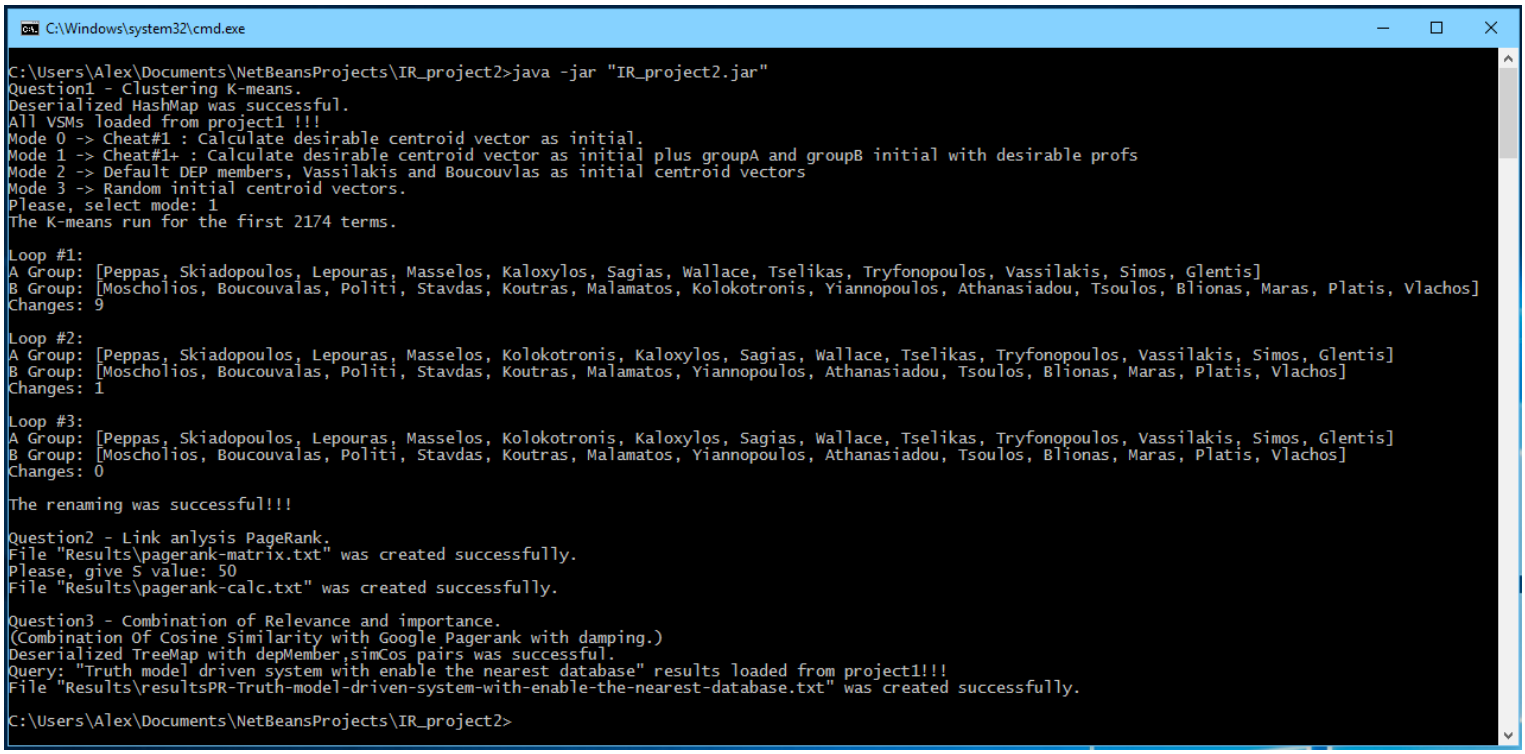
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΕΛΕΠΙΚΟΙΝΩΝΙΩΝ ΤΗΣ ΣΧΟΛΗΣ ΟΙΚΟΝΟΜΙΑΣ

Ανάκτηση Πληροφορίας

Προγραμματιστική Άσκηση 2

Ομαδοποίηση και ανάλυση συνεργασιών των μελών ΔΕΠ



ΑΛΈΞΑΝΔΡΟΣ ΠΛΈΣΣΙΑΣ

Χειμερινό εξάμηνο 2015-2016

Περιεχόμενα

Περιεχόμενα....2

Οδηγίες εκτέλεσης....3
Παραδείγματα εκτέλεσης....3
Ιεραρχία πακέτων και κλάσεων project....3
Βασικά στοιχειά υλοποίησης....4
Λεπτομέρειες υλοποίησης....4
Ανάλυση ερωτημάτων άσκησης....4

Οδηγίες εκτέλεσης

Αρχικά, ο χρήστης θα πρέπει να ανοίξει το project με ένα IDE(κατά προτίμηση NetBeans) για java και να το **εκτελέσει το project πατώντας Run είτε να εκτέλεση την εντολή *java -jar "IR_project2.jar"* μέσω τερματικού(cmd)** από τον Φάκελο του project **IR_project2** είναι πολύ σημαντικό να αναφερθεί ότι **το πρόγραμμα είναι ποιο αποδοτικό και γρήγορο όταν τρέχει από το IDE περιβάλλον**. Τότε θα του ζητηθεί να δώσει τιμή στην παράμετρο S χρειάζονται για το 2ο ερώτημα και επηρεάζει και το 3ο ερώτημα. Αφού, ο χρήστης έχει εισάγει το input τότε θα αρχίσουν να εκτελούνται όλα τα ερωτήματα και θα ενημερώνετε μέσω μηνυμάτων από την κονσόλα(console) για την πορεία εκτέλεσης τους. Τέλος, όλα τα αποτελέσματα αποθηκεύονται σε αρχεία κειμένου στον Φάκελο Results του project και το κάθε αρχείο περιέχει “header” στην πρώτη του σειρά που εξηγεί το format των αρχείων.

Παραδείγματα εκτέλεσης

Υπάρχουν στην ενότητα ανάλυση ερωτημάτων άσκησης μαζί με αναλυτικό σχολιασμό και συμπεράσματα που προκύπτουν.

Ιεραρχία πακέτων και κλάσεων project

Ακολουθεί πίνακας με τα ονόματα των πακέτων του προγράμματος και τις κλάσεις που περιέχει το κάθε πακέτο, τα ονόματα τον κλάσεων είναι αντιπροσωπευτικά της λειτουργικότητας τους. Όλες οι κλάσεις περιέχουν JavaDoc και εκτενή σχόλια σε σημεία που έχουν δύσκολη κατανόηση ώστε να υπάρχει **σαφήνεια**.

ir_project2	**Packet Files**
IR_project2(main)	FilesOfFolder
LoadVSMs	Clustering_Kmeans
LoadQueryResults	PageRank
MemberCoAuthors	RelevanceAndImportance
AllMembersCoAuthors	-

Βασικά στοιχειά υλοποίησης

Επέλεξα να **φορτώνω από την πρώτη άσκηση** τα διανύσματα όλων των μελών ΔΕΠ καθώς και την ομοιότητα συνημίτονου για συγκεκριμένα queries που έχει το κάθε μέλος, έτσι είναι ποιο **γρήγορο** το πρόγραμμα και ποιο **κομψό** το τελικό πρόγραμμα διότι το μέγεθος του παραμένει μικρο. Όλες οι κλάσεις θα μπορούσαν να υλοποιηθούν σαν μέθοδοι πράγμα όμως που εάν γινόταν θα είχε σαν αποτέλεσμα έναν πυκνό κώδικα σε μια πολύ μεγάλη main που **θα επηρέαζε την σαφήνεια αλλά και τη κομψότητα**. Επίσης στην ενότητα ανάλυση ερωτημάτων άσκησης περιγράφονται οι υλοποιήσεις του κάθε ερωτήματος και **παραλλαγές** όπου αυτές υπάρχουν.

Λεπτομέρειες υλοποίησης

Για την αποθήκευση όλων των ποσοστών επέλεξα μεταβλητές τύπου **float διότι ήθελα ακρίβεια αλλά και να μην σπαταλήσω(double διπλάσια bits) πολύ χώρο**. Σε μερικές κλάσεις **υπάρχουν μέθοδοι που τυπώνουν τα περιεχόμενα τους και θα τα βρείτε τις κλήσεις αυτών στην main σαν comments**. Σε όλα τα αρχεία αποτελεσμάτων η πρώτη σειρά μας δείχνει το format των δεδομένων. Υπάρχουν και αρκετές ακόμα λεπτομέρειες που θα αναλυθούν στις παρακάτω ενότητες ανάλογα με το σε ποιο ερώτημα εφαρμόζονται.

Ανάλυση ερωτημάτων άσκησης

Ερώτηση 1 (κλάση Clustering_Kmeans):

Αρχικά μέσω της **κλάσης LoadVSMs** φορτώνω όλα τα διανύσματα όλων των μελών ΔΕΠ από την πρώτη άσκηση, κάθε διάνυσμα περιέχει όλους τους όρους του λεξικού σε ένα TreeMap (άρα είναι ταξινομημένοι) μαζί με την βάρυνση tf-idf του κάθε όρου. Έπειτα όλα τα διανύσματα χρησιμοποιούνται από την **κλάση Clustering_Kmeans** η οποία θα εφαρμόσει την ομαδοποίηση. Για την αλγόριθμο K-means έχω **4 διαφορετικές παραλλαγές (modes)** στις οποίες υπολογίζω με διαφορετικούς τρόπους τα αρχικά κέντρα της κάθε ομάδας:

Mode 0 ("Cheat"#1) χωρίζω από μόνος μου τα μέλη ΔΕΠ σε 2 ομάδες από την μια τα μέλη που θεωρώ tety και από την άλλη τα μέλη που θεωρώ tett και έπειτα υπολογίζω τους μέσους όρους όλων των όρων τους, έτσι δημιουργώ τα 2 αρχικά κέντρα που θα πάρει ο αλγόριθμος. Έτσι καταφέρνω να συγκλίνει ο αλγόριθμος γρήγορα και με τα αποτελέσματα τα οποία προσεγγίζουν αυτό από θέλω. Η υλοποίηση υπάρχει στην μέθοδο **cheatCenters()**.

Mode 1 ("Cheat"#1+) είναι ίδιο με το mode 0 αλλά επιπλέον τα μελή της κάθε ομάδας υπάρχουν ήδη στην δομή που αποθηκεύει τα μέλη κάθε ομάδας που θέλω

έτσι στο πρώτο loop του αλγόριθμου δεν γίνονται 26 αλλαγές όπως θα ήταν αναμενόμενο αλλά λιγότερες και ο αλγόριθμος συγκλίνει ένα βήμα νωρίτερα.
Mode 2 (Default DEP members) εδώ σαν αρχικά διανύσματα κέντρων επιλέγονται αυτά του κ.Vassilakis και κ. Boucouvalas.
Mode 3 (Random initial centroid) εδώ σαν αρχικά διανύσματα κέντρων επιλέγονται 2 τυχαία διανύσματα από όλα τα μέλη ΔΕΠ και σε περίπτωση που τα 2 μέλη είναι ίδια επιλέγω τα κέντρα του Mode 2.

Ο αλγόριθμος k-means διαχωρίζει όλα τα μέλη ανάλογα με το σε ποιο από τα δυο κέντρα είναι ποιο κοντά, για τον υπολογισμό της απόστασης αυτής χρησιμοποιείτε η Ευκλείδεια απόσταση, έπειτα μετρώ τις αλλαγές που έχουν γίνει συνολικά στις ομάδες εάν δεν υπάρχουν αλλαγές τότε ο αλγόριθμος τερματίζεται αλλιώς υπολογίζονται τα νέα κέντρα με την βοήθεια των **μεθόδων calcGroupAnewCenter() και CalcGroupBnewCenter()** και υπολογίζονται και πάλι οι Ευκλείδειες αποστάσεις από τα νέα κέντρα κοκ. Αφού έχουν δημιουργηθεί οι 2 ομάδες καλείτε η **μέθοδος renameFiles()** που διαβάζει την τελευταία ουσιαστική γραμμή του κάθε αρχείου (aGroup.txt και bGroup.txt) και κοιτά εάν περιέχουν το μέλος ΔΕΠ Vassilakis ή Boucouvalas, αυτά τα 2 μέλη είναι ορόσημα για κάθε τμήμα (Vassilakis για tety και Boucouvalas για tett). Εάν ένα αρχείο περιέχει τον κ.Vassilakis μετονομάζεται σε tety.txt και αν περιέχει τον κ. Boucouvalas μετονομάζεται σε tett.txt αν και οι δύο βρίσκονται στο ίδιο αρχείο, τα αρχεία δεν αλλάζουν ονόματα.

Ο αλγόριθμος λαμβάνει υπόψιν του μόνο τους πρώτους **L** όρους του λεξικού. Έτσι είναι αναμενόμενο για τιμές μικρότερες του μεγέθους του λεξικού η ομαδοποίηση να μην είναι η επιθυμητή αυτό συμβαίνει διότι σε όλους τους υπολογισμούς λαμβάνονται υπόψιν οι πρώτοι **L** όροι του κάθε μέλους ΔΕΠ πράγμα το οποίο και δεν τους αντιπροσωπεύει. Ενώ αν σαν **L** επιλεχτεί όλο το λεξικό τότε για κάθε μέλος ΔΕΠ λαμβάνονται υπόψη όλοι οι όροι του και η ομαδοποίηση είναι σίγουρα ποιο σωστή.

Ο k-mean εξαρτάτε σε πολύ μεγάλο βαθμό από τα αρχικά διανύσματα που θα επιλεγούν σαν κέντρα, αυτό σημαίνει ότι με "λάθος" κέντρα η ομαδοποίηση θα είναι κακή, τώρα για το συγκεκριμένο ερώτημα κάνοντας πολλές προσομοιώσεις κατέληξα στο **συμπέρασμα** ότι οι δύο ομάδες θα μπορούσαν άνετα να γίνουν τρεις ώστε να έχουμε καλύτερη ομαδοποίηση.

Ερώτηση 2 (κλάση PageRank):

Με την βοήθεια της **κλάσης allMembersCoAuthors** ανοίγουμε το αρχείο κάθε μέλους ΔΕΠ και με την **κλάσης MemberCoAuthors** διαβάζονται όλα τα πεδία authors για το κάθε μέλος και δημιουργείτε ένα TreeMap με όλα τα μέλη που υπάρχουν στον φάκελο papers και για κάθε ένα από αυτά αποθηκεύετε πόσες φορές έχει συνεργαστεί, εννοείτε πως δεν λαμβάνονται υπόψη σαν συν-συγγραφείς το ίδιο το μέλος ΔΕΠ που εξετάζουμε κάθε φορά. Έπειτα η κλάση PageRank φτιάχνει τον **κανονικοποιημένο πίνακα συνδέσεων B** αξιοποιώντας τα δεδομένα της κλάσης allMembersCoAuthors μετά ακολουθεί η δημιουργία του **πίνακα R** όπου όλα του τα στοιχεία ισούνται με 1/n.

Εδώ για να προχωρήσει ο αλγόριθμος θα πρέπει πρώτα να επιλεχτεί μια τιμή για την **μεταβλητή d** δηλαδή τον συντελεστής απόσβεσης που ουσιαστικά πρόκειται για την πιθανότητα επίσκεψις μιας τυχαίας εξερχόμενης σελίδα/κόμβου από αυτόν που είμαστε. Η μεταβλητή d καθορίζει την τιμή που θα έχει η **μεταβλητή a** (a=1-d) που ουσιαστικά πρόκειται για την πιθανότητα τηλεμεταφοράς σε άλλη τυχαία σελίδα/κόμβου από αυτόν που είμαστε. Αν και η Google χρησιμοποιεί την τιμή d=0.85 εδώ **επιλέγω η τιμή να είναι 0.60** διότι δεν υπάρχουν πολλές συνεργασίες ανάμεσα στα μέλη με αποτέλεσμα μόνο 2 με 3 καθηγητές να ξεχωρίζουν, έτσι με την τιμή της απόσβεσης στο 0.60 κάνω την **τιμή της τηλεμεταφοράς 0.40** ώστε να **ευνοούνται λίγο και οι υπόλοιπη** 23 με 24 και να υπάρχει **δικαιοσύνη** χωρίς να ζημιώνονται σημαντικά οι 2 με 3 καθηγητές αφού η σειρά τους δεν αλλάζει. Αφού έχουν πλέον δοθεί τιμές στις μεταβλητές d, a και οι πίνακες B, R έχουν φτιαχτεί υπολογίζουμε τον **πίνακα G(google) που ισούται με d*B + a*R**. Τέλος, για να υπολογίσουμε την σημαντικότητα των μελών στον τμήμα τρέχουμε τον τύπο **Wk = G * W(k-1), για S φορές**, όπου το S δίνεται από το χρήστη. Να σημειωθεί ότι για τις πράξεις μεταξύ αριθμού με πίνακα ή πίνακα με πίνακα χρησιμοποιήθηκαν οι **μέθοδοι numberMultiplyArray2D , array2DMultiplyArray1D, sumTwo2DArrays**.

Τα αποτελέσματα μας δείχνουν πως **υπάρχει μια διασπορά σε 3 περιοχές** δηλαδή 2 με 3 καθηγητές που έχουν τα **υψηλά** ποσοστά, 2 με 3 καθηγητές που έχουν τα **χαμηλότερα** ποσοστά και όλοι οι υπόλοιπη στην **μέση με κοντινά ποσοστά**. Αυτό σημαίνει πως οι καθηγητές που έχουν τα υψηλά ποσοστά είναι οι ποιο έμπειρη στον τομέα τους και ενδεχομένως καλύπτουν μεγάλη γκάμα θεμάτων του αντικειμένου τους, αυτοί που έχουν τα χαμηλότερα ποσοστά είναι καθηγητές που το αντικείμενο τους μπορεί να είναι σημαντικό γενικότερα αλλά δεν ταυτίζεται με αυτό του τμήματος άμεσα/ισχυρά ενώ όλοι οι υπόλοιποι ασχολούνται με παρεμφερή αντικείμενα σε σχέση με τους υπόλοιπους που είναι στην μέση.

Ερώτηση 3 (κλάση RelevanceAndImportance):
Η **κλάσης LoadQueryResults** φορτώνει από την άσκηση 1 ένα TreeMap με τα ονόματα όλων των μελών ΔΕΠ και την ομοιότητα συνημίτονου που είχε ο καθένας τους σε σχέση με το **query "Truth model driven system with enable the nearest database"** το οποίο χρησιμοποίησα και στην άσκηση 1 και παρακάτω θα εξηγηθεί γιατί επέλεξα πάλι αυτό. Στην συνέχεια θα χρησιμοποιήσω από την κλάση PageRank ένα TreeMap με τα ονόματα όλων των μελών ΔΕΠ και την σημαντικότητα τους έπειτα από S βήματα. Αφού έχω τις παραπάνω δομές διαθέσιμες καλώ την κλάση RelevanceAndImportance για τον υπολογισμό της φόρμουλας του υπολογίζει το score που συνδυάζει σχετικότητα δηλαδή ομοιότητα συνημίτονου με σημαντικότητα δηλαδή PageRank score. Αφού όλα τα score υπολογιστούν έπειτα ταξινομούνται σε φθίνουσα σειρά και γράφονται στο αρχείο.

Για την εύρεση της φόρμουλας του **υπολογίζει το score δοκίμασα διαφορετικούς τρόπους** συνδυασμού της σχετικότητα με την σημαντικότητα όπως άθροισμα, γινόμενο, αρμονικό μέσο όρο και ζυγισμένο άθροισμα και αφού **τους δοκίμασα, τους ταξινόμησα από τον καλύτερο στον χειρότερο** και κατέληξα στην εξής σειρά: γινόμενο, αρμονικό μέσο όρο, ζυγισμένο άθροισμα και άθροισμα. Το άθροισμα ουσιαστικά επέστρεφε την σειρά που υπήρχε και στην ομοιότητα συνημίτονων, στο ζυγισμένο άθροισμα υπήρχε το πρόβλημα του τι ποσοστά να δώσω και γιατί έδωσα τα ποσοστά που έδωσα και γιατί τα ποσοστά που επέλεξα πχ είναι 70-30 και δεν είναι 69-31 άρα έχει κενά, ο αρμονικός μέσος όρος συμπεριφερόταν καλά μόνο στις πρώτες θέσεις έπειτα η σειρά χαλούσε τελείως. Άρα επέλεξα το **γινόμενο** όχι λόγο άτοπου (αν και φαίνεται έτσι) διότι είχε ναι μεν τα καλύτερα αποτελέσματα αλλά και γιατί κατ' εμέ **έχει το πλεονέκτημα** ότι όταν υπάρχει χαμηλή σημαντικότητα ή σχετικότητα το score είναι χαμηλό πράγμα που χρειάζεται όταν **έχουμε να κάνουμε με ερώτηση πάνω στα ενδιαφέροντα μελών ΔΕΠ και με την σημαντικότητα αυτών των μελών**.

Μια **ενδιαφέρουσα περίπτωση** είναι αυτή που επέλεξα να εξετάσω με το query "Truth model driven system with enable the nearest database" που στην άσκηση 1 έδινε σαν αποτελέσματα την σειρά: Koutras, Malamatos, Kaloxylos, Tsoulos, Wallace, Skiadopoulos, Vassilakis, Tryfonopoulos, Lepouras, Politi, Vlachos, Athanasiadou, Boucouvalas, Blionas, Sagias, Moscholios, Stavdas, Masselos, Yiannopoulos, Glentis, Simos, Tselikas, Kolokotronis, Maras, Platis και Peppas. Ενώ τωρα η σειρα είναι Vassilakis, Lepouras, Tsoulos, Tryfonopoulos, Skiadopoulos, Kaloxylos, Koutras, Malamatos , Wallace, Blionas, Masselos, Sagias, Boucouvalas, Stavdas, Politi, Athanasiadou, Yiannopoulos, Tselikas, Glentis, Vlachos, Moscholios, Peppas, Simos, Maras, Platis, Kolokotronis. Εδώ φαίνεται ότι η σειρά αλλάζει και πρώτα έρχονται τα μέλη που σαν αντικείμενο μελέτης έχουμε τις databases ή τα systems γιατί έχουν μεγάλη σημαντικότητα στο τμήμα μας και όχι μέλη που έχουν

σαν αντικείμενο μελέτης τους όρους τις truth και nearest που έχουν μικρότερη σημαντικότητα.