τρίγραμμα με αυτό που λάβαμε στην είσοδο. Ταυτόχρονα αυξάνουμε την μεταβλητή position κατά ένα την φορά για να ξέρουμε σε ποιο τρίγραμμα είμαστε. Συγκρίνουμε κάθε τρίγραμμα που διαβάσαμε από το αρχείο με το τρίγραμμα που δεχθήκαμε ως είσοδο αν είναι ίδια τότε βρέθηκε το τρίγραμμα και απλά επιστρέφουμε την θέση του που την υποδηλώνει η μεταβλητή position.

Αν δεν βρεθεί Ανοίγουμε ξανά το αρχείο των κόμβων αλλά αυτή την φορά για εγγραφή και γράφουμε το τρίγραμμα που δεν υπάρχει ήδη. Στο τέλος επιστρέφει ως θέση του καινούριου τριγράμματος το πλήθος των τριγραμμάτων που υπήρχαν από πριν και τα μετρήσαμε όλα μέσω της position συν ένα για αυτό που γράψαμε τώρα.

Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
public static int compare ( char cbuf[], String nodesFile  ) {

int position=-1; // The position of the 3-Gram / node in the file of nodes.
char [] existedNode= new char[3];

try {
BufferedReader in = new BufferedReader (new FileReader (nodesFile)) ;
boolean found=true; // found declares if exists the 3-Gram / node which I look for or no.
  do{
  position++; // It increases position by one for any 3-Gram / node I read.
  if (in.read(existedNode)==-1){ found =false; break;} // It reada the 3-grams / nodes. If it //is in
end it goes out of the loop and turns found to false.
    if(Arrays.equals(cbuf, existedNode)){ // If it found the 3-Gram / node it goes out of the //loop
keeps found as true.
    break;}
  }while(true);
in . close () ; // It closes the file of nodes.

  if (!found){ // It has not found the 3-Gram / node in any position so It adds it in the end //of the
```

```
edges file.
  try {
  File file = new File(nodesFile);
  // if file doesn't exists, then create it
  if (!file.exists()) {file.createNewFile();}

  FileWriter fw = new FileWriter(nodesFile,true);
  BufferedWriter bw = new BufferedWriter(fw);
  bw.write(cbuf);
  numberOfNodes++; // numberOfNodes has already the number of the last 3-Gram / //node. Cau
It adds one more node, it increases numberOfNodes by one.
  bw.close();

  System.out.println("--->The 3-Gram |"+ new String(cbuf) + "| was added in the position:
"+(numberOfNodes-1));
  } catch (IOException e) {e.printStackTrace();}
  }
}
catch( IOException e ) {e . printStackTrace () ;}

return position;

}
```

8.2.3 public static void prepend(String fileName,String data)

Η μέθοδος prepend που την χρησιμοποιήσαμε στην main_Input έχει σκοπό να βάλει στην αρχή του αρχείου των ακμών το πλήθος των κόμβων και ακμών που έχουμε.

Ως είσοδο η μέθοδος δέχεται το όνομα του αρχείου που έχει τις ακμές και ένα String που περιλαμβάνει το πλήθος των κόμβων, μια αλλαγή γραμμής και το πλήθος των ακμών.

Αρχικά ανοίγουμε το αρχείο των ακμών. Έπειτα σε ένα StringBuilder βάζουμε το πλήθος των κόμβων και ακμών που λάβαμε ως είσοδο. Στην συνέχεια διαβάζουμε όλα τα περιεχόμενα γραμμή προς γραμμή από το αρχείο και εκτός από τις δύο πρώτες γραμμές τα επισυνάπτουμε στο τέλος του StringBuilder. Το StringBuilder το καταχωρούμε σε ένα String και το String αυτό το σώζουμε εκ νέου στο αρχείο των ακμών.

Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
public static void prepend(String fileName,String data) throws IOException{

BufferedReader in = new BufferedReader(new FileReader(fileName));
StringBuilder reply = new StringBuilder();
reply.append(data); // It adds the nodes and edges in the StringBuilder reply.
String str = new String();

while( (str = in.readLine())!=null) // It reads one by one the lines from Edges.txt
reply.append(str+"\n"); // It appends one by one the lines in the StringBuilder reply.
str = reply.toString();
BufferedWriter out= new BufferedWriter(new FileWriter(fileName));
out.write(str); // It writes in Edges.txt the number of nodes, the number of edges and all //the edg

if(in !=null){
try{ in.close();} catch(IOException e){e.printStackTrace();}
if(out!=null)
try{out.close();} catch(IOException ex) {ex.printStackTrace();}
}

}
```

8.2.4 public static void initialize(String tweetFile, String nodesFile, String edgesFile)

Η μέθοδος initialize έχει σκοπό να κάνει κάποιες αρχικές τροποποιήσεις και ελέγχους στα αρχεία που διαχειρίζεται το πρόγραμμα μας.

Αρχικά ελέγχει αν στο αρχείο με τα tweets επαρκούν οι χαρακτήρες για να σχηματιστεί ο ελάχιστος αριθμός κόμβων και ακμών. Έπειτα ακόμη και αν υπάρχουν τα αρχεία των ακμών και των κόμβων από προηγούμενες χρήσεις του προγράμματος τα διαγράφει και τα επαναδημιουργεί από την αρχή.

Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
public static void initialize(String tweetFile, String nodesFile, String edgesFile) throws IOExcep

int numberOfChars=0;
```

```
numberOfEdges=-1;
numberOfNodes=0;
try {
do{ // It checks if tweet.txt has at least five chars.
BufferedReader in = new BufferedReader (new FileReader (tweetFile)) ;
do{numberOfChars++;
}while(in.read()!=-1);
if (numberOfChars>5) break;
System.out.println("The file "+ tweetFile +" has less than 5 chars replace it with a bigger and cli
enter ");
System.in.read();
in.close();
}while(true);
}
catch( IOException e ) {
e . printStackTrace () ;
}
// It deletes the old file of nodes and makes a new one.
String path1 = (nodesFile);
File fileDelNodes = new File(path1);
fileDelNodes.delete();
File f1 = new File(path1);
f1.createNewFile();

// It delete the old file of edges and makes a new one.
String path2 = (edgesFile);
File fileDelEdges = new File(path2);
fileDelEdges.delete();
File f2 = new File(path2);
f2.createNewFile();
}

}
```

8.3 Compare2Graphs

Η κλάση compare2Graphs έχει σκοπό να δεχθεί δύο γράφους να τους συγκρίνει και να δώσει ως αποτέλεσμα έναν αριθμό που βρίσκετε ανάμεσα στο μηδέν και το ένα και που θα εκφράζει το κατά πόσο αυτοί οι δύο γράφοι είναι όμοιοι. Το μηδέν θα υποδηλώνει ότι οι δύο

γράφοι δεν μοιάζουν καθόλου και το ένα ότι οι δύο γράφοι μοιάζουν πολύ. Μπορούμε σε κάποιες περιπτώσεις να έχουμε το ένα ακόμη και όταν οι γράφοι δεν ταυτίζονται. Παράδειγμα αυτής της περίπτωσης είναι αν ο ένας είναι υπογράφος του άλλου.

Κατανοούμε την σημασία του να έχουμε μια καλή μέθοδο για να βρίσκει την ομοιότητα μεταξύ δύο γράφων γιατί η ομοιότητα των δύο γράφων συνεπάγεται έμμεσα και την ομοιότητα μεταξύ των tweets των χρηστών. Αναμένουμε είναι ότι υψηλή ομοιότητα μεταξύ δύο γράφων θα σημαίνει και υψηλή ομοιότητα μεταξύ των χρηστών και αυτό θα συνεπάγεται ότι οι χρήστες θα έχουν κοινά ενδιαφέροντα, κοινές ασχολίες και κοινές δραστηριότητες.

Η αναγνώριση και βαθμονόμηση αυτών των κοινών στοιχείων μας κάνει να πιστεύουμε στην αποτελεσματικότητα και ισχύ του recommendation συστήματος που κάνει χρήση τριγραμμάτων.

Χρησιμοποιούμε δύο μεθόδους βαθμονόμησης της ομοιότητας δύο γράφων την Containment Similarity και την Value Similarity. Η Containment Similarity συγκρίνει τους απλούς γράφους που παράγαμε από τα αρχεία των tweets ενώ η Value Similarity συγκρίνει γράφους που έχουν βάρη στις ακμές τους. Παρακάτω θα δούμε αναλυτικά την προγραμματιστική τους υλοποίηση και θα περιγράψουμε τις ιδιαιτερότητες της κάθε μεθόδου.

8.3.1 static public float valueSimilarity(String G1, String G2, String SimpleG1, String SimpleG2, String FileNameNodes1, String FileNameNodes2)

Η μέθοδος valueSimilarity υποδηλώνει πόσες από της ακμές που υπάρχουν στον πρώτο γράφο υπάρχουν και στον δεύτερο γράφο υπολογίζοντας και το βάρος που έχει κάθε ακμή. Οπότε δεν μας ενδιαφέρει μόνο αν ένα τρίγραμμα ακολουθείτε από ένα άλλο μέσα στα tweets αλλά και το πόσες φορές συναντάμε αυτό το τρίγραμμα να ακολουθείτε από ένα άλλο. Αυτός ο αριθμός που εκφράζει το πόσο συχνά ένα τρίγραμμα ακολουθεί ένα άλλο τον αποκαλούμε βάρος της ακμής και αυτά τα βάρη τα συγκρίνουμε. Όταν οι αριθμοί των βαρών της ίδιας ακμής είναι κοντά η μέθοδος θα το εκφράζει ως το ότι υπάρχει μεγαλύτερη ομοιότητα από ότι αν οι αριθμοί διαφέρουν κατά πολύ. Έχουμε δύο γράφους τον G_{t_i} και τον G_{T_P}

Αν μια κοινή ακμή έχει βάρος στον πρώτο γράφο $w^{t_i}(e)$ και στον άλλο γράφο $w^{T_P}(e)$ αντίστοιχά τότε η συμμετοχή της στο συνολικό άθροισμα της ομοιότητας θα είναι

$$VR(e) = \frac{min(w^{t_i}(e), w^{T_P}(e))}{max(w^{t_i}(e), w^{T_P}(e))}$$

Βλέπουμε ότι αν τα βάρη των ακμών συμπίπτουν το αποτέλεσμα θα είναι ένα ενώ αν μια ακμή υπάρχει στον πρώτο γράφο και δεν υπάρχει στον δεύτερο το αποτέλεσμα θα είναι μηδέν. Έπειτα αθροίζουμε όλους τους παραπάνω αριθμούς και τους διαιρούμε δια το μέγιστο πλήθος ακμών που έχει ο κάθε γράφος. Η τελική μας εξίσωση είναι η

$$VS(G_{t_i}, G_{T_P}) = \frac{\sum_{e \in G_{t_i}} \frac{min(w^t i(e), w^T P(e))}{max(w^t i(e), w^T P(e))}}{max(|G_{t_i}|, |G_{t_P}|)}$$

Ξανά επιστρέφουμε πίσω στην προγραμματιστική υλοποίηση της μεθόδου αυτής. Βλέπουμε ότι ως είσοδο δεχόμαστε έξι Strings αυτά είναι από ένα String που περιγράφει τον κάθε πλήρη γράφο με βάρη, από ένα String που περιγράφει τον κάθε απλό γράφο και τα ονόματα των δύο αρχείων που έχουν ποια τριγράμματα αντιστοιχούν σε κάθε κόμβο.

Στις μεταβλητές numberOfEdges1, numberOfEdges2 καταχωρούμε το πλήθος των ακμών που έχει κάθε γράφος. Χρειαζόμαστε να ξέρουμε ποιο είναι το μέγιστο πλήθος ακμών γιατί με αυτό θα διαιρέσουμε όπως είδαμε παραπάνω το άθροισμα των συνεισφορών κάθε ακμής. Η κλίση της μεθόδου numberOfEdges() είναι αυτή που θα υπολογίσει το πλήθος ακμών του κάθε γράφου και στην μεταβλητή maxNumberOfEdges θα έχουμε το μέγιστο πλήθος των ακμών.

Στην συνέχεια ακολουθεί ένας επαναληπτικός βρόχος όπου επεξεργαζόμαστε τον γράφο ως ένα String. Κάθε κόμβος αντιστοιχεί με μια γραμμή του String. Στην αρχή της γραμμής έχουμε τον τρέχον κόμβο και έπειτα όλους τους άλλους κόμβους με τους οποίους συνδέεται ο πρώτος κόμβος μέσω μιας ακμής.

Στον επαναληπτικό βρόχο χωρίζουμε το String του πρώτου γράφου σε γραμμές και διατρέχουμε μια μια την κάθε γραμμή. Για κάθε ακμή που σχηματίζεται μεταξύ του πρώτου κόμβου με κάποιον από τους επόμενους κάνουμε τα εξής. Καλούμε την μέθοδο weightOfEdge που μας αναφέρει το βάρος αυτής της ακμής στον πρώτο γράφο και έπειτα βρίσκουμε τα τριγράμματα που αντιστοιχούν σε αυτούς τους κόμβους για να αναζητούμε μέσω της μεθόδου weihgtOfEdgeGraph2 αν αυτή η ακμή με τα ίδια τριγράμματα υπάρχει στον δεύτερο γράφο.

Τα τριγράμματα τα βρίσκουμε καλώντας την μέθοδο nodes3Gram. Η μέθοδος nodes3Gram θα δεχθεί έναν πίνακα με δύο ακεραίους και το όνομα απο ένα αρχείο. Με τους δύο ακεραίους θα βρεί ποια τριγράμματα τους αντιστοιχούν για να τα αποδίδει στις μεταβλητές Node1_1 και Node1_2.

Η μέθοδος weihgtOfEdgeGraph2 που καλούμε στην συνέχεια και δέχεται ως είσοδο τα Node1_1 και Node1_2 κάνει κάτι περισσότερο από το να αποφαίνεται αν υπάρχει ή οχι μια ακμή. Αν δεν υπάρχει επιστρέφει τον αριθμό -1 αν όμως υπάρχει μας επιστρέφει το βάρος που έχει αυτή η ακμή στον δεύτερο γράφο.

Τα δύο βάρη των ακμών από τον πρώτο και από τον δεύτερο γράφο τα αποθηκεύουμε σε

δύο μεταβλητές την weight1 και την weight2 αυτά τα συγκρίνουμε και υπολογίζουμε το ελάχιστο των δύο δια το μέγιστο και τα αθροίζουμε με το συνολικό άθροισμα της μεταβλητής sum

Η μεταβλητή sum αθροίζει για όλες τις ακμές του πρώτου γράφου την συμβολή του, έναν αριθμό μεταξύ του μηδέν και του ένα που υποδηλώνει κατά πόσο έχουν παρόμοια βαρύτητα σύμφωνα με την εξίσωση VR(e) που αναφέραμε σε προηγούμενη παράγραφο. Τέλος διαιρούμε το sum με το μέγιστο αριθμό ακμών μεταξύ των δύο γράφων και το παρουσιάζουμε στον χρήστη.

Καθ' όλη την διάρκεια της μεθόδου ενημερώνουμε τον χρήστη για το ποια ακμή από τον πρώτο γράφο εξετάζουμε, ποια τριγράμματα της αντιστοιχούν, τι βάρος έχει, αν βρέθηκε αυτή η ακμή στον δεύτερο γράφο και αν ναι τι βάρος έχει στον δεύτερο γράφο.

Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
static public float valueSimilarity(String G1, String G2, String SimpleG1, String SimpleG2, Stri
FileNameNodes1, String FileNameNodes2) throws IOException{
String line,Graph1;
int
node1st=0,anotherNode=0,numberOfLine=-1,numberOfEdges1,numberOfEdges2,maxNumberO
dges;
int max, min;
float sum=0;
int [] edge= new int[2];
char [] Node1_1= new char[3]; // Array of chars. They contain the chars of each 3-Gram node.
char [] Node1_2= new char[3];
numberOfEdges1=numberOfEdges(SimpleG1); // The number of edges which belong to simple
Graph.
numberOfEdges2=numberOfEdges(SimpleG2);
System.out.println("-----***The comparison of two Graphs has begun.***-----");
if(numberOfEdges1>numberOfEdges2){maxNumberOfEdges=numberOfEdges1;}else{maxNu
OfEdges=numberOfEdges2;} // It takes the maximum number of nodes.

System.out.print("[1st Graph] The number of Edges is: "); System.out.println(numberOfEdges1)
System.out.print("[2nd Graph] The number of Edges is: "); System.out.println(numberOfEdges2

// It separates the string of simple graph into lines. Each line represent a node of the Graph.
Graph1=SimpleG1.replace(": ", " ");
Pattern COMMA = Pattern.compile("\n");
Pattern COMMA2=Pattern.compile(" ");
  for (String token : COMMA.split(Graph1)) {
  line=token.replace("\r", "");numberOfLine++;
    if(numberOfLine>=1){boolean is1stNode=true,                    is1stNodeAgain=false;;
      // It separates the line of a node into numbers. Each number represent another node. These tv
```

```
nodes make an edge.
     for (String node : COMMA2.split(line)) { // Each edge has the 1st node which is the same fo
the edges of the line and another node.
     is1stNodeAgain=true;
       if(is1stNode){ // It checks if this node is the 1st node of the edge.
       node1st=Integer.parseInt(node);
       is1stNode=false;is1stNodeAgain=false;}
       else{anotherNode=Integer.parseInt(node);}

       if(is1stNodeAgain){
         if(node1st<=anotherNode){ // If node1st>=anotherNode then It has already checked this
edge.
         edge[0]=node1st;
         edge[1]=anotherNode;
         nodes3Gram(FileNameNodes1,edge,Node1_1,Node1_2); // It gets the 3-Grams from the
number of node.
         System.out.print("[1st Graph] The Edge ");System.out.print(node1st);System.out.print(" -
System.out.print(anotherNode);
         System.out.print("  with 3-Grams |"+new String(Node1_1) +"| - |"+new String(Node1_2)+
");
         System.out.print("has weight: ");System.out.println(weightOfEdge(node1st,anotherNode,
         int weight1=weightOfEdge(node1st,anotherNode,G1); // weight1 has the weight of the ed
Graph1.
         int weight2=weihgtOfEdgeGraph2(G2, SimpleG2, FileNameNodes2, Node1_1, Node1_2
weight2 has the weight of the edge in Graph2.
           if(weight1>=weight2){max=weight1; min=weight2;} // It compares the weights to take
min(weight1,weight2)/max(weight1,weight2)
           else{max=weight2;min=weight1;};
         sum=sum+(float)min/(float)max; // Sum represents the similarity of all the edges.
         }
       }
     }
   }
  }

System.out.print("\nThe Value Similarity is: ");
return sum/maxNumberOfEdges;}
```

8.3.2 private static int numberOfEdges(String graph)

Η μέθοδος numberOfEdges είναι μια βοηθητική μέθοδος που έχει ως σκοπό να δεχθεί έναν γράφο με την μορφή String και να επιστρέψει τον αριθμό των ακμών που έχει. Αυτό το επιτυγχάνει με το να απομονώνει μέσω της κλήσης της substring μόνο την ακολουθία χαρακτήρων που υποδηλώνει το πλήθος των ακμών που έχει ο γράφος. Τέλος μετατρέπει αυτή την ακολουθία χαρακτήρων σε ακέραιο και την επιστρέφει.

Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
private static int numberOfEdges(String graph) {
graph=graph.substring(graph.indexOf(",") + 2);
graph=graph.substring(0, graph.indexOf("d")-2);
return Integer.parseInt(graph);
}
```

8.3.3 static private int weihgtOfEdgeGraph2(String G2, String SimpleG2, String FileNameNodes2,char[] Node1_1, char[] Node1_2)

Η μέθοδος weihgtOfEdgeGraph2 δέχεται δύο τριγράμματα βρίσκει αν σχηματίζουν ακμή και επιστρέφει το βάρος αυτής της ακμής. Για να το επιτύχουμε αυτό θα χρειαστούμε εκτός από τα τριγράμματα το αρχείο των κόμβων όπου θα γίνει η αντιστοίχηση των τριγραμμάτων με τον αριθμό που αντιστοιχεί στην θέση του τριγράμματος. Επίσης θα χρειαστούμε τον γράφο στην απλή του μορφή και στην μορφή του με βάρη.

Αρχικά η μέθοδος ξεκινά με το να καταχωρούμε στις μεταβλητές node1 και node2 τους αριθμούς των δύο κόμβων. Αυτό το πετυχαίνουμε με την κλήση της μεθόδου findNumberOfNode. Αν δεν υπάρχει αυτό το τρίγραμμα τότε στην μεταβλητή θα καταχωρηθεί ο αριθμός -1 και θα ενημερώσουμε τον χρήστη ότι δεν σχηματίζεται καμία ακμή στον γράφο με αυτά τα τριγράμματα.

Αν βρεθούν οι δύο αριθμοί που αντιστοιχούν σε αυτά τα τριγράμματα και είναι διάφοροι του -1 τότε καλούμε την μέθοδο weightOfEdge όπου υπολογίζει το βάρος αυτής της ακμής.

Τέλος στον χρήστη παρουσιάζουμε την ακμή με τα τριγράμματα που αντιστοιχούν στους κόμβους της, τους αριθμούς , και το βάρος της. Αυτό το βάρος το επιστρέφουμε ως αποτέλεσμα μέσω της return στο σημείο από όπου κλήθηκε η μέθοδος.

Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
static private int weihgtOfEdgeGraph2(String G2, String SimpleG2, String FileNameNodes2,ch
Node1_1, char[] Node1_2) throws IOException{
int node1=findNumberOfNode(FileNameNodes2,  Node1_1 ); // It takes a 3-Gram node and
returns the number of this node.
int node2=findNumberOfNode(FileNameNodes2,  Node1_2 ); // If the method returns -1 then th
3-Gram node doesn't exist.
int result=0;
System.out.print("[2nd Graph] The Edge |"); System.out.print(Node1_1);System.out.print("| -
|");System.out.print(Node1_2);
  if (node1>-1 && node2>-1){ // It checks that both nodes exist.
  System.out.print("| exists as "); System.out.print(node1);System.out.print(" -
");System.out.print(node2); System.out.print(" and has weight: ");
  result=weightOfEdge(node1, node2, G2);
  System.out.println(result);
  }else System.out.println("| DOES NOT exist in the 2nd Graph.");
return result;
}
```

8.3.4 static public int weightOfEdge(int node1st, int node2nd, String graph)

Η μέθοδος weightOfEdge δέχεται ως είσοδο τον πλήρη γράφο και δύο αριθμούς που αντιστοιχούν σε δύο κόμβους και επιστρέφει το βάρος της ακμής που σχηματίζουν αυτοί οι δύο κόμβοι.

Η μέθοδος weightOfEdge διαφοροποιείται από την weihgtOfEdgeGraph2 στο ότι βρίσκει το βάρος της ακμής με βάση τους αριθμούς που αντιστοιχούν στην θέση του κόμβου και όχι στα αντίστοιχα τριγράμματα. Στην πραγματικότητα η weihgtOfEdgeGraph2 χρησιμοποιεί μέσα της την weightOfEdge ως μια μέθοδο που την καλεί.

Ο γράφος έχει δοθεί ως ένα String όπου η κάθε γραμμή του αντιστοιχεί με έναν κόμβο του. Γνωρίζουμε τον αριθμό του πρώτου κόμβου οπότε ανατρέχουμε στην αντίστοιχη γραμμή του String του γράφου. Εκεί διαβάζουμε στην συνέχεια της γραμμής να δούμε πόσες φορές υπάρχει ο δεύτερος κόμβος. Το βάρος της ακμής θα είναι ίσο με το πόσες φορές υπάρχει αυτός ο κόμβος.

Υπάρχει μια περίπτωση όταν η ακμή είναι self loop δηλαδή οι δύο κόμβοι της ακμής να

ταυτίζονται τότε θα τους έχουμε μετρήσει διπλάσιες φορές από όσες πράγματι υπάρχουν. Αυτό οφείλεται στον τρόπο που κάνουμε την αναπαράσταση του γράφου. Στον γράφο κάθε ακμή την καταγράφουμε δύο φορές. Μία στην γραμμή που αντιστοιχεί στην ακμή του πρώτου κόμβου γράφουμε τον αριθμό του δεύτερο κόμβο και μία στην γραμμή που αντιστοιχεί στον δεύτερο κόμβο γράφουμε τον αριθμό του πρώτου κόμβου. Αυτό γίνεται ούτως ώστε με μια απλή επισκόπηση σε κάθε κόμβο να ξέρουμε με ποιους άλλους κόμβους σχηματίζει ακμές. Για αυτόν τον λόγο όταν οι δύο κόμβοι ταυτίζονται σε μια ακμή στην αντίστοιχη γραμμή του κόμβου γράφονται δύο φορές οι κόμβοι. Αυτό το πρόβλημα το επιλύουμε με το να ελέγχουμε στο τέλος αν έχουμε self loop. Στην περίπτωση που έχουμε self loop διαιρούμε δια του δύο το βάρος της ακμής που έχουμε καταμετρήσει. Τέλος επιστρέφουμε ως αποτέλεσμα της μεθόδου το βάρος που μας έχει ζητηθεί.

Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
static public int weightOfEdge(int node1st, int node2nd, String graph){
String line;
int numberOfLine=-2,weight=0;
graph=graph.replace(": ", " ");
Pattern COMMA = Pattern.compile("\n");
Pattern COMMA2=Pattern.compile(" ");
Boolean secondTimeOrMore=false;
  for (String token : COMMA.split(graph)) { //It reads all the lines of the string - Graph.
  line=token.replace("\r", "");numberOfLine++;
    if(numberOfLine==node1st){ //It finds the right line - node.
      for (String node : COMMA2.split(line))
{if((node2nd==Integer.parseInt(node))&&(secondTimeOrMore)){weight++;}
      secondTimeOrMore=true;} // It counts how many times the node2nd exists in the line.
    }
  }
if(node1st==node2nd)weight=weight/2;
return weight;
}
```

8.3.5 static public void contSimilarity(String FileNameNodes1, String FileNameEdges1, String FileNameNodes2, String FileNameEdges2)

Ο δεύτερος αλγόριθμος που υλοποιήσαμε για την σύγκριση των δύο γράφων και να δούμε κατά πόσο είναι όμοιοι είναι ο containment similarity. Ο containment similarity κάνει χρήση του απλού γράφου. Διατρέχει όλες τις ακμές του πρώτου γράφου και αναζητά για κάθε μια αν υπάρχει και στον δεύτερο γράφο. Αν υπάρχει προσθέτει ένα στο γενικό άθροισμα αλλιώς δεν προσθέτει κάτι. Στο τέλος το γενικό άθροισμα διαιρείται με το ελάχιστο πλήθος ακμών μεταξύ των δύο γράφων. Η εξίσωση που μας δίνει το αποτέλεσμα είναι

$$CS(G_{t_i}, G_{T_p}) = \sum_{e \in G_{t_i}} \mu(e, G_{T_P}) / min(|G_{t_i}|, |G_{T_p}|)$$

Το θέμα είναι λίγο ποιο σύνθετο στην προγραμματιστική υλοποίηση μας γιατί δεν μας ενδιαφέρει να κάνουμε σύγκριση των αριθμών των κόμβων από τις οποίες αποτελείται ο γράφος αλλά να ανατρέξουμε πίσω να δούμε ποια τριγράμματα αντιστοιχούν σε ποιους αριθμούς κόμβων και τελικά να γίνει σύγκριση με βάση τα σωστά τριγράμματα που σχηματίζουν τις ακμές.

Η μέθοδος δέχεται το όνομα του αρχείου των κόμβων και το όνομα του αρχείου των ακμών και για τους δύο γράφους. Αρχικά ανοίγει τα αρχεία των ακμών για ανάγνωση. Στις μεταβλητές numberNodesFile1, numberEdgesFile1, numberNodesFile2, numberEdgesFile2 καταχωρεί το πλήθος των κόμβων και των ακμών αντίστοιχα και για τα δύο αρχεία. Αυτά τα δεδομένα τα διαβάζει από το αρχείο των ακμών που υπάρχουν στις δύο πρώτες γραμμές.

Σε έναν επαναληπτικό βρόχο διαβάζεται γραμμή προς γραμμή τα περιεχόμενα του αρχείου των ακμών. κάθε γραμμή του αρχείου αντιστοιχεί σε μια ακμή όπου ο πρώτος αριθμός θα είναι ο πρώτος κόμβος και ο δεύτερος αριθμός θα είναι ο δεύτερος κόμβος. Αυτούς τους δύο ακέραιους αριθμούς τους καταχωρούμε σε έναν πίνακα με το όνομα edges.

Έπειτα καλούμε την μέθοδο nodes3Gram όπου αναθέτει στις μεταβλητές Node1_1 και Node1_2 τα δύο τριγράμματα που αντιστοιχούν στους αριθμούς κόμβων της ακμής που διαβάσαμε πριν.

Στην συνέχεια η μέθοδος findNumberOfNode θα καταχωρίσει στις μεταβλητές numberOfNode2_1 και numberOfNode2_2 τους αριθμούς της θέσης των κόμβων όπου βρίσκονται τα τριγράμματα που διαβάσαμε πριν στο δεύτερο αρχείο κόμβων.

Πολύ σημαντικό ρόλο στην εξέλιξη της μεθόδου έχει ο λογικός έλεγχος που γίνεται μέσω της μεθόδου doWeHaveEdgefromNodes. Η μέθοδος doWeHaveEdgefromNodes ελέγχει αν υπάρχει ή όχι ακμή στον δεύτερο γράφο που να σχηματίζεται από τους αριθμούς των κόμβων numberOfNode2_1 και numberOfNode2_2. Αν υπάρχει τότε αυξάνουμε κατά ένα την αριθμιτική μεταβλιτή NumberOfCommonEdges.

Η μεταβλητή NumberOfCommonEdges έχει το πλήθος των ακμών που βρέθηκαν και

στους δύο γράφους. Στο τέλος διαιρούμε τον αριθμό αυτόν με τον ελάχιστο αριθμό κλάδων που έχουν οι δύο γράφοι. Το αποτέλεσμα είναι ο αριθμός ομοιότητας μεταξύ των δύο γράφων. Ο αριθμός αυτός κυμαίνεται μεταξύ του μηδέν και του ένα. Με το μηδέν να δηλώνει ότι οι οι γράφοι δεν έχουν τίποτα κοινό και το ένα ότι οι γράφοι ταυτίζονται.

Υπάρχουν ιδικές περιπτώσεις όπως το ότι ο ένας γράφος είναι υπογράφος του άλλου όπου θα δεχθούμε ότι η ομοιότητα είναι μονάδα αυτό έγκειται στον τρόπο που ορίσαμε τον αλγόριθμο του containment similarity και θα μπορούσαμε ότι για τις ανάγκες κατασκευής ενός recommendation συστήματος μας εξυπηρετεί και είναι λογικά ευσταθές.

Καθ' όλη την διάρκεια που εκτελείται η μέθοδος παρουσιάζονται στον χρήστη τα διάφορα στάδια σύγκρισης των ακμών. Παρουσιάζεται η κάθε ακμή που υπάρχει στον πρώτο γράφο σε ποια τριγράμματα αντιστοιχεί έπειτα αν υπάρχει αυτή η αντίστοιχη ακμή στον δεύτερο γράφο και από ποιους αριθμούς κόμβων αποτελείται. Στην περίπτωση που δεν υπάρχει ενημερώνουμε τον χρήστη αντίστοιχα.

Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
static public void contSimilarity(String FileNameNodes1, String FileNameEdges1, String
FileNameNodes2, String FileNameEdges2){
int numberNodesFile1, numberNodesFile2, numberEdgesFile1, numberEdgesFile2;
int [] edge1= new int[2];
char [] Node1_1= new char[3];
char [] Node1_2= new char[3];
int i;
int NumberOfCommonEdges=0,NumberOfAllEdges=0;
try { // It opens the files of edges.
BufferedReader edges1 = new BufferedReader (new FileReader(FileNameEdges1)) ;
BufferedReader edges2 = new BufferedReader (new FileReader(FileNameEdges2)) ;
numberNodesFile1=Integer.parseInt(edges1.readLine()); // It keeps in variables the number of e
and nodes.
numberEdgesFile1=Integer.parseInt(edges1.readLine());
numberNodesFile2=Integer.parseInt(edges2.readLine());
numberEdgesFile2=Integer.parseInt(edges2.readLine());
System.out.println("-----***The comparison of two Graphs has begun.***-----");
System.out.println("[1st Graph] The number of Nodes is: " + Integer.toString(numberNodesFile
The number of Edges is: "+numberEdgesFile1);
System.out.println("[2nd Graph] The number of Nodes is: " + Integer.toString(numberNodesFil
The number of Edges is: "+numberEdgesFile2);

Pattern COMMA = Pattern.compile(" ");
String line;
```

```
  while ((line = edges1.readLine()) != null) {  // The line is a string. It has letters-numbers which
separated from space ' '
  i=0; // It reads all the lines one by one each line has two number which represent an edge.
  // It separates the two numbers of the line. Each number represent a node.
  // The array Edge[] has the number of the 1st node in the 1st position and the number of the 2nd
node in the 2nd position.
  // These two nodes make an edge.
  for (String token : COMMA.split(line)) {
    try {
    edge1[i++] = Integer.parseInt(token);} // Edge1 is an array which keeps the number of the 1st
node in 1st position and
    catch (NumberFormatException ex){      // the number of the 2nd node in the 2nd position.
    System.err.println(token + " is not a number");}
  }

  nodes3Gram( FileNameNodes1,edge1,Node1_1,Node1_2); // It takes the edge1 and returns two
3-Grams which adjoins the edge.

  int numberOfNode2_1, numberOfNode2_2;
  numberOfNode2_1=findNumberOfNode(FileNameNodes2,Node1_1); // It gets a 3-Gram and
returns the number which represent it in nodes2.txt
  numberOfNode2_2=findNumberOfNode(FileNameNodes2,Node1_2); // A 3-Gram / node may
be represented with different numbers in each graph.

    // It checks if there is an edge in Graph2 which joins numberOfNode2_1 to numberOfNode2_
edges.txt
    if(doWeHaveEdgefromNodes(FileNameEdges2, numberEdgesFile2, numberOfNode2_1,
numberOfNode2_2))
    {NumberOfCommonEdges++; // The case that the edge from Graph1 is also existed in Graph2
    System.out.print("The "+ Integer.toString(NumberOfAllEdges++)+ " edge: ");
    System.out.print(edge1[0]);
    System.out.print(" - ");
    System.out.print(edge1[1]); System.out.print(" with 3-Grams nodes |");
    System.out.print(Node1_1); System.out.print("| - |"); System.out.print(Node1_2);
    System.out.println("| exists in both Graphs.");
    }else{ // The case that it doesn't exist.
    System.out.print("The "+ Integer.toString(NumberOfAllEdges++)+ " edge: ");
    System.out.print(edge1[0]);
    System.out.print(" - ");
    System.out.print(edge1[1]); System.out.print(" with 3-Grams nodes |");
    System.out.print(Node1_1); System.out.print("| - |"); System.out.print(Node1_2);
    System.out.println("| DOES NOT exist in both Graphs.");}
  }
```

```
System.out.println("The number of Common Edges is:
"+Integer.toString(NumberOfCommonEdges));
System.out.print("\nThe Containment Similarity is: ");
  if(numberEdgesFile1< numberEdgesFile2){ //numberEdgesFile1 counts the number of edges a
more for the number of edges and nodes. So I subtract 2.
  System.out.println((float)NumberOfCommonEdges/(numberEdgesFile1));}
  else{ System.out.println((float)NumberOfCommonEdges/(numberEdgesFile2));}
edges1.close();
edges2.close();
}
catch( IOException e ) {
e . printStackTrace () ;}

}
```

8.3.6 static private void nodes3Gram(String FileNameNodes1, int[] edge1, char[] Node1_1, char[] Node1_2)

Η μέθοδος nodes3Gram δέχεται μια ακμή που αναπαρίσταται με δύο αριθμούς που αντιστοιχούν σε δύο κόμβους και επιστρέφει την ίδια ακμή με τριγράμματα αντί για αριθμούς. Εκτός από τους δύο πίνακες με τα τριγράμματα και τους αριθμούς δέχεται ως είσοδο και το όνομα του αρχείου των κόμβων.

Με μια επαναληπτική μέθοδο διατρέχουμε το αρχείο των κόμβων ανά τρεις χαρακτήρες που αντιστοιχούν σε ένα τρίγραμμα. Θα διατρέξουμε τόσα τριγράμματα όσος είναι ο αριθμός που έχουμε δεχθεί ως είσοδο. Στο τέλος της επαναληπτικής διαδικασίας θα έχουμε φτάσει στο σωστό τρίγραμμα που αναζητούμε.

Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
static private void nodes3Gram(String FileNameNodes1, int[] edge1, char[] Node1_1, char[]
Node1_2) throws IOException {
int j;
try {
BufferedReader nodes1 = new BufferedReader (new FileReader(FileNameNodes1)) ; // It opens
file of nodes.
```

```
for ( j=0; j<=edge1[0]; j++){  nodes1.read(Node1_1);} // It reads all the nodes until the node I lo
for.
nodes1.close();

BufferedReader nodes1a = new BufferedReader (new FileReader(FileNameNodes1)) ; // Same a
above.
for ( j=0; j<=edge1[1]; j++){nodes1a.read(Node1_2);}
nodes1a.close();}

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

}
```

8.3.7 private static boolean doWeHaveEdgefromNodes(String FileNameEdges, int numberOfLines, int numberOfNode1_1, int numberOfNode1_2)

Η μέθοδος doWeHaveEdgefromNodes δέχεται ως είσοδο το όνομα του αρχείου με τις ακμές, το πλήθος των ακμών και τους αριθμούς που αντιστοιχούν σε δύο κόμβους. Αν υπάρχει μέσα στο αρχείο των ακμών ακμή που να συνδέει αυτούς τους δύο κόμβους τότε η μέθοδος επιστρέφει την λογική τιμή true αλλιώς επιστρέφει false.

Αρχικά ανοίγουμε το αρχείο των ακμών και διατρέχουμε μία προς μια τις γραμμές του που αντιστοιχούν σε μία ακμή. Αν οι δύο ακέραιοι αριθμοί που βρίσκονται στην τρέχουσα γραμμή είναι ίδιοι με τους δύο ακέραιους που δεχθήκαμε ως είσοδο τότε βρέθηκε η ακμή και επιστρέφουμε true. Αν αναζητήσουμε μέχρι τέλος το αρχείο και δεν έχει βρεθεί η ακμή τότε επιστρέφουμε false.

Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
private static boolean doWeHaveEdgefromNodes(String FileNameEdges, int numberOfLines, in
numberOfNode1_1, int numberOfNode1_2) throws IOException {
int i;
String line;
BufferedReader edges2 = new BufferedReader (new FileReader(FileNameEdges)) ; // It opens t
file of edges.
```

```
  for(i=0;i<=numberOfLines+1;i++){ // It reads all the lines - edges.
  line = edges2.readLine();
    if (line.equals(Integer.toString(numberOfNode1_1)+ " " + Integer.toString(numberOfNode1_
    edges2.close(); return true;} // It compares the contains of each line - edge with the number of
edges.
  }
edges2.close();

return false;

}
```

8.3.8 private static int findNumberOfNode(String FileNameNodes2, char[] node1_1)

Η μέθοδος findNumberOfNode δέχεται ως είσοδο ένα τρίγραμμα και το όνομα του αρχείου των κόμβων και επιστρέφει τον αριθμό που αντιστοιχεί σε αυτό το τρίγραμμα. Ο αριθμός αυτός είναι η θέση που έχει το τρίγραμμα μέσα στο αρχείο των κόμβων.

Αρχικά ανοίγουμε το αρχείο των κόμβων και διατρέχουμε με μια επαναληπτική διαδικασία ένα προς ένα όλα τα τριγράμματα. Για κάθε τρίγραμμα που διαβάζουμε αυξάνουμε την μεταβλητή position κατά ένα. Αν το τρίγραμμα βρεθεί τότε βγαίνουμε από την επαναληπτική διαδικασία και έχουμε στην μεταβλητή position την θέση του. Αν δεν βρεθεί επιστρέφουμε -1.

Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
private static int findNumberOfNode(String FileNameNodes2, char[] node1_1 ) throws
IOException{
boolean founded=false;
char [] readed= new char[3]; // The 3-Gram which is read by the file.
int position=-1;
BufferedReader nodes2 = new BufferedReader (new FileReader(FileNameNodes2)); // It opens
file of nodes.
  while(nodes2.read(readed)!=-1){ // It reads all the 3-Gram nodes one by one.
  position++; // The position of the 3-Gram / node which is compared.
  if(Arrays.equals(node1_1,readed)){founded=true; break;}
}
```

```
nodes2.close();

if (founded)
return position;
else return -1;

}
```

8.3.9 public static void simpleGraph(String fileNameEdges, String fileNameSimpleEdges)

Η μέθοδος simpleGraph δέχεται έναν γράφο που έχει παράλληλες ακμές και παράγει έναν γράφο με απλές ακμές. Ως είσοδο έχουμε μόνο τα ονόματα των δύο αρχείων που έχουν τις ακμές.

Αρχικά ανοίγουμε το αρχείο που έχει τις παράλληλες ακμές για να το διαβάσουμε. Διαβάζουμε πόσες ακμές έχει σύνολο. Έπειτα ανοίγουμε το αρχείο που θα γράψουμε τις απλές ακμές.

Με μια επαναληπτική διαδικασία διατρέχουμε όλες της ακμές από το αρχείο των παράλληλων ακμών ελέγχουμε αν αυτή η ακμή υπάρχει ήδη με την κλήση της μεθόδου isIt1stTimetoWrite. Αν δεν υπάρχει τότε την καταγράφουμε στο αρχείο των απλών ακμών με την μέθοδο writeNewSimpleEdge. Αν υπάρχει απλώς προχωράμε στον επαναληπτικό βρόχο χωρίς να εκτελέσουμε καμία παραπάνω εργασία.

Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
public static void simpleGraph(String fileNameEdges, String fileNameSimpleEdges) throws
IOException{
int numberOfNewEdges=0;
int numberEdgesFile, i;
String line,numberOfNodes;
BufferedReader in = new BufferedReader (new FileReader (fileNameEdges)); // It opens the file
the edges (parallel edges included).
numberOfNodes=in.readLine();

FileWriter file = new FileWriter(fileNameSimpleEdges); // It creates the file of the edges (parall
edges NOT included).
BufferedWriter bf = new BufferedWriter(file);
```

```
bf.write(numberOfNodes+"\n0\n"); // It initializes the simpleEdges.txt It writes the right number
nodes but 0 for edges.
bf.close();

numberEdgesFile=Integer.parseInt(in.readLine());
  for(i=0; i<numberEdgesFile;i++){ // It reads all the lines - edges.
  line=in.readLine();
  if(isIt1stTimetoWrite(fileNameSimpleEdges, line)){writeNewSimpleEdge(fileNameSimpleEdg
line,++numberOfNewEdges);}
  //It checks for each edge if it already exists in the fileSimpleEdges. If not, it writes it. If yes, it d
write it.
  }
in.close();

}
```

8.3.10 private static boolean isIt1stTimetoWrite(String fileNameSimpleEdges, String line)

Η μέθοδος isIt1stTimetoWrite είναι μια βοηθητική μέθοδος που την συναντήσαμε να την καλούμε απο την μέθοδο simpleGraph. Η μέθοδος isIt1stTimetoWrite ελέγχει αν υπάρχει ήδη μια ακμή στο αρχείο των ακμών ή όχι. Δέχεται ως είσοδο το όνομα του αρχείου των ακμών και ένα String που αναπαριστά την ακμή.

Αρχικά ανοίγει το αρχείο των ακμών και διαβάζει μία προς μία όλες τις ακμές αν μια ακμή είναι ίδια με το String της ακμής που δώσαμε στην είσοδο της τότε επιστρέφει false. Αν φτάσει στο τέλος του αρχείου και δεν την έχει βρει τότε επιστρέφει true.

Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
private static boolean isIt1stTimetoWrite(String fileNameSimpleEdges, String line) throws
IOException {
BufferedReader in = new BufferedReader (new FileReader (fileNameSimpleEdges)) ; // It opens
file of simple edges.
int numberOfEdges, i;
in.readLine();
numberOfEdges = Integer.parseInt(in.readLine());
  for(i=0;i<numberOfEdges;i++){ // It reads and compare all the lines - edges one by one with th
edge of input.
```

```
  if(line.equals(in.readLine())){in.close();return false;}
  }
in.close();
return true;

}
```

8.3.11 private static void writeNewSimpleEdge(String fileNameSimpleEdges, String line,int numberOfNewEdges)

Η μέθοδος writeNewSimpleEdge είναι μια βοηθητική μέθοδος που την συναντήσαμε να την καλούμε από την μέθοδο simpleGraph. Δέχεται ως είσοδο το όνομα του αρχείου των ακμών, μία ακμή με την μορφή ενός String και το πλήθος των ακμών. Σκοπός της είναι να καταγράψει μέσα στο αρχείο των ακμών την καινούρια ακμή.

Σημείο που αξίζει να δώσουμε προσοχή είναι ότι αφου ανοιχτεί το αρχείο των απλών ακμών, καταγραφεί η καινούρια ακμή και το κλείσουμε θα έχουμε μια παραπάνω ακμή στο αρχείο των ακμών χωρίς να έχουμε αντικαταστήσει το συνολικό πλήθος των ακμών με έναν αριθμό κατά ένα μεγαλύτερο. Καλούμε την μέθοδο replace για να αλλάξει την δεύτερη γραμμή του αρχείου που έχει το πλήθος των ακμών με το σωστό αριθμό ακμών.

Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
private static void writeNewSimpleEdge(String fileNameSimpleEdges, String line,int
numberOfNewEdges) throws IOException {
FileWriter file = new FileWriter(fileNameSimpleEdges,true); // It opens the file of simple edges
write one more edge.
BufferedWriter bf = new BufferedWriter(file);
bf.append(line);
bf.close();
replace(fileNameSimpleEdges,numberOfNewEdges);
// It replaces the second line of simple edges file with a number increased by one.
//It is necessary cause we added one more edge in the end of the file.

}
```

8.3.12 public static void replace(String oldFileName, int numberOfNewEdges)

Η μέθοδος replace είναι μια βοηθητική μέθοδος που την συναντήσαμε να την καλούμε από την μέθοδο writeNewSimpleEdge. Έχει σκοπό να διορθώσει το πλήθος των ακμών που αναγράφει το αρχείο των ακμών αντικαθιστώντας τον με τον σωστό.

Η μέθοδος replace δέχεται το όνομα του αρχείου των ακμών και το πλήθος των ακμών. Ανοίγει το αρχείο για διάβασμα και ένα αρχείο που το έχει ονομάσει tmp_try.dat για γράψιμο. Έπειτα μια μια προς μία της γραμμές που έχει τις διαβάζει και τις γράφει στο αρχείο mp_try.dat έτσι όπως είναι. Μόνο την γραμμή που αναφέρει το πλήθος των ακμών την αντικαθιστά με το νέο σωστό πλήθος ακμών. Τέλος διαγράφει το αρχείο από όπου διάβαζε και μετονομάζει το αρχείο απο mp_try.dat στο κανονικό ονομα του αρχείου των ακμών.

Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
public static void replace(String oldFileName, int numberOfNewEdges) throws IOException {
String tmpFileName = "tmp_try.dat"; // It makes a temporary file which has the old data.
String newNumber=Integer.toString(numberOfNewEdges);
BufferedReader br = null;
BufferedWriter bw = null;
br = new BufferedReader(new FileReader(oldFileName));
bw = new BufferedWriter(new FileWriter(tmpFileName));
String line;
line=br.readLine();
bw.write(line+"\n"); // It writes the old number of nodes.
line=br.readLine();
bw.write(newNumber+"\n"); // It writes the NEW number of edges.

// It writes all the data as they are from third line onwards.
while ((line = br.readLine()) != null) {bw.write(line+"\n");}
// Once everything is complete, delete old file.
br.close();bw.close();
File oldFile = new File(oldFileName);
oldFile.delete();

// And rename tmp file's name to old file name.
File newFile = new File(tmpFileName);
```

```
newFile.renameTo(oldFile);
}
```

8.4 Οι κλάσεις Bag, Graph, In, Input, Stack, StdIn, StdOut

Για την αναπαράσταση του γράφου χρησιμοποιήσαμε μια σειρά έτοιμων κλάσεων που τις πήραμε από το βιβλίο *Algorithms, 4th Edition* by Robert Sedgewick and Kevin Wayne. Στο τέταρτο κεφάλαιο αυτού του βιβλίου αναπτύσσονται θεωρητικά τα βασικότερα αλγοριθμικά θέματα γύρω από τους γράφους και δίνονται κάποιες προγραμματιστικές υλοποιήσεις στην γλώσσα java.

8.5 το πρόγραμμα FromTwitterToUsers

Μας έχει δοθεί μια σειρά αρχείων πολύ μεγάλου μεγέθους, από δύο έως είκοσι gigabyte με tweets από πάρα πολλούς χρήστες. Μόνο του το πιο μικρό αρχείο περιέχει 18.572.084 tweets. Τα tweets αυτά καταγράφονται στο αρχείο με την χρονική σειρά που αναρτήθηκαν στο twiiter.

Τα tweets έχουν συγκεκριμένη μορφή μέσα στο αρχείο. Κάθε tweet αποτελείται από τρεις γραμμές και μια κενή γραμμή διαχωρίζει τα tweets αναμεταξύ τους. Η πρώτη γραμμή ξεκινά με το γράμμα T και περιλαμβάνει την ώρα που δημοσιεύθηκε το tweet. Η δεύτερη γραμμή ξεκινά με το γράμμα U και αναφέρει το όνομα του χρήστη που δημοσίευσε το tweet. Η τρίτη γραμμή ξεκινά με το γράμμα W και έχει το περιεχόμενο από το τι δημοσίευσε ο χρήστης.

Το αρχείο περιέχει τα tweets όλων των χρηστών με χρονική ακολουθία. Από αυτό το αρχείο θέλουμε να φτιάξουμε μια σειρά αρχείων που το κάθε αρχείο αντιστοιχεί σ' ένα χρήστη και να περιέχει τα tweets που έχει αναρτήσει. Αυτήν την εργασία θα επιτελέσει το πρόγραμμά μας FromTwitterToUsers. Το πρόγραμμα αποτελείται από δύο μεθόδους την main και την writeToFile.

8.5.1 public static void main(String[] args)

Στην κύρια μέθοδο αρχικά ανοίγουμε το αρχείο fileIn.txt που περιέχει όλα τα tweets. Έπειτα με μια επαναληπτική διαδικασία το διαβάζουμε γραμμή προς γραμμή. Αν μια γραμμή ξεκινά από το U http://twitter.com/ καταλαβαίνουμε ότι έπεται το όνομα του χρήστη. Το οποίο παρεμπιπτόντως γνωρίζουμε ότι ξεκινά μετά τον εικοστό πρώτο χαρακτήρα. Οπότε καταχωρούμε στην μεταβλητή name το όνομα του χρήστη με την βοήθεια της substring έπειτα από τον εικοστό πρώτο χαρακτήρα της γραμμής που διαβάσαμε. Συνεχίζουμε και καταχωρούμε στην μεταβλητή tweet την επόμενη γραμμή που περιλαμβάνει το tweet που δημοσίευσε ο χρήστης. Τέλος καλούμε την μέθοδο writeToFile που αναλαμβάνει να γράψει στο αντίστοιχο αρχείο του χρήστη το tweet που διαβάστηκε.

Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
public static void main(String[] args) throws IOException {
        //      String fileToRead= "fileIn.txt";
                int numberOfLine=0;
                String line,name,tweet;
                System.out.println(System.getProperty("user.dir")) ;
                        BufferedReader fileIn = new BufferedReader (new FileReader("fileIn.txt'

                        while ((line = fileIn.readLine()) != null) {
                                numberOfLine++;
                                if (line.startsWith("U  http://twitter.com/")){//21
                                name=line.substring(21);
                                tweet=fileIn.readLine();
                                writeToFile(name,tweet.substring(2),numberOfLine);
                                numberOfLine++;
                                }
                        }
                        fileIn.close();



                 System.out.println("The Program finished!");
                 //writeToFile(".Nikos","grafo pola tweets!",40);
        }
```

8.5.2 public static void writeToFile(String nameOfUser, String TweetToWrite, int

numberOfLine)

Η μέθοδος writeToFile έχει σκοπό να γράψει στο αρχείο που αντιστοιχεί στον κάθε χρήστη το tweet που διάβασε η κύρια μέθοδος. Δέχεται ως είσοδο το όνομα του χρήστη, το tweet και σε πια σειρά διαβάστηκε.

Λόγο του ότι έχουμε πολλούς χρήστες και δεν μπορούμε να δημιουργήσουμε τόσα πολλά αρχεία σ' έναν φάκελο επιλέξαμε να δημιουργήσουμε έναν υποφάκελο για κάθε γράμμα της αλφαβήτου και έναν ακόμη για οποιαδήποτε άλλο χαρακτήρα μπορεί να ξεκινά το όνομα ενός χρήστη.

Στην μεταβλητή nameOfFolder καταχωρούμε το πρώτο γράμμα από το όνομα καθενός χρήστη και στις μεταβλητές whereToWrite, whereToWriteNum την θέση μέσα στο σκληρό δίσκο μαζί με το όνομα του χρήστη όπου θα αποθηκευτούν τα tweets.

Επιλέξαμε το κάθε tweet να το αποθηκεύουμε σε δύο αρχεία ένα με το να είναι τα tweets στο αρχείο έτοιμα να δοθούν ως είσοδο στο πρόγραμμα μας thesis και ένα άλλο που θα είναι σαν το προηγούμενο με την μόνη διαφορά ότι προσθέσαμε στην αρχή καθενός tweet την γραμμή όπου μέσα στο αρχείο βρέθηκε το tweet. Οπότε τα απλά θα αποθηκεύονται στην διεύθυνση whereToWrite και αυτά που θα έχουν και την θέση όπου βρέθηκαν στην διεύθυνση whereToWriteNum

Η διαδικασία απο εδώ και πέρα είναι απλή. Ανοίγουμε τα αρχεία, στο τέλος του αρχείου γράφουμε το String με το Tweet και έπειτα το κλείνουμε. Μία φορά και για τις δύο διευθύνσεις που έχουμε σχηματίσει πριν για να τα γράψουμε.
Παρακάτω παραθέτουμε τον κώδικα της μεθόδου:

```
public static void writeToFile(String nameOfUser, String TweetToWrite, int numberOfLine) t
IOException{
        char nameOfFolder=nameOfUser.charAt(0);
        String whereToWrite="here",whereToWriteNum="here";
        if(Character.isLetter(nameOfFolder))
        {whereToWrite= "simple\\"+Character.toString(nameOfFolder)+"\\"+nameOfUser+".txt

whereToWriteNum="numbered\\"+Character.toString(nameOfFolder)+"\\"+nameOfUser+"Num
}
        else{whereToWrite="simple\\"+"other"+"\\"+nameOfUser+".txt";
        whereToWriteNum="numbered\\"+"other"+"\\"+nameOfUser+"Num.txt";}

        FileWriter file = new FileWriter(whereToWrite,true);
        BufferedWriter fileOut = new BufferedWriter(file);
        fileOut.write(TweetToWrite+"\n");
```

```
        fileOut.close();

        FileWriter fileNum = new FileWriter(whereToWriteNum,true);
        BufferedWriter fileOutNum = new BufferedWriter(fileNum);
        fileOutNum.write(Integer.toString(numberOfLine)+" "+TweetToWrite+"\n");
        fileOutNum.close();

}
```

8.6 Τα αρχεία που χρησιμοποιεί η προγραμματιστική υλοποίηση του recommendation system με τριγράμματα.

Η προγραμματιστική μας υλοποίηση κάνει χρήση μιας σειράς αρχείων από όπου διαβάζει τα tweets, τα καταχωρεί ανά χρήστες, σχηματίζει τον γράφο όπου επιτρέπονται οι παράλληλες ακμές και τον απλό γράφο. Ας δούμε όλα τα αρχεία αναλυτικά

8.6.1 Το αρχείο fileIn.txt

Το αρχείο fileIn.txt περιέχει όλα τα tweets από όλους τους χρήστες με την χρονική σειρά που έχουν δημοσιευθεί. Είναι πολύ μεγάλο αρχείο για αυτό και είναι δύσκολο από εδώ να διαβάσει κατευθείαν τα tweets το πρόγραμμα μας thesis. Το αρχείο αυτό θα δοθεί ως είσοδος στο πρόγραμμα FromTwitterToUsers όπου θα σχηματίσει τα tweets για κάθε χρήστη ξεχωριστά.

Η μορφή που έχει το αρχείο είναι στην πρώτη γραμμή να αναγράφεται ο συνολικός αριθμός των tweets που περιέχει. Έπειτα ακολουθούν όλα τα tweets με μία κενή γραμμή να τα διαχωρίζει αναμεταξύ τους. Κάθε tweet αποτελείται από τρεις γραμμές. Η πρώτη γραμμή ξεκινά με το T και αναγράφει την ώρα δημοσίευσης του. Η δεύτερη γραμμή ξεκινά με το U και αναγράφει τον χρήστη που το δημοσίευσε και η τρίτη γραμμή ξεκινά με το W και αναγράφει τα περιεχόμενα του tweet.

Ακολουθεί παράδειγμα της μορφής που είναι τα tweets μέσα στο αρχείο

T 2009-06-11 17:19:09
U http://twitter.com/alltheyevaneed
W I love ya twitter gangstas hahahahaha

T 2009-06-11 17:19:09
U http://twitter.com/bekkargh
W I've had pikachu down for a few days and i've read the letters a few
times, bu everything else is up there.

T 2009-06-11 17:19:09
U http://twitter.com/jghitzert
W @davidcmolina Got a call from the United Way, Thanks a bunch my man.

8.6.2 Τα αρχεία Tweet1.txt και Tweet2.txt

Τα αρχεία αυτά περιέχουν για έναν χρήστη το καθ' ένα τα tweets που δημοσίευσε. Τα tweets αυτά διαχωρίζονται το ένα από το άλλο με μια αλλαγή γραμμής. Δηλαδή κάθε γραμμή αντιστοιχεί και σ' ένα tweet. Δεν υπάρχει κάποιο ιδιαίτερο σημείο που πρέπει να αναφέρουμε πέρα από το ότι μετά το τελευταίο tweet δεν ακολουθεί άλλη αλλαγή γραμμής αλλά στο τελευταίο γράμμα του τελευταίου tweet είναι και το τέλος του αρχείου.

8.6.3 Τα αρχεία Edges1.txt και Edges2.txt

Αφού το πρόγραμμα διαβάσει διαβάσει το αρχείο των tweets θα σχηματίσει τον γράφο που του αντιστοιχεί. Τα αρχεία Edges1.txt και Edges2.txt περιέχουν τις ακμές για τους γράφους των δύο χρηστών. Παράλληλες ακμές και self loops υπάρχουν.

Στην πρώτη γραμμή του αρχείου αναγράφεται το πλήθος των κόμβων. Στην δεύτερη ακμή το πλήθος των ακμών και έπειτα κάθε γραμμή θα αντιστοιχεί σε μια ακμή όπου ο πρώτος ακεραίως είναι ο ένας κόμβος της ακμής και ο δεύτερος ακεραίως ο δεύτερος κόμβος της ακμής. Αναμεταξύ τους παρεμβάλλεται ένας κενός χαρακτήρας για να τα διαχωρίσει.

Ακολουθεί η αρχή από αρχείο κλάδων.

```
8
54
1 0
2 0
2 1
3 0
3 1
3 2
4 1
4 2
4 3
5 2
...
```

Όπως παρατηρείται το αρχείο αυτό έχει οχτώ κόμβους και πενήντα τέσσερις ακμές οι οποίες δεν αναγράφονται όλες.

8.6.4 Τα αρχεία SimpleEdges1.txt και SimpleEdges2.txt

Τα αρχεία SimpleEdges1.txt και SimpleEdges2.txt είναι μια απλοποιημένη μορφή των προηγούμενων αρχείων. Η διαφορά που έχουμε με τα προηγούμενα είναι ότι όλοι οι παράλληλοι κλάδοι αφαιρούνται Κάθε κλάδος υπάρχει το πολύ μια φορά. Πάλι στις δύο πρώτες γραμμές αναφέρουμε το πλήθος των κόμβων και των ακμών.

8.6.5 Τα αρχεία Nodes1.txt και Nodes2.txt

Τα αρχεία Nodes1.txt και Nodes2.txt περιέχουν όλα τα τριγράμματα που σχηματίστηκαν από τα αρχεία των tweets. Το κάθε τρίγραμμα αντιστοιχεί σ' έναν κόμβο του γράφου. Τα τριγράμματα δεν χωρίζονται αναμεταξύ τους από κάποιο χαρακτήρα άλλα το ένα ακολουθεί το άλλο σε σειρά. Όταν θα αναφερόμαστε στον κόμβο εφτά θα καταμετρούμε στην σειρά οχτώ τριγράμματα για να το εντοπίσουμε. Το πρώτο τρίγραμμα αντιστοιχεί στον κόμβο μηδέν.

Ακολουθεί παράδειγμα του ποία τριγράμματα θα υπάρχουν στο αρχείο αν διαβαστεί ως είσοδος μόνο το tweet home_phone.

homomeme_e_p_phphohonone

8.6.6 Το αρχείο tmp_try.dat

Κατά την εκτέλεση της μεθόδου replace χρειάζεται να μετονομάσουμε το αρχείο των απλών ακμών για να του κάνουμε κάποια τροποποίηση στα δεδομένα που έχει. έτσι προκύπτει για λίγο το tmp_try.dat το οποίο έπειτα μετονομάζεται σε simpleEdge1.txt ή simpleEdge2.txt αντίστοιχα.

8.7 Πειραματικά αποτελέσματα

Διαλέξαμε πέντε συγκεκριμένα θέματα και για το κάθε θέμα πέντε χρήστες που να γράφουν tweets για αυτό. Από τον κάθε χρήστη πήραμε 20 tweets. Οι χρήστες και τα θέματα έχουν την εξής δομή.

Cooking	[C-User1]	C-User2	C-User3	C-User4	C-User5
Gardening	[G-User1]	G-User2	G-User3	G-User4	G-User5
Linux	[L-User1]	L-User1	L-User3	L-User4	L-User5
Wines	[W-User1]	W-User2	W-User3	W-User4	W-User5
Fitness	[F-User1]	F-User2	F-User3	F-User4	F-User5

Πίνακας 3Χρήστες και θέματα που ανήκουν

Οι χρήστες που βρίσκονται στην πρώτη στήλη και το όνομα τους περιέχεται μέσα σε αγκύλες επιλέχθηκαν και έτσι θα τους αποκαλούμε από εδώ και πέρα ως χρήστες αναφοράς. Οι χρήστες αυτοί που επιλέχθηκαν τυχαία θα συγκριθούν με όλους τους υπόλοιπους της ομάδας τους με το πρόγραμμα που κάνει χρήση των τριγραμμάτων και τις μεθόδους Containment Similarity (CS), Value Similarity (VS) που αναφέραμε πιο πριν.

Αρχικά συγκρίνουμε όλους τους χρήστες που ανήκουν σε ένα θέμα με τον χρήστη αναφοράς. Αυτές οι συγκρίσεις περιμένουμε να μας δώσουν έναν υψηλό βαθμό ομοιότητας αφού οι χρήστες ασχολούνται με παρόμοια θέματα. Αυτό θα το κάνουμε και για τα πέντε θέματα που έχουμε επιλέξει

και για τις δύο μεθόδους σύγκρισης που έχουμε.

Έπειτα θα συγκρίνουμε χρήστες που ασχολούνται με ανόμοια θέματα. Θα επιλέξουμε κάποιους τυχαίους χρήστες από κάθε θέμα και θα τους συγκρίνουμε αναμεταξύ τους. Αυτό θα το κάνουμε για να δούμε δύο τυχαίοι χρήστες τι ομοιότητα παρουσιάζουν. Αναμένουμε ότι οι συγκρίσεις αυτές θα παρουσιάσουν ένα χαμηλό αριθμό ομοιότητας.

Για να θεωρήσουμε πετυχημένη την μέθοδο σύγκρισης των τριγραμμάτων θα πρέπει όσο το δυνατόν σε μεγαλύτερο ποσοστό να είναι πιο υψηλές οι τιμές που παίρνουμε από τις συγκρίσεις μεταξύ χρηστών που ασχολούνται με όμοια θέματα από ότι με ανόμοια θέματα.

Για κάθε χρήστη έχουμε επιλέξει 20 τυχαία tweets. Την πρώτη φορά θα συγκρίνουμε τα tweets όπως ακριβώς τα δημοσίευσαν οι χρήστες τους. Πολλά από τα tweets περιέχουν διευθύνσεις url που παραπέμπουν σε άλλες ιστοσελίδες. Την δεύτερη φορά θα επιλέξουμε να μην περιλαμβάνονται αυτά τα url στα tweets συγκρίνουμε. Την τρίτη φορά θα αφαιρέσουμε από τα tweets όλα τα σημεία στίξης, αριθμούς και για τον σχηματισμό των τριγραμμάτων δεν θα λάβουμε υπόψιν μας αν τα γράμματα είναι κεφαλαία ή μικρά. Στο τέλος θα συγκρίνουμε τα tweets χωρίς να περιλάβουμε τους κενούς χαρακτήρες που υπάρχουν μεταξύ των λέξεων. Τριγράμματα θα σχηματίζονται μόνο εντός της κάθε λέξης.

Από τους διάφορους τρόπους που μπορούμε να συγκρίνουμε τα tweets αναμεταξύ τους θα προσπαθήσουμε να εντοπίσουμε ποια παράγει μεγαλύτερη διαφορά στους βαθμούς ομοιότητας μεταξύ των χρηστών που ασχολούνται με παρόμοια θέματα από ότι στους χρήστες με ανόμοια θέματα. Ιδανικά θα θέλαμε να βρούμε μια τιμή ομοιότητας όπου να πούμε ότι πάνω από αυτή την τιμή οι χρήστες ασχολούνται με παρόμοια θέματα και κάτω από αυτήν την τιμή με διαφορετικά θέματα. Την τιμή αυτή από εδώ και πέρα θα την ονομάζουμε οριακή τιμή. Τιμές που βρίσκονται κοντά στην διαχωριστική τιμή ομοιότητας θα είναι τιμές για τις οποίες από μόνες τους δεν θα μπορούσαμε να μιλήσουμε με σιγουριά. Τιμές κατά πολύ μεγαλύτερες από την οριακή τιμή θα προσδιορίζουν μια μεγάλη ομοιότητα στα θέματα που ασχολούνται. Ενώ τιμές κατά πολύ μικρότερες από την οριακή τιμή προσδιορίζουν ότι οι χρήστες ασχολούνται με ανόμοια θέματα.

Ας δούμε αναλυτικά τις συγκρίσεις αυτές.

8.7.1 Σύγκριση Tweets με Urls

Αρχικά συγκρίναμε τα tweets όπως ακριβώς τα συνέταξαν και τα δημοσίευσαν οι χρήστες. Αυτό μας γλιτώνει από κάποιον χρόνο προεπεξεργασίας που όπως θα δούμε θα κάνουμε στις επόμενες συγκρίσεις αλλά επιβαρύνει τον γράφο με δεδομένα που δεν έχουν σημασία. Ένα μεγάλο ποσοστό των tweets περιλαμβάνουν urls προς συνδέσμους που ασχολούνται με το θέμα που διαπραγματεύεται ο χρήστης. Αν τύχαινε δύο χρήστες να αναρτήσουν τον ίδιο σύνδεσμο τότε θα είχαμε μια μεγάλη αύξηση της ομοιότητας των δύο γράφων και κατά συνέπεια της τιμής της σύγκρισης. Πιο συνηθισμένο όμως είναι το ίδιο περιεχόμενο μιας σελίδας (ή ακόμη και παρόμοιο) να το μοιράζονται πολλές διαφορετικές διεύθυνσης Τότε κάθε χρήστης θα αναφέρεται στο ίδιο περιεχόμενο άλλα με άλλα urls. Αυτό δεν

θα αυξήσει την ομοιότητα των γράφων γιατί θα είναι εντελώς διαφορετικά τα urls και εντελώς διαφορετικά τα τριγράμματα που θα παραχθούν από τους χαρακτήρες του url. Τα urls συνήθως αποτελούν άχρηστη πληροφορία που επιβαρύνουν και στο μέγεθος του γράφου και στον χρόνο επεξεργασίας

Με Urls							
Subject			C-User2	C-User3	C-User4	C-User5	Average
Cooking	CS	[C-User1]	0,1350	0,1635	0,0920	0,1432	0,1334
	VS	[C-User1]	0,0821	0,1056	0,0539	0,0966	*0,0845*
			G-User2	G-User3	G-User4	G-User5	
Gardening	CS	[G-User1]	0,0989	**0,0732**	0,1176	0,0862	0,0940
	VS	[G-User1]	0,0529	0,0405	0,0777	0,0419	*0,0533*
			L-User2	L-User3	L-User4	L-User5	
Linux	CS	[L-User1]	0,0874	0,1564	0,0985	0,0812	0,1059
	VS	[L-User1]	0,0543	0,0924	0,0612	0,0456	*0,0634*
			W-User2	W-User3	W-User4	W-User5	
Wines	CS	[W-User1]	**0,0699**	0,0851	**0,0511**	**0,0720**	0,0695
	VS	[W-User1]	0,0465	0,0578	0,0326	0,0404	*0,0443*
			F-User2	F-User3	F-User4	F-User5	
Fitness	CS	[F-User1]	0,1039	0,1348	0,1320	0,1345	0,1263
	VS	[F-User1]	0,0824	0,0815	0,0934	0,0643	*0,0804*
						total CS:	**0,1058**
						total VS:	***0,0652***
			[L-User1]	[G-User1]	[W-User1]	[F-User1]	Average
Mixed	CS	[C-User1]	0,0690	**0,1053**	**0,0825**	0,0753	0,0830
	VS	[C-User1]	0,0518	0,0550	0,0511	0,0356	*0,0484*
			G-User2	L-User2	W-User2	F-User5	
Mixed	CS	C-User2	0,0696	0,0584	0,0699	**0,0947**	0,0731
	VS	C-User2	0,0396	0,0240	0,0431	0,0620	*0,0421*

			C-User4	G-User4	L-User3	F-User3	
Mixed	CS	W-User3	0,0500	**0,0884**	0,0800	**0,0949**	0,0783
	VS	W-User3	0,0357	0,0384	0,0526	0,0257	*0,0381*
			C-User5	L-User4	W-User4	F-User4	
Mixed	CS	G-User5	0,0516	0,0615	0,0332	0,0339	0,0451
	VS	G-User5	0,0328	0,0346	0,0225	0,0188	0,0272
						total CS:	**0,0699**
						total VS:	**0,0389**

Πίνακας 4 Πειραματικά αποτελέσματα από Tweets με Urls

Από την σύγκριση των tweets όπως αναμέναμε στο μεγαλύτερο ποσοστό τα tweets που διαπραγματεύονται παρόμοια θέματα παρουσιάζουν μεγαλύτερη ομοιότητα από τα tweets που διαπραγματεύονται ανόμοια θέματα. Ο γενικός μέσος όρος με την μέθοδο containment similarity στα tweets με παρόμοιοι θέμα ήταν 0,1058 ενώ με ανόμοιο θέμα 0,0699. Αντίστοιχα με την μέθοδο valuse similarity 0,0652 έναντι 0,0389. Αν βάζαμε μια οριακή τιμή CS=0,08 θα είχαμε τέσσερις αστοχίες από τους χρήστες με κινά θέματα και τέσσερις από τους χρήστες με αναμεμιγμένα θέματα. Συνολικά έχουμε κάνει τριάντα έξι συγκρίσεις. Αυτό μας δίνει ότι η μέθοδός μας έχει ένα ποσοστό επιτυχίας 77.8% στο να βρίσκει αν δύο τυχαίοι χρήστες διαπραγματεύονται το ίδιο θέμα ή όχι.

Στις επόμενες παραγράφους θα προτείνουμε βελτιώσεις που αυξάνουν το ποσοστό επιτυχίας της μεθόδου των τριγραμμάτων και στο τέλος θα εξηγήσουμε γιατί υπάρχουν κάποιες αστοχίες.

8.7.2 Σύγκριση Tweets χωρίς Urls

Από τα tweets που μαζέψαμε από τον κάθε χρήστη αφαιρέσαμε τα urls και κρατήσαμε το καθαρό κείμενο που έχει γράψει μαζί με τα σημεία στίξης και την διαφορά μεταξύ κεφαλαίων και μικρών γραμμάτων. Ξανά τρέξαμε το πρόγραμμα των τριγραμμάτων, ξανά κάναμε τις συγκρίσεις και συμπληρώσαμε τον παρακάτω πίνακα.

Λόγο της μείωσης του μεγέθους των δεδομένων φτιάχτηκε μικρότερος γράφος για κάθε χρήστη και αυτό είχε το καλό ότι χρειάστηκαν λιγότεροι επεξεργαστικοί πόροι.

Χωρίς Urls							

Subject			C-User2	C-User3	C-User4	C-User5	Average
Cooking	CS	[C-User1]	0,1419	*0,1782*	*0,1054*	*0,1566*	0,1455
	VS	[C-User1]	0,0920	0,1111	0,0530	*0,1062*	0,0906
			G-User2	G-User3	G-User4	G-User5	
Gardening	CS	[G-User1]	0,0874	**0,0698**	0,1133	0,0878	0,0896
	VS	[G-User1]	0,0555	0,0360	0,0787	0,0459	0,0540
			L-User2	L-User3	L-User4	L-User5	
Linux	CS	[L-User1]	0,0964	0,1633	0,1009	**0,0780**	0,1097
	VS	[L-User1]	0,0600	0,0960	0,0602	0,0423	0,0646
			W-User2	W-User3	W-User4	W-User5	
Wines	CS	[W-User1]	**0,0699**	0,0808	**0,0439**	**0,0702**	0,0662
	VS	[W-User1]	0,0497	0,0569	0,0281	0,0533	0,0470
			F-User2	F-User3	F-User4	F-User5	
Fitness	CS	[F-User1]	0,0933	0,1312	0,1387	0,1521	0,1289
	VS	[F-User1]	0,0735	0,0729	0,0880	0,0675	0,0755
						total CS:	**0,1080**
						total VS:	***0,0663***
			[L-User1]	[G-User1]	[W-User1]	[F-User1]	Average
Mixed	CS	[C-User1]	0,0720	**0,1003**	0,0848	0,0817	0,0847
	VS	[C-User1]	0,0529	0,0569	0,0507	0,0378	*0,0496*
			G-User2	Linux	W-User2	F-User5	
Mixed	CS	C-User2	0,0758	0,0504	0,0701	0,0713	0,0669
	VS	C-User2	0,0448	0,0285	0,0534	0,0287	*0,0388*
			C-User4	G-User4	L-User3	F-User3	
Mixed	CS	W-User3	0,0540	0,0840	0,0798	**0,0924**	0,0776
	VS	W-User3	0,0369	0,0396	0,0577	0,0219	*0,0390*
			C-User5	L-User4	W-User4	F-User4	
Mixed	CS	G-User5	0,0577	0,0636	0,0330	0,0353	0,0474
	VS	G-User5	0,0356	0,0318	0,0217	0,0225	*0,0279*
						total CS:	**0,0692**

						total VS:	**0,0388**

Πίνακας 5 Πειραματικά αποτελέσματα από Tweets χωρίς Urls

Από τις τριάντα έξη συγκρίσεις που πραγματοποιήθηκαν ο μέσος όρος ομοιότητας για χρήστες που διαπραγματεύονται κινά θέματα είναι 0,1080 ενώ για χρήστες που διαπραγματεύονται διαφορετικά θέματα είναι 0,0692. Αν θεωρήσουμε οριακή τιμή CS=0,085 θα έχουμε πέντε αστοχίες για χρήστες που διαπραγματεύονται κοινά θέματα και δύο για χρήστες που διαπραγματεύονται διαφορετικά θέματα. Ο μέσος όρος επιτυχίας της μεθόδου σύγκρισης τριγραμμάτων χωρίς urls είναι 80,6%

8.7.3 Σύγκριση Tweets χωρίς Urls, σημεία Στίξεως, Αριθμούς και διαφορά μεταξύ Κεφαλαίων και Μικρών Γραμμάτων.

Από τα tweets των χρηστών αφαιρέσαμε τους συνδέσμους urls, τα σημεία στίξης, τους αριθμούς και θεωρήσαμε τους χαρακτήρες ίδιους αν είναι κεφαλαία ή μικρά. Ας δούμε αναλυτικά για ποιους λόγους θελήσαμε να εξετάσουμε την ομοιότητα με αυτές τις αλλαγές.

Κάθε σημείο στίξης παράγει και διαφοροποιεί τα τριγράμματα που σχηματίζονται Τα σημεία στίξεως δεν προσφέρουν κάποια θεματική πληροφορία. Δεν μπορούν να μας δώσουν κάποια ένδειξη για το ποιο θέμα γράφει ο χρήστης τους. Οπότε θεωρήσαμε ότι θα ήταν καλύτερο να τα παραλείψουμε. Επίσης κάθε χρήστης κάνει διαφορετική χρίση σημείων στίξεως κάποιοι τους αρέσει να τελειώνουν τις προτάσεις τους με τελεία. Κάποιοι άλλοι χρήστες με θαυμαστικό για να δώσουν έμφαση στα λεγόμενα τους και κάποιοι άλλοι αρέσκονται να τελειώνουν τις προτάσεις τους με πολλά θαυμαστικά ή ερωτηματικά.

Από την άλλη πλευρά θα μπορούσαμε να θεωρήσουμε ότι ο τρόπος χρήσης των σημείων στίξεως είναι ένα στοιχείο που θα μπορούσε να προσδιορίσει το στιλ γραφής του κάθε χρήστη. Για παράδειγμα οι νέοι χρησιμοποιούν πιο συχνά σημεία στίξεως συνεχόμενα για να κάνουν πιο έντονο αυτό που λένε. Παρόλα αυτά πιστεύουμε ότι η συνεισφορά αυτής της ιδιαίτερης περίπτωσης είναι σπάνια και δεν είναι αρκετή για να συνεισφέρει ουσιαστικά.

Αριθμοί συναντιούνται αρκετά συχνά στα tweets που αναρτούν οι χρήστες. Αριθμοί μπορεί να υπάρχουν γιατί προσδιορίζουν την ποσότητα κάποιου προϊόντος όπως για παράδειγμα πόσες κουταλιές για ένα συστατικό σε μιας συνταγής μαγειρικής. Πόσοι άνθρωποι παρακολούθησαν ένα event ακόμη και έναν ταχυδρομικό κώδικα. Στις περισσότερες περιπτώσεις ο αριθμός δεν προσδιορίζει το θέμα που διαπραγματευόμαστε δεν έχει τίποτα κοινό αν μιλήσουμε για 200 γραμμάρια αλεύρι ή 200 παρευρισκόμενους σε ένα συνέδριο. Ακόμη και οι αριθμοί που χρησιμοποιούνται για να αναφέρουν μια ημερομηνία παρουσιάζουν προβλήματα γιατί ο κάθε άνθρωπος επιλέγει διαφορετικό τρόπο να γράψει μια ημερομηνία.

Η διαφορά μεταξύ κεφαλαίων και μικρών γραμμάτων είναι ξεκάθαρο ότι δεν πρέπει να παίζει κανένα ρόλο στο να δημιουργούνται διαφορετικά τριγράμματα. Είτε κάποιος χρήστης αρέσκεται στο να γράφει κάποιες λέξεις με κεφαλαία είτε γιατί το πρώτο γράμμα μιας πρότασης είναι κεφαλαίο δεν έχει να πει κάτι διαφορετικό από έναν άλλο χρήστη που γράφει με μικρά γράμματα ή ξεκινά με μιαν άλλη λέξη την πρότασή του.

Lowercased without URLS, punctions, numbers							
Subject			C-User2	C-User3	C-User4	C-User5	Average
Cooking	CS	[C-User1]	0,1713	0,2174	*0,1326*	*0,1941*	0,1789
	VS	[C-User1]	0,1141	0,1360	0,0677	*0,1313*	*0,1123*
			G-User2	G-User3	G-User4	G-User5	
Gardening	CS	[G-User1]	**0,1062**	0,1266	0,1340	0,1714	0,1345
	VS	[G-User1]	0,0664	0,0682	0,0907	0,0957	*0,0803*
			L-User2	L-User3	L-User4	L-User5	
Linux	CS	[L-User1]	0,1290	0,2413	0,1583	0,1240	0,1631
	VS	[L-User1]	0,0804	0,1351	0,0919	0,0672	*0,0936*
			W-User2	W-User3	W-User4	W-User5	
Wines	CS	[W-User1]	**0,1003**	**0,1112**	**0,0755**	**0,1057**	0,0982
	VS	[W-User1]	0,0730	0,0771	0,0466	0,0759	*0,0682*
			F-User2	F-User3	F-User4	F-User5	
Fitness	CS	[F-User1]	0,1224	0,1549	0,1695	0,1923	0,1598
	VS	[F-User1]	0,0936	0,0849	0,1127	0,0831	*0,0936*
						total CS:	**0,1469**
						total VS:	***0,0896***
			[L-User1]	[G-User1]	[W-User1]	[F-User1]	Average
Mixed	CS	[C-User1]	0,0965	0,1212	0,1136	0,1054	0,1092
	VS	[C-User1]	0,0712	0,0676	0,0724	0,0492	*0,0651*
			G-User2	L-User2	W-User2	F-User5	
Mixed	CS	C-User2	0,1071	0,0705	0,0895	0,1224	0,0974

	VS	C-User2	0,0634	0,0385	0,0695	0,0888	*0,0651*
			C-User4	G-User4	L-User3	F-User3	
Mixed	CS	W-User3	0,0818	0,1114	0,1162	**0,1292**	0,1096
	VS	W-User3	0,0538	0,0570	0,0796	0,0312	*0,0554*
			C-User5	L-User4	W-User4	F-User4	
Mixed	CS	G-User5	**0,1383**	0,1071	0,0583	0,0860	0,0974
	VS	G-User5	**0,0832**	0,0543	0,0370	0,0590	*0,0584*
						total CS:	**0,1034**
						total VS:	***0,0610***

Πίνακας 6 Πειραματικά αποτελέσματα από Tweets χωρίς Urls, σημεία στίξεως, αριθμούς

Από τις τριάντα έξη συγκρίσεις που πραγματοποιήθηκαν ο μέσος όρος ομοιότητας για χρήστες που διαπραγματεύονται κοινά θέματα είναι **0,1469** ενώ για χρήστες που διαπραγματεύονται διαφορετικά θέματα είναι 0,1034. Αν θεωρήσουμε οριακή τιμή CS=0,122 θα έχουμε πέντε αστοχίες για χρήστες που διαπραγματεύονται κοινά θέματα και δύο για χρήστες που διαπραγματεύονται διαφορετικά θέματα. Ο μέσος όρος επιτυχίας της μεθόδου σύγκρισης τριγραμμάτων χωρίς urls είναι 80,6%

8.7.4 Σύγκριση tweets χωρίς Κενούς Χαρακτήρες, Urls, Σημεία Στίξεως, Αριθμούς και διαφορά μεταξύ Κεφαλαίων και Μικρών Γραμμάτων.

Στης προηγούμενες μεθόδους όταν κατασκευάζαμε τα τριγράμματα που αποτελούν τους κόμβους του γράφου περιλαμβάναμε και τον κενό χαρακτήρα που τον συναντάμε συνήθως να χωρίζει δύο λέξεις. Αυτό έχει ως αποτέλεσμα να υπάρχουν πολλά τριγράμματα που αποτελούνται από δύο γράμματα και ένα κενό με το κενό να είναι στην αρχή, στο τέλος ή στην μέση του τριγράμματος. Πιστεύουμε ότι αυτού του είδους τα τριγράμματα δεν περιέχουν πολλή ωφέλιμη πληροφορία Αν για παράδειγμα είχαμε δίπλα τις δύο λέξεις bought bycicle τα τριγράμματα “ht ”, “t b”, “ by” δεν προσφέρουν πολύ στο να αντιπροσωπεύσουν το νόημα του κειμένου.

Δοκιμάσαμε να σχηματίσουμε τριγράμματα μόνο με το περιεχόμενο των γραμμάτων που υπάρχουν μέσα στις λέξεις και όχι με τον τελευταίο χαρακτήρα μιας λέξεις, το κενό και τον πρώτο της επόμενης. Αν και δεν έχουμε επιτρέψει να υπάρχει κόμβος που να αποτελείται από χαρακτήρες δύο διπλανών λέξεων αφήσαμε να υπάρχουν ακμές, που να ενώνουν κόμβους που σχηματίστηκαν από διπλανές λέξεις. Εννοείται ότι αυτές οι ακμές ενώνουν κόμβους μέσα στο

παράθυρο όπως αναφέραμε σε προηγούμενη παράγραφο. Επομένως η νοηματική συνάφεια μεταξύ δύο γειτονικών λέξεων παραμένει χωρίς να παράγονται κόμβοι αμφίβολης ποιότητας.

Ας δούμε την εξής πρόταση "One apple is here." για να κατανοήσουμε πως παράγονται τα τριγράμματα. Από την πρόταση θα φτιαχτούν οι εξής κόμβοι "one", "app", "ppl", "ple" "her", "ere". Παρατηρούμε ότι από την λέξη "is" που έχει μόνο δύο γράμματα και περιβάλλεται από κενά δεν σχηματίζεται κανένα τρίγραμμα. Από την λέξη "one" παράγεται αυτούσιο το τρίγραμμα που αποτελείται από τα τρία γράμματα της. Από το "apple" παράγονται τρία τριγράμματα και από το "here" δύο. Παρόλα αυτά ο κόμβος "one" θα σχηματίσει ακμές με τους κόμβους "app", "ppl", "ple" που είναι οι τρεις πιο κοντινοί του.

Παρακάτω παραθέτουμε τα πειραματικά αποτελέσματα έτσι όπως εξήχθησαν από την εκτέλεση του προγράμματος πάνω στα δεδομένα που συλλέξαμε και προεπεξεργαστήκαμε με τους τρόπους που αναφέραμε στις προηγούμενες παραγράφους.

Lowercased without spaces, URLS, punctions, numbers							
Subject			C-User2	C-User3	C-User4	C-User5	Average
Cooking	CS	[C-User1]	0,1072	0,1502	*0,0754*	*0,1351*	0,1170
	VS	[C-User1]	0,0737	0,0972	0,0355	*0,0887*	*0,0738*
			G-User2	G-User3	G-User4	G-User5	
Gardening	CS	[G-User1]	0,0601	0,0607	0,0901	0,1394	0,0876
	VS	[G-User1]	0,0356	0,0316	0,0639	0,0714	*0,0506*
			L-User2	L-User3	L-User4	L-User5	
Linux	CS	[L-User1]	0,0565	0,1838	0,0998	0,0713	0,1029
	VS	[L-User1]	0,0382	0,1067	0,0577	0,0413	*0,0610*
			W-User2	W-User3	W-User4	W-User5	
Wines	CS	[W-User1]	**0,0499**	0,0608	**0,0265**	**0,0492**	0,0466
	VS	[W-User1]	0,0332	0,0470	0,0164	0,0365	*0,0333*
			F-User2	F-User3	F-User4	F-User5	
Fitness	CS	[F-User1]	0,1004	0,1189	0,0987	0,1066	0,1062
	VS	[F-User1]	0,0705	0,0643	0,0763	0,0445	*0,0639*
						total CS:	**0,0920**
						total VS:	***0,0565***

			[L-User1]	[G-User1]	[W-User1]	[F-User1]	Average
Mixed	CS	[C-User1]	0,0422	0,0524	**0,0598**	0,0326	0,0467
	VS	[C-User1]	0,0365	0,0296	0,0342	0,0157	*0,0290*
			G-User2	L-User2	W-User2	F-User5	
Mixed	CS	C-User2	0,0497	0,0198	0,0276	0,0482	0,0363
	VS	C-User2	0,0349	0,0108	0,0228	0,0352	*0,0259*
			C-User4	G-User4	L-User3	F-User3	
Mixed	CS	W-User3	0,0404	0,0442	0,0430	0,0541	0,0454
	VS	W-User3	0,0250	0,0267	0,0298	0,0144	*0,0240*
			C-User5	L-User4	W-User4	F-User4	
Mixed	CS	G-User5	**0,0796**	0,0505	0,0188	0,0343	0,0458
	VS	G-User5	0,0495	0,0263	0,0111	0,0222	*0,0273*
						total CS:	**0,0436**
						total VS:	***0,0266***

Πίνακας 7 Πειραματικά αποτελέσματα από Tweets χωρίς Urls, σημεία στίξεως, αριθμούς και κενά

Στις τις τριάντα έξη συγκρίσεις που πραγματοποιήθηκαν ο μέσος όρος ομοιότητας για χρήστες που διαπραγματεύονται κοινά θέματα προέκυψε 0,0920 ενώ για χρήστες που διαπραγματεύονται διαφορετικά θέματα είναι 0,0436. Αν θεωρήσουμε οριακή τιμή CS=0,55 θα έχουμε τρεις αστοχίες για χρήστες που διαπραγματεύονται κοινά θέματα και δύο για χρήστες που διαπραγματεύονται διαφορετικά θέματα. Ο μέσος όρος επιτυχίας της μεθόδου σύγκρισης τριγραμμάτων χωρίς urls είναι 86,1%

Η βελτίωση στον μέσο όρο ομοιότητας είναι σημαντική από της αλλαγές και τις βελτιώσεις που πραγματοποιήθηκαν στα δεδομένα που αντιστοιχούν στα tweets του κάθε χρήστη. Αντίστοιχη βελτίωση υπήρξε και στο μέγεθος των γράφου που αναπαριστούν τους χρήστες με αποτέλεσμα και η απαίτηση σε υπολογιστικούς πόρους να μειωθεί αισθητά.

8.8 Αιτιολόγηση μη Αναμενόμενων Τιμών.

Πολύ σημαντικό θέμα που μας απασχολεί είναι να βρούμε για ποιον λόγο κάποιες τιμές ανεξάρτητα από τις βελτιώσεις που κάναμε πριν επιμένουν να μην προκύπτουν όπως θα αναμέναμε. Για αυτό ανατρέξαμε στα tweets του κάθε χρήστη και με την βοήθεια των δεδομένων

που δίνει ως έξοδο το πρόγραμμα μας βρήκαμε τα κοινά τριγράμματα, τις κοινές λέξεις ή την έλλειψη κοινών λέξεων. Παρακάτω θα αναφέρουμε κάθε ασυμφωνία αριθμών που είχαμε και θα αιτιολογήσουμε γιατί συνέβη. Δεν ασχολούμαστε μόνο με τις τιμές που παραβαίνουν την οριακή τιμή CS=0,55 αλλά οποιαδήποτε την προσεγγίζει.

W-User4: Ο χρήστης W-User4 έχει πολύ μικρή ομοιότητα με τους άλλους χρήστες του θέματος του γιατί αν και ασχολείται με το θέμα τον κρασιών, αναφέρει στα tweets του μάρκες κρασιών πολύ εξειδικευμένες που άλλοι χρήστες δεν τις αναφέρουν. Το περισσότερο κείμενο αποτελείται από ονόματα κρασιών τα οποία δεν τα συναντάμε να τα αναφέρουν άλλοι χρήστες. Η έλλειψη κοινών λέξεων οδηγεί σε έλλειψη κοινών γειτονικών τριγραμμάτων και αυτό κάνει την μέθοδο μας να μην μπορεί να βρεί κοινά δεδομένα για να εξάγει μια αναμενόμενα πιο ψιλή τιμή στην σύγκριση με τους άλλους χρήστες του ιδίου θέματος.

W-User2 και W-User5: Αντίστοιχη αιτιολόγηση με αυτήν που δώσαμε για τον χρήστη W-User4 έχουμε για τους χρήστες W-User2 και Wine όταν συγκρίνονται με τον χρήστη Vinopolis. Αν και ασχολούνται με το ίδιο θέμα ο καθένας αναφέρει άλλα είδη κρασιών ή με άλλες εκφράσεις θέλει να επιδοκιμάσει το κρασί που αυτό έχει ως αποτέλεσμα την έλλειψη κοινών λέξεων και τριγραμμάτων. Αναμέναμε εδώ να συναντούσαμε λέξεις όπως glass, botle, drink, tanins, grape, red wine, white wine αλλά δεν υπήρχαν.

Εμείς όσο και να βελτιώσαμε την μέθοδο προσπαθώντας να αφαιρέσουμε όλα τα τριγράμματα τα οποία δεν συνεισφέρουν ουσιαστικά στην μέθοδο μας ακόμη και αν αφαιρούσαμε συνδετικές λέξεις όπως and, therefore, while το πρόβλημα δεν θα λυνόταν.

G-User5 C-User5: Εδώ συναντάμε το αντίθετο πρόβλημα από ότι πριν. Αν και οι χρήστες ασχολούνται με διαφορετικά θέματα υπάρχουν πολλές κοινές λέξεις όπως vegetables, tomato, food,delicious, healthy.

Τα δύο θέματα τις κηπουρικής και της μαγειρικής μοιράζονται πολλές κοινές λέξεις οπότε είναι πιθανό να παρουσιάζουν υψηλή ομοιότητα αν και διαπραγματεύονται διαφορετικά θέματα.

Σε αυτό το σημείο θα σχολιάσουμε το εξής αν και η κηπουρική με την μαγειρική είναι διαφορετικά θέματα παρουσιάζουν μια μικρή σχέση. Είναι πολύ πιθανό ένας χρήστης που καλλιεργεί κάποια λαχανικά έπειτα να τον ενδιαφέρει σε ποιες συνταγές θα μπορούσε να τα χρησιμοποιήσει. Ή και το αντίστροφο ένας χρήστης που χρησιμοποιεί συστηματικά δυόσμο στην μαγειρική του θα εκπλήσσονταν με το να μάθει πόσο εύκολα θα μπορούσε να τον καλλιεργήσει στην αυλή του ή ακόμη και σε μια γλάστρα. Οπότε αυτή η μέθοδος μπορεί να παράγει γνώση και να εξάγει συμπεράσματα που δεν θα μπορούσαν άλλες μέθοδοι εύκολα.

[C-User1]-[G-User1]: Είναι χρήστες που ασχολούνται με ανόμοια θέματα ο αλγόριθμος το αποτέλεσμα που εξάγει είναι κάτω από την οριακή τιμή αλλά γατί την πλησίασε αναζητήσαμε την αίτια. Η αιτία που είχαμε αυτή την τιμή ομοιότητας είναι κατά ένα ποσοστό ίδια με την προηγούμενη περίπτωση. Η σχέση μεταξύ της μαγειρικής και της κηπουρικής. Κοινές λέξεις tomatoes, garden. Όπως και το ότι υπήρχαν κοινές συνδετικές λέξεις που θα μπορούσαν να βρεθούν σε κάθε tweet σαν το almost.

F-User3 - W-User3: Δεν ξεπερνάνε την οριακή τιμή οπότε η μέθοδος μας τους θεωρεί ανόμοιους όπως και πράγματι είναι παρόλα αυτά γιατί είχαν μια λίγο υψηλή ομοιότητα και προσεγγίζουν την οριακή τιμή τους εξετάσαμε. Οι χρήστες F-User3 - W-User3 είχαν πολλές κοινές λέξεις όπως τις great, top, day, happy, from, get, this, let, that , your. Καμία από αυτές τις λέξεις δεν προσδιορίζει κάποιο συγκεκριμένο θέμα. Κάποιες από αυτές τις λέξεις θα μπορούσαν σε μα βελτιστοποίηση της μεθόδου να αφαιρεθούν.

[C-User1] - [W-User1]: Η χρήστης[[C-User1]] που ασχολείται με την μαγειρική και ο χρήστης W-User1 που ασχολείται με τον οίνο παρουσιάζουν υψηλή ομοιότητα που ξεπερνά την οριακή τιμή. Ασχολούνται με διαφορετικά θέματα παρόλα αυτά έχουν πολλές κοινές λέξεις όπως τις learn, dinner, light, night, summer, delicious, produce, recipie. Κάποιες από αυτές δείχνουν να υπάρχει μια συγγένεια μεταξύ του θέματος της μαγειρικής με αυτό του οίνου. Πολύ πιθανόν ένας χρήστης που ασχολείται με την μαγειρική να ενδιαφερθεί και με ποιο κρασί να συνοδεύσει ένα φαγητό. Η εύρεση ομοιότητας μεταξύ αυτών των χρηστών μπορεί να αποτελεί στοιχείο ότι η μέθοδος μπορεί να ανακαλύψει κριμένες συνάφειες και σχέσεις μεταξύ των χρηστών που διαφορετικά δύσκολα θα ανακαλύπτονταν.

Πραγματικά μεταξύ ενός χρήστη που ασχολείται με την μαγειρική και σε κάποιον που ασχολείται με το κρασί υπάρχουν κοινές λέξεις και θα μπορούσε ο ένας να ενδιαφέρει τον άλλον. Υπάρχει μια μικρή σχέση που φαίνεται από τις κοινές τους λέξεις recipie, dinner, delicious. Υπάρχουν βέβαια και περιπτώσεις όπως το discover- cover που προκύπτει ομοιότητα αλλά είναι τυχαία.

G-User5 - L-User4: Οι χρήστες G-User5 και L-User4 είναι δύο χρήστες που διαπραγματεύονται εντελώς διαφορετικά θέματα παρόλα αυτά κάνανε χρήση κάποιων κοινών λέξεων που τους οδήγησε σε μια σχετικά υψηλή ομοιότητα. Ευτυχώς όμως όχι τόσο υψηλή που να ξεπερνάει την οριακή τιμή. Κοινές λέξεις που είχανε είναι οι package, useful, times, community, projects. Οι λέξεις αυτές δεν προσδιορίζουν κάποιο συγκεκριμένο θέμα.

[C-User1] - [L-User1]: Έχουν κοινές τις λέξεις gathering, complete και this weekend. Είναι εντυπωσιακό πως αν δύο λέξεις όπου η μία ακολουθεί την άλλη βρεθούν και στα δύο κείμενα η ομοιότητα ανεβαίνει πάρα πολύ. Αν αυτές οι λέξεις προσδιορίζουν κάτι από το θέμα που

διαπραγματεύονται θα είναι πολύ καλό γιατί ανεβαίνει η ομοιότητα όσον αφορά το κοινό τους θέμα. Αν όμως οι λέξεις που η μια ακολουθεί την άλλη είναι τυχαίες τότε η ομοιότητα ανεβαίνει χωρίς να υπάρχει πραγματικό κοινή θεματολογία. Όπως εδώ έγινε με το this weekend.

C-User2 - F-User5: Και εδώ όπως και στην προηγούμενη παράγραφό βρέθηκαι το this weekend. Αν υπάρχουν δύο λέξεις συνεχόμενες και στα δύο κείμενα η ομοιότητα αυξάνεται. Ευτυχώς εδώ όπως και στην προηγούμενη περίπτωση δεν αυξήθηκε τόσο ώστε να ξεπεραστεί η οριακή τιμή.

Ένα δεύτερο θέμα που αξίζει να αναφέρουμε είναι ότι στο ένα κείμενο βρέθηκε η λέξη should και στο άλλο shoulder. Είναι δύο λέξεις που δεν πηγάζουν απο την ίδια ρίζα λέξεων, δεν προσδιορίζουν κανένα κοινό θέμα παρόλα αυτά παράγοντα πολλά κοινά τριγράμματα που συνορεύουν. Αυτό είναι από τις μεγαλύτερες αδυναμίες της μεθόδου και είναι δύσκολο να ξεπεραστεί. Ευτυχώς αυτό το φαινόμενο συμβαίνει σπάνια οπότε δεν μας επηρεάζει αισθητά τα αποτελέσματα.

Από την άλλη πλευρά η ίδια μέθοδος που πριν μπορεί να ανεβάζει εσφαλμένα την ομοιότητα λόγο του ότι υπάρχουν κοινά τριγράμματα μεταξύ του should και shoulder εντοπίζει πολύ σωστά την ομοιότητα στις λέξεις grilling - grilled που βρέθηκαν επίσης στο κείμενο.

W-User3 - L-User3: Όπως και πριν συναντάμε το φαινόμενο λέξεις να αποτελούνται από κοινές ακολουθίες γραμμάτων χωρίς όμως να έχουν κάποια κοινή ρίζα ή κάποια νοηματική σχέση. Στους χρήστες W-User3 - L-User3 βρέθηκαν οι λέξεις enterprise - surprise, custom - customer που δεν έχουν κάποια νοηματική σχέση. Από την άλλη, πολλή καλώς, βρέθηκαν και οι program - programing, offerings - offers που έχουν κάποιο νοηματικό νόημα. Άλλες κοινές λέξεις που υπήρχαν ήταν οι valley και spanish.

W-User3 - G-User4: Στις δύο λέξεις preserve-reserva παρατηρούμε ξανά το φαινόμενο για το οποίο μιλήσαμε και ποιο πριν. Ανόμοιες νοηματικά λέξεις να μοιράζονται κοινές ακολουθίες γραμμάτων. Άλλες κοινές λέξεις που βρέθηκαν στο κείμενο είναι η great και beautiful.

C-User2 - G-User2: Σε αυτούς τους δύο χρήστες όπως και σε πολλούς πιο πριν αναζητήσαμε στα αρχικά tweets τους να βρούμε για ποιο λόγο προκύπτει η ομοιότητα που έχουν αν και δεν ξεπερνούσε την οριακή τιμή. Αυτό το κάναμε για να δούμε τι είδους λέξεις μπορεί να μοιράζονται δύο ανόμοιοι χρήστες. Οι κοινές λέξεις που μοιράζονται είναι το going should beautiful, ideas, italian. Το should σε μια βελτίωση της μεθόδου θα μπορούσε εύκολα να παραληφθεί. Για τις υπόλοιπες λέξεις είναι ένα ανοιχτό θέμα ποιες από αυτές και με ποιο τρόπο θα έπρεπαι να αφαιρεθούν.

Οι χρήστες W-User2, W-User4 και Wine είδαμε στις προηγούμενες συγκρίσεις ότι είχαν πολύ μικρή ομοιότητα με τον χρήστη αναφοράς στο θέματους. Μετά την αιτιολόγηση που

κάναμε σε προηγούμενους παραγράφους θελήσαμε να αναζητήσουμε, ως μια παραπάνω έρευνα τι αποτελέσματα θα είχαμε αν τους συγκρίναμε με τους υπόλοιπους χρήστες αναφοράς. Στον παρακάτω πίνακα είναι καταχωρημένα τα αποτελέσματα έπειτα από την σύγκριση

		[L-User1]	[C-User1]	[G-User1]	[F-User1]
CS	W-User2	0,0357	0,0415	0,0431	**0,0581**
VS	W-User2	0,0283	0,0341	0,0238	0,0275
CS	W-User4	0,0210	0,0276	0,0185	0,0256
VS	W-User4	0,0182	0,0228	0,0131	0,0110
CS	W-User5	0,0335	0,0409	**0,0578**	0,0423
VS	W-User5	0,0287	0,0297	0,0315	0,0187

Πίνακας 8 Περαιτέρω πειραματικά αποτελέσματα χρηστών που είχαν μικρή ομοιότητα στο θέμα τους

Από τις δώδεκα συγκρίσεις που πραγματοποιήθηκαν παρατηρούμε ότι στις περισσότερες έχουμε μικρό αποτέλεσμα ομοιότητας και μάλιστα μικρότερο από το αποτέλεσμα που είχαμε όταν συγκρίθηκαν με τον χρήστη αναφοράς του θέματός που ανήκουν. Από αυτό μπορούμε να συμπεράνουμε ότι δεν θα γινόταν εύκολα η λάθος κατάταξη τους σε μια διαφορετική θεματολογία από αυτήν που ανήκουν.

8.9 Συμπεράσματα σχόλια και πιθανές βελτιώσεις στο σύστημα συστάσεων που κάνει χρήση τριγραμμάτων.

Κάναμε μια ανασκόπηση στα κοινωνικά δίκτυα που υπάρχουν. Είδαμε τις διάφορες τεχνολογίες που χρησιμοποιούν. Εστιάσαμε στα συστήματα συστάσεων. Είδαμε ένα μεγάλο πλήθος από συστήματα συστάσεων Σχεδιάσαμε και υλοποιήσαμε το δικό μας σύστημα με χρήση των γειτονικών τριγραμμάτων. Κάναμε μια σειρά από πειράματα σε πραγματικά δεδομένα με αυτό το σύστημα Σχολιάσαμε και ανατρέξαμε στις βαθύτερες αιτίες που είχαμε αυτά τα αποτελέσματα.

Τώρα είμαστε σε θέση να κάνουμε έναν απολογισμό της μεθόδου συστάσεων που κάνει χρήση γειτονικών τριγραμμάτων να πούμε που υπερτερεί που υστερεί και πιθανές βελτιώσεις που θα μπορούσαν να γίνουν.

8.9.1 Προτερήματα της μεθόδου Συστάσεων που κάνει χρήση των Τριγραμμάτων.

Η μέθοδος Συστάσεων που κάνει χρήση τριγραμμάτων είναι μια αρκετά αποδοτική και ευέλικτη μέθοδος που μπορεί αξία να σταθεί αλλά και να ξεχωρίσει ανάμεσα στις μεθόδους συστάσεων που εφαρμόζονται στα κοινωνικά δίκτυα. Παρακάτω παραθέτουμε μια σειρά από τα δυνατά της σημεία.

- Είναι μια μέθοδος που έχει αρκετά μεγάλη ευστοχία στο να βρίσκει χρήστες που έχουν αναμεταξύ τους κοινά στοιχεία και ασχολούνται με κοινά θέματα. Η μέθοδος θα μπορούσε να αποφανθεί για οποιονδήποτε χρήστη τι σχέση έχει με οποιονδήποτε άλλον. Επίσης μπορεί να κάνει για κάθε χρήστη μια ταξινόμηση σε φθίνουσα σειρά κατά πόσο έχει κοινά ενδιαφέροντα με οποιονδήποτε άλλο. Έτσι μπορούμε να βρίσκουμε τους n πιο όμοιους.

- Η μέθοδος έχει την ικανότητα να προβάλει συστάσεις και να βρίσκει ομοιότητες ακόμη και από χρήστες που δεν έχουν δηλωθεί ως φίλοι ή με οποιαδήποτε μέθοδο subscribing.

- Μπορεί να βρεί κρυφές θεματικές σχέσεις ανάμεσα σε χρήστες που δεν δηλώνουν ξεκάθαρα ότι μπορεί να συσχετίζονται όπως είδαμε με χρήστες που ασχολούνται με το Gardening και το cooking.

- Η μέθοδος τα δεδομένα που συλλέγει και η επεξεργασία που κάνει είναι σιωπηλή δεν ζητάει από τον χρήστης να δηλώσει θεματολογίες του τι τον ενδιαφέρει ή με οποιονδήποτε τρόπο να τον παρακινήσει να κάνει κάτι διαφορετικό από την απλή χρίση του κοινωνικού δικτύου στο οποίο είναι μέρος. Αυτό είναι πολύ σημαντικό γιατί δεν καλείται ο χρήστης να κάνει ενέργειες πέρα από αυτά που κάνει συνήθως.

- Μπορεί εύκολα η μέθοδος αυτή να συνδυαστεί και με άλλες μεθόδους. πχ Να βρίσκει χρήστες που ταιριάζουν περισσότερο μέσα σε μια κατηγορία που δηλώθηκε ρητά ή που προσδιορίστηκε από κάποια άλλη μέθοδο.

- Η μέθοδος μπορεί να είναι δυναμική. Με την πάροδο του χρόνου ένας χρήστης να αλλάζει ενδιαφέροντα οπότε να αλλάζει και ο γράφος που τον αντιπροσωπεύει και έτσι να αλλάζουν οι συστάσεις που του γίνονται σύμφωνα με τα νέα του ενδιαφέροντα.

- Με την αντίστοιχη γενίκευση. Η μέθοδος είναι αρκετά ευέλικτη να συνδυαστεί όχι μόνο με tweets αλλά και σε οποιαδήποτε περίπτωση ο χρήστης συγγράφει κείμενα που τον αντιπροσωπεύουν. Θα μπορούσε να λυθεί το πρόβλημα του cold start με το να φτιάχνεται ο πρώτος γράφος που να αναπαριστά τον χρήστη από κείμενο που υπάρχει στα mail που

έχει στείλει. Αντίστοιχα θα μπορούσαμε να χρησιμοποιήσουμε τον γράφο που αναπαριστά τον χρήστη για να του προταθούν ιστοσελίδες να επισκεφτεί.

- Από την στιγμή που έχει γράψει ένας χρήστης κάποια tweets δεν είναι ανάγκη η συνεχής παρακολούθηση του για να γίνουν συστάσεις, αλλά η περιοδική. Θα μπορούσαμε ανά μία χρονική περίοδο να δειγματοληπτούμε στα tweets που έχει γράψει και να σχηματίζουμε τον καινούριο γράφο που τον αναπαριστά.

- Η επεξεργασία των δεδομένων των tweets για την εξαγωγή των συμπερασμάτων μπορεί να γίνεται οποτεδήποτε υπάρχουν διαθέσιμοι πόροι ελεύθεροι στο σύστημα μας. Θα μπορούσε το σύστημα άλλη ώρα να εξυπηρετεί τους χρήστες για τις ανάγκες του κοινωνικού δικτύου και άλλη ώρα πιθανός το βράδυ να επαναφέρει τα δεδομένα που έχουν καταχωρηθεί ώστε να εφαρμόζει τον αλγόριθμο συστάσεων και να έχει έτοιμες συστάσεις για τις επόμενες μέρες.

8.9.2 Αδυναμίες της μεθόδου συστάσεων που κάνει χρήση των τριγραμμάτων.

Δυσκολίες που καλείται να ξεπεράσει η μέθοδος συστάσεων με χρήση τριγραμμάτων καθώς και παράγοντες που μπορούν να μειώσουν την αποδοτικότητα της. Ευτυχώς άλλα από αυτά τα προβλήματα μπορούν να ξεπεραστούν και άλλα η επίδραση τους δεν είναι σοβαρή και έτσι δεν επηρεάζουν πολύ τα αποτελέσματα. Σίγουρα όμως θα πρέπει να τις αναφέρουμε για να είμαστε σε θέση να τις ξεπεράσουμε.

- Μπορεί πολλοί χρήστες να ασχολούνται με τα ίδια θέματα αλλά ο πλούτος της γλώσσας και τις έκφρασης κάνει τους ανθρώπους ο κάθε ένας να το διατυπώνει με διαφορετικό τρόπο ή χρησιμοποιεί συνώνυμες λέξεις ή τις τοποθετεί με άλλη σειρά.

- Ακόμη και το ίδιο το θέμα που ενδιαφέρει πολλούς χρήστες μπορεί να είναι τόσο πλούσιο που ο κάθε ένας να έχει ασχοληθεί με άλλο κομμάτι του. Έτσι μπορεί δύο χρήστες να μην μοιράζονται λέξεις κλειδιά αναμεταξύ τους αλλά ο ένας να ενδιαφέρεται για αυτά που λέει ο άλλος. Πχ δύο χρήστες γράφουν συνταγές μαγειρικής που έχουν εντελώς διαφορετικά υλικά. Αυτό το πρόβλημα το συναντήσαμε όταν βρίσκαμε τις ομοιότητες μεταξύ των χρηστών που ασχολούνται με τον οίνο.
  Θα μπορούσαμε να αναφέρουμε ότι αυτό είναι το μεγαλύτερο πρόβλημα που μπορεί να

προκύψει όταν δύο χρήστες ασχολούνται με το ίδιο θέμα. Άλλο παράδειγμα είναι όταν έχουμε δύο πολιτικούς ή διαφορετικές αθλητικές ομάδες. Μπορεί τα tweets που αναρτά ο κάθε ένας να έχει ελάχιστες κοινές λέξεις με τα tweets που αναρτά ο άλλος. Μπορεί ο ένας πολιτικός να αναφέρει που έκανε περιοδεία και με ποια κόμματα σκοπεύει να συνεργαστεί και ο άλλος πολιτικός να εκφράζει τις θέσεις του και τις προτάσεις του. Μπορεί μια αθλητική ομάδα να αναφέρει ποιες μεταγραφές σκοπεύει να κάνει και σχόλια για την προσωπική ζωή του κάθε αθλητή και η άλλη αθλητική ομάδα τους αγώνες που έδωσε και στρατηγικές που εφάρμοσε σε κάθε παιχνίδι.

- Υπάρχουν tweets που αναγράφουν δηλώσεις που οποιοσδήποτε χρήστης όπου ασχολείται με διαφορετικά θέματα μπορεί να τις έχει γράψει. Όπως αναφορές για το ότι βρήκε ένα αντικείμενο στο amazon ή στο ebay που τον ενδιαφέρει χωρίς όμως να δηλώνει ποιο είναι αυτό. Ή συναντήσεις που θα γίνουν κάποια ημερομηνία χωρίς να αναφέρουν το θέμα της συνάντησης. Η μέθοδος μπορεί να εντοπίσει τις κοινές εκφράσεις όπως “I will buy it” ή “attend a meeting” αλλά δεν θα μπορεί να ξεχωρίσει ότι το αντικείμενο που αναφέρεται είναι εντελώς διαφορετικό ή οτι τα θέματα των meeting είναι διαφορετικά.

- Γενικές εκφράσεις που χρησιμοποιούν οι χρήστες όπως You don't have to pay to be creative που την είπε ο gardeningguru θα μπορούσε να μπορούσε να χρησιμοποιήθει από κάποιον άλλο χρήστη που ασχολείται με οποιοδήποτε άλλο θέμα όπως gardening, cooking, painting, any art. Υπάρχουν πολλές εκφράσεις που θα μπορούσαν να γραφτούν από χρήστες που διαπραγματεύονται εντελώς διαφορετικά θέματα αυτές θα ανεβάζουν την ομοιότητα της μεθόδου χωρίς να υπάρχει ουσιαστική ομοιότητα. Ένα άλλο συνηθισμένο παράδειγμα είναι χρήστες να χαιρετάνε ανά αραιά χρονικά διαστήματα όλους τους followers τους.

- Όταν ένας καινούριος χρήστης εισέρχεται στο σύστημα μας δεν υπάρχουν tweets που να τον αντιπροσωπεύουν. Χρειάζεται ένα χρονικό διάστημα για να γράψει ένα πλήθος από tweets για να μπορέσουμε να έχουμε μια εικόνα για τον χρήστη. (cold start problem). Παρόλα αυτά έχουμε προτείνει τρόπους αντιμετώπισης αυτού του προβλήματος με το να δεχόμαστε δεδομένα, εφόσον γίνεται, από άλλες πηγές όπως τα mail που έχει γράψει και μέσω αυτών να σχηματίζεται ο πρώτος γράφος που τον αναπαριστά.

- Μιά λέξη μπορεί να έχει πολλά νοήματα. Αν η λέξη αυτή είναι και μεγάλη τότε παράγονται πολλοί κόμβοι και ακμές που θα ανεβάσουν πολλή την τιμή της ομοιότητας. Κλασικό παράδειγμα τέτοιας λέξης είναι η λέξη windows που άλλο νόημα έχει για τον προγραμματιστή λειτουργικών συστημάτων και άλλο για έναν πολιτικό μηχανικό.

- Ο περιορισμένος αριθμός χαρακτήρων που επιτρέπει το twiiter να χρησιμοποιούνται (140)

έχει κάνει κάποιους χρήστες να χρησιμοποιεί κάποιες συντμημένες εκφράσεις ή χαρακτήρες που να σημαίνουν κάποια πράγματα. Παράδειγμα τέτοιων συντμήσεων είναι το "u" που χρησιμοποιείται αντί του "you" ,το "4" που σημαίνει "for", το "w" που σημαίνει with και το "w/" που σημαίνει without. Κάποιοι χρήστες κάνουν χρήση τέτοιον εκφράσεων και κάποιοι άλλοι όχι ακόμη και αν διαπραγματεύονται το ίδιο θέμα. Βέβαια το αν κάνουν χρήση ή όχι μιας τέτοιας γραφής μπορεί να είναι ένα κριτήριο ή όχι για το κατά πόσο ταιριάζουν. Τέτοιες εκφράσεις χρησιμοποιούνται συχνότερα από νεότερους και ποιο εξοικειωμένους με τα κοινωνικά δίκτυα οπότε ήδη θα έχουμε βγάλει ένα μικρό συμπέρασμα αλλά όχι αρκετό.

8.9.3 Πιθανές μελλοντικές βελτιώσεις της μεθόδου συστάσεων που κάνει χρήση των τριγραμμάτων.

Οποιοσδήποτε προγραμματιστικό σύστημα ή μέθοδος προταθεί στην επιστήμη της πληροφορικής γνωρίζουμε καλά ότι δεν είναι κάτι στατικό. Στατικό μπορεί να παραμείνει μόνο όσο πάψει να υπάρχει ενδιαφέρον για αυτό. Αυτό οφείλεται είτε στην εφευρετικότητα του ανθρώπινου μυαλού είτε στις απαιτήσεις και στις ανάγκες του ανθρώπου που συνέχεια αυξάνονται. Για αυτό πάντα θα βρίσκονται βελτιώσεις και συνδυασμοί των μεθόδων που θα βελτιώνουν την αποδοτικότητα κάθε συστήματος.

Στο δικό μας σύστημα συστάσεων είδαμε πως από το απλώς να επεξεργαζόμαστε τα tweets έτσι όπως τα δημοσίευσαν οι χρήστες περάσαμε με μια σειρά βελτιώσεων που αύξησαν την ακρίβεια του συστήματος και ταυτόχρονα μείωσαν την απαίτηση σε επεξεργαστικούς πόρους. Παρακάτω αναφέρουμε τις βελτιώσεις που κάναμε καθώς και μελλοντικές βελτιώσεις που πιθανών θα μπορούσαν να αυξήσουν την αποτελεσματικότητα ακόμη παραπάνω.

- Η πρώτη βελτίωση που παρατηρήσαμε και κάναμε ήταν να αφαιρέσουμε τους συνδέσμους (urls) προς εξωτερικές σελίδες που συνοδεύουν πολύ συχνά το κείμενο που είναι γραμμένο μέσα στα tweets. Μια μεγάλη σειρά χαρακτήρων που αποτελούν το url είναι συνήθως άχρηστα δεδομένα για το σύστημα συστάσεων μας. Αυτοί οι χαρακτήρες δεν αντιπροσωπεύουν κάποιο λεκτικό νόημα που θα μπορούσε να παραπέμψει σε μια θεματική ενότητα. Εξαίρεση αποτελεί η περίπτωση που η διεύθυνση url έχει μέσα της τον τίτλο της ιστοσελίδας που απευθύνεται και όχι τυχαίους χαρακτήρες. Αυτή η περίπτωση δεν συμβαίνει συχνά οπότε προτιμούμε να μην σχηματίζουμε τριγράμματα - κόμβους από τα urls.

- Ένα αντίστοιχο θέμα που καλό θα ήταν να ερευνηθεί και πειραματικά προκύπτει από το

ότι πολύ συχνά στα tweets αναφέρονται ονόματα άλλων χριστών είτε ως αναδημοσίευση του tweet ενός άλλου χρήστη (retweet) είτε ως το ότι το συγκεκριμένο tweet που είναι γραμμένο απευθύνεται σε έναν συγκεκριμένο χρήστη. Όταν απευθύνεται σε κάποιον άλλον χρήστη συνήθως το όνομα του χρήστη ακολουθείτε από τον χαρακτήρα "@". Ενώ όταν είναι αναδημοσιεύσει από κάποιον άλλον χρήστη προηγούνται οι χαρακτήρες "RT". Κάποιες φορές τα ονόματα των χρηστών είναι τυχαία αλλά κάποιες άλλες φορές σχετίζονται με το θέμα διαπραγματεύονται. Εμείς επιλέξαμε να τα αφήσουμε ως έχουν όταν κάναμε τις πειραματικές μας δοκιμές.

- Λέξεις όπως and, whether, etc που καλούνται stop words συνέχεια τις συναντάμε ανεξάρτητα από το θέμα που διαπραγματευόμαστε. Αυτές οι λέξεις μπορούν να επηρεάσουν την ομοιότητα χωρίς να μπορούν να προσδιορίσουν κάτι από την θεματολογία του tweet στο οποίο βρίσκονται. Υπάρχουν συστηματικοί μέθοδοι που μπορούν να βρουν κατά πόσο μια λέξη είναι σημαντική για ένα κείμενο ούτως ώστε να την παραλείψουμε

- Μια ακόμη βελτίωση την οποία ήδη την δοκιμάσαμε με επιτυχία ήταν να αφαιρέσουμε τα σημεία στίξης. Τα σημεία στίξης εκτός από το ότι είναι ένας χαρακτήρας που δεν συμβάλει στο να προσδιοριστεί το νόημα μιας λέξης δεν χρησιμοποιούνται από όλους τους χρήστες με τον ίδιο τρόπο. Άλλοι χρήστες χρησιμοποιεί μόνο τα βασικά (τελεία, κόμμα), άλλοι χρήστες χρησιμοποιούν ποιο μεγάλη ποικιλία σημείων στίξης (θαυμαστικά, άνω τέλειες, εισαγωγικά) και κάποιοι κάνουν υπερβολική χρήση σημείων στίξεως με το να βάζουν συνεχόμενα θαυμαστικά (!!!) ή ερωτηματικά (???) ή ακόμη και να χρησιμοποιούν ascii χαρακτήρες για να κάνουν ascii art. Η ύπαρξη και μόνο ενός σημείου στίξεως μπορεί να δημιουργήσει διαφορετικούς κόμβους και ακμές. Για αυτό επιλέξαμε να τα αφαιρέσουμε και πράγματι είδαμε μια βελτίωση στα αποτελέσματα.

- Επίσης δοκιμάσαμε να μετατρέψουμε όλους τους χαρακτήρες σε μικρά γράμματα.
  Δεν υπάρχει λόγος τα ίδια τριγράμματα να αναγνωρίζονται ως διαφορετικά αν βρίσκονται στην αρχή λέξεων που η μια ξεκινά με κεφαλαίο και η άλλη με μικρό γράμμα. Πραγματικά είδαμε βελτίωση στα αποτελέσματα.

- Μεγάλη βελτίωση στα αποτελέσματα είδαμε όταν δοκιμάσαμε να σχηματίσουμε τριγράμματα χωρίς να παίρνουμε υπόψιν μας τα κενά μεταξύ των λέξεων. Τρίγραμμα μπορεί να δημιουργηθεί από το τελευταίο γράμμα μιας λέξεις τον κενό χαρακτήρα " " και το πρώτο γράμμα τις επόμενης λέξης. Επίσης μπορεί από τον κενό χαρακτήρα σε συνδυασμό με τα δύο πρώτα ή δυο τελευταία γράμματα μια λέξης. Επιλέξαμε αυτού του είδους τα τριγράμματα να τα παραλείψουμε. Τριγράμματα και αντίστοιχα κόμβους να έχουμε μόνο αυτούς που σχηματίζονται από μια λέξη. Όσον αφορά τους κλάδους

επιτρέψαμε να υπάρχουν μεταξύ τριγραμμάτων διαφορετικών λέξεων.

- Λόγω της αλγοριθμικής της πολυπλοκότητας είναι μια απαιτητική μέθοδο σε επεξεργαστική ισχύ αν θέλουμε κάθε χρήστη να τον συγκρίνουμε τους υπόλοιπους ενός συνόλου χρηστών. Για αυτό θα πρέπει πάντα να προσπαθούμε να κρατάμε τον γράφο όσο τον δυνατό μικρότερο αλλά και ταυτόχρονα να κρατάει μέσα του όσο το δυνατόν μεγαλύτερη πληροφορία από περισσότερα tweets.

- Η μέθοδος μπορεί να εφαρμοστεί για οποιαδήποτε γλώσσα με την ίδια ευκολία. Συστάσεις σε άτομα που μιλάνε την ίδια γλώσσα είναι πολύ εύκολο να γίνουν. Αν και χρήστες που γράφουν tweets σε διαφορετικές γλώσσες θα δυσκολέψουν το σύστημα και θα πρέπει να γίνουν τροποποιήσεις για να ξεπεραστεί αυτό το πρόβλημα. Μέθοδοι αναγνώρισης γλώσσας θα πρέπει να εφαρμοστούν για κάθε tweet. Έπειτα για κάθε γλώσσα που χρησιμοποιεί ο χρήστης να φτιαχτεί ένας γράφος και να συγκριθεί με τους αντίστοιχους γράφους της ίδιας γλώσσας.

- θα πρέπει να γίνει περαιτέρω έρευνα για το αν υπάρχουν κριτήρια επιλογής για το ποια tweets θα πρέπει να ενσωματωθούν στον γράφο του κάθε χρήστη και ποια όχι. Όλα τα tweets δεν εκφράζουν τον χρήστη στον ίδιο βαθμό. Κριτήρια όπως η ώρα δημοσίευσης, το πλήθος χαρακτήρων που έχει, αν είναι retweet, αν απευθύνεται σε κάποιον άλλον χρήστη μπορεί να μας προσδιορίζουν κατά πόσο η πληροφορία που μεταφέρει το tweet είναι σημαντική και εκφράζει τον χρήστη.

Βιβλιογραφία

Η βιβλιογραφία υπάρχει στο τέλος κάθε κεφαλαίου. Επίσης παραθέτουμε μια συνοπτική βιβλιογραφία των δημοσιεύσεων πάνω στις οποίες βασίστηκε η εκπόνηση της διπλωματικής.

Who do trust? Combining Recommender Systems and Social Networking for Better Advice από τον Philip Bonhard

Ubiquitous Recommendation Systems από τον David W. McDonald

Recommender Systems απο τους Paul Resnick και Hal R. Varian

Recommender Systems based on Collaborative Filtering απο τον Zheng Wen]

Recommender Systems: A GroupLens Perspective απο τους Joseph A. Konstan , John Riedl, AI Borchers και Jonathan L. Herlocker]

Content-based Recommendation Systems από τους Michael J. Pazzani και Daniel Billsus

Personalized Links Recommendation Based on Data Mining in Adaptive Educational Hypermedia Systems από τους Cristóbal Romero , Sebastián Ventura , Jose Antonio Delgado και Paul De Bra]

Jeffrey Pearsons, Paul Ralph και ηKatherine Gallagher: Using Viewing Time to Infer User Preference in Recommender Systems

Henry Kautz, Bart Selman και Mehul Shah: Referal web

Conceptual Recommender system for CiteSeer Ajith Kodakateri Susan Gauch Hiep Luong Joshua Eno

Social Media Recommendation based on People and Tags Ido Guy, Naama Zwerdling, Inbal Ronen, David Carmel, Erel Uziel

FAB: content-based, collaborative recommendation από τους Marko Balabanovic και Yoav Shoham

PHOAKS: a system for sharing recommendations. Απο τους Loren Terveen, Will Hill, Brian Amento, David McDonald και Josh Creter

Yoda: An Accurate and Scalable Web-based Recommendation Syste από τους Cyrus Shahabi, Farnoush Banaei-Kashani, Yi-Shin Chen και Dennis McLeod

Association-Rules-Based Recommender System for Personalization in Adaptive Web-Based Applications απο τους Daniel Mican και Nicolae Tomai

Short and Tweet: Experiments on Recommending Content from Information Streams απο τούς Jilin Chen , Rowan Nairn , Les Nelson , Michael Bernstein , Ed H. Chi

Introduction to the Special Issue on Summarization από τους Dragomir R. Radev και Kathleen McKeown

Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization απο την Rada Mihalcea