ΕΠΕΞΕΡΓΑΣΙΑ ΣΥΣΤΗΜΑΤΩΝ ΑΝΑΖΗΤΗΣΗΣ ΜΕ ΛΕΞΕΙΣ-ΚΛΕΙΔΙΑ ΣΕ ΣΧΕΣΙΑΚΕΣ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Η

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην ορισθείσα
από την Γενική Συνέλευση Ειδικής Σύνθεσης
του Τμήματος Μηχανικών Η/Υ και Πληροφορικής
Εξεταστική Επιτροπή

από τον

Λάμπρο Βατσιλίδη

ως μέρος των υποχρεώσεων για την απόκτηση του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΟ ΛΟΓΙΣΜΙΚΟ

Πανεπιστήμιο Ιωαννίνων
Φεβρουάριος 2017

ΑΦΙΕΡΩΣΗ

Στη μητέρα μου Ελευθερία και στον γιο μου Κωνσταντίνο-Ελευθέριο

Οία ηώ ω υιέ αεί ει

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω θερμά την επιβλέπουσα καθηγήτριά μου κυρία Ευαγγελία Πιτουρά για την καθοδήγηση, τη βοήθεια και τις πολύτιμες γνώσεις που μου παρείχε, καθώς και για την υπομονή, τον χρόνο και την υποστήριξη που μου προσέφερε σε όλη τη διάρκεια της εκπόνηση της μεταπτυχιακής αυτής εργασίας. Χωρίς την εμπειρία της και τις καθοδηγητικές της παρατηρήσεις θα ήταν αδύνατη η ολοκλήρωση της προσπάθειάς μου.

Επίσης θα ήθελα να ευχαριστήσω τους φίλους και συναδέλφους Κωνσταντίνο Σεμερτζίδη και Πέτρο Μανούση για το χρόνο και τις γνώσεις που μου διέθεσαν κατά τη διάρκεια της αυτής εργασίας, καθώς και τους υπόλοιπους συμφοιτητές της σχολής για την οποιαδήποτε συνεργασία και ανταλλαγή απόψεων και γνώσεων είχαμε σε όλη τη διάρκεια των σπουδών μου. Δε θα μπορούσα φυσικά να παραλείψω τους καθηγητές του τμήματος από τους οποίους αποκόμισα πολύτιμες γνώσεις κατά την παρακολούθηση των μαθημάτων τους.

Τέλος, θα ήθελα να ευχαριστήσω θερμά από το κοντινό συγγενικό μου περιβάλλον τους γονείς μου, τον αδερφό και την αδερφή μου, τη σύζυγό μου και ιδιαίτερα τον γιο μου για την υποστήριξη μου παρείχαν σε όλους τους τομείς αλλά και για την κατανόησή τους για τις ατελείωτες στιγμές που τους στέρησα καθ’ όλη τη διάρκεια των σπουδών μου.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

ΕΠΕΞΗΓΗΣΕΙΣ ΣΥΜΒΟΛΙΣΜΩΝ

Kw-κόμβος: κόμβος γράφου δεδομένων που περιέχει μια λέξη-κλειδί
Kw-πίνακας: υποσύνολο πλειάδων ενός πίνακα της βάσης δεδομένων
JNT: δίκτυο συνενωμένων πλειάδων
JNTS: δίκτυο συνενωμένων kw-πινάκων
MTJNT: δίκτυο συνενωμένων πλειάδων που πληροί τις προϋποθέσεις τελικής απάντησης

ΠΕΡΙΛΗΨΗ

Λάμπρος Βατσιλίδης του Κωνσταντίνου και της Ελευθερίας. MSc, Τμήμα Μηχανικών Η/Υ & Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Φεβρουάριος, 2017. Επεξεργασία ερωτημάτων με λέξεις-κλειδιά σε σχεσιακές βάσεις δεδομένων. Επιβλέπουσα: Ευαγγελία Πιτουρά.

Οι αναζητήσεις σε σχεσιακές βάσεις δεδομένων γίνονται συνήθως με τη χρήση ειδικά δομημένων ερωτημάτων στα οποία ο χρήστης πρέπει να γνωρίζει λεπτομερώς τα χαρακτηριστικά κάθε βάσης δεδομένων για να μπορέσει να ανακτήσει πληροφορίες. Με την πάροδο των ετών αυξάνεται ραγδαία ο όγκος πληροφορίας που αποθηκεύεται στις σχεσιακές βάσεις δεδομένων με αποτέλεσμα να αυξάνονται οι απαντήσεις που ανακτούν τα συστήματα αναζήτησης με λέξεις-κλειδιά. Με την εξέλιξη του διαδικτύου έχει επέλθει τεράστια αύξηση στο πλήθος των χρηστών που χρειάζονται πρόσβαση σε online βάσεις δεδομένων χωρίς να γνωρίζουν τα ιδιαίτερα χαρακτηριστικά τους, καθώς οι μισοί σχεδόν χρήστες του διαδικτύου πραγματοποιούν καθημερινά περισσότερες από τέσσερα δισεκατομμύρια αναζητήσεις [14][35]. Παρότι τα τελευταία χρόνια έχουν προταθεί αρκετά συστήματα αναζήτησης με λέξεις-κλειδιά σε σχεσιακές βάσεις δεδομένων, το πρόβλημα της ανάκτησης των αντιπροσωπευτικότερων απαντήσεων σε σύντομο χρονικό διάστημα από σχεσιακές βάσεις δεδομένων μεγάλου όγκου παραμένει ενεργό πεδίο ανάπτυξης στις αντίστοιχες ερευνητικές κοινότητες.

Στην εργασία αυτή ερευνούμε κάποια από τα γνωστότερα συστήματα αναζητήσεων σε σχεσιακές βάσεις δεδομένων τα οποία βασίζονται σε ερωτήματα με λέξεις-κλειδιά. Μελετάμε τα ιδιαίτερα χαρακτηριστικά τους αναδεικνύοντας τα πλεονεκτήματα και τα μειονεκτήματά τους και τα συγκρίνουμε μεταξύ τους χρησιμοποιώντας διάφορες μετρικές. Δημιουργούμε μια παραλλαγή ενός από τα συστήματα αναζήτησης προσθέτοντας την δυνατότητα να λαμβάνει υπόψιν συγκεκριμένες προτιμήσεις χρηστών για την ανάκτηση απαντήσεων με ανάλογο περιεχόμενο. Εξετάζουμε τις αλλαγές που συμβαίνουν στις επιδόσεις των

συστημάτων αναζήτησης, αλλάζοντας τις τιμές των παραμέτρων των βασικών μηχανισμών λειτουργίας τους και προτείνουμε μια μέθοδο για τη μέτρηση της ποιότητας των απαντήσεων που επιστρέφουν τα συστήματα αναζήτησης με λέξεις-κλειδιά.

ABSTRACT

Lampros Vatsilidis. MSc, Computer Science & Engineering Department, University of Ioannina, Greece. February, 2017. Keyword-based query processing on relational databases. Thesis Supervisor: Evaggelia Pitoura.

Searches in relational databases usually made using specially structured queries to which the user needs to know in detail the characteristics of each database in order to retrieve information. Over the years it has grown rapidly the amount of information stored in databases which increase the answers recover keyword search systems. With the evolution of the Internet has been a huge increase in the number of users who need access to online databases without knowing their special characteristics, as almost half of users make daily more than four billion searches [14] [35]. Although recent years have suggested several keyword search systems in relational databases, the problem of recovery of the most representative answers in a short time from large volume relational databases remains active development field in the respective research communities.

In this thesis we explore some of the best known search systems in relational databases that are based on queries with keywords. We study their special features highlighting the advantages and disadvantages and compare them using various metrics. We create a variation of one of the search systems by adding the capability to take into account specific user preferences to retrieve answers with similar content. We examine the changes occurring in the performance of search systems, changing the values of parameters of the main operating mechanisms.

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

Οι λέξεις-κλειδιά αποτελούν όρους που θεωρούνται αντιπροσωπευτικοί για το εκάστοτε θέμα που περιγράφουν. Η μέθοδος αναζήτησης με χρήση λέξεων-κλειδιών ως ερωτήματα έχει γίνει τα τελευταία χρόνια η πιο διαδεδομένη μέθοδος ανάκτησης πληροφοριών καθώς το κέρδος το οποίο επιτυγχάνεται είναι πολλαπλό. Για τους χρήστες είναι ευκολότερο να πληκτρολογούν λίγες λέξεις αντί να πρέπει να θυμούνται και να πληκτρολογούν ολόκληρες προτάσεις ενώ τα συστήματα επεξεργασίας των ερωτημάτων ολοκληρώνουν τις αναζητήσεις σχετικά γρήγορα επιστρέφοντας συνήθως αντιπροσωπευτικά αποτελέσματα.

1.1. Διατύπωση του προβλήματος

Η ανάπτυξη μεθόδων αναζήτησης με λέξεις-κλειδιά στα συστήματα σχεσιακών δεδομένων όπως είναι οι σχεσιακές βάσεις δεδομένων αποτελεί ενεργό ερευνητικό πεδίο καθώς το ποσοστό πληροφορίας που αποθηκεύεται σε σχεσιακές βάσεις δεδομένων γιγαντώνεται. Αποτέλεσμα της αύξησης αυτής είναι να υπάρχουν προβλήματα στην ανάκτηση των λίγων και αντιπροσωπευτικότερων απαντήσεων μέσα από ένα πολύ μεγάλο πλήθος απαντήσεων σε μικρό χρονικό διάστημα. Το πλήθος των απαντήσεων αυτών συνεχίζει αυξάνει όσο αυξάνεται ο όγκος των βάσεων δεδομένων με αποτέλεσμα το πρόβλημα αυτό να υφίσταται ακόμη και στα νεότερα συστήματα αναζήτησης με λέξεις-κλειδιά.

Τα τελευταία χρόνια έχουν προταθεί αρκετά συστήματα αναζήτησης με λέξεις-κλειδιά για την ανάκτηση αντιπροσωπευτικών απαντήσεων από σχεσιακές βάσεις δεδομένων πετυχαίνοντας ικανοποιητικά αποτελέσματα. Στην παρούσα εργασία εξετάζουμε κάποια από τα γνωστότερα συστήματα αναζήτησης και συγκρίνουμε την απόδοσή τους σε διάφορους τομείς. Προσπαθούμε να εντοπίσουμε τους λόγους που συμβάλλουν στο πρόβλημα που αναφέραμε και προτείνουμε κάποιες πιθανές λύσεις.

1.2. Στόχοι της εργασίας

Στην παρούσα εργασία εξετάζουμε τέσσερα από τα γνωστότερα συστήματα εκτέλεσης αναζητήσεων με λέξεις-κλειδιά σε σχεσιακές βάσεις δεδομένων. Τα συγκρίνουμε μεταξύ τους χρησιμοποιώντας ένα αρκετά μεγάλο πλήθος πραγματικών ερωτημάτων σε δυο διαφορετικές σχεσιακές βάσεις δεδομένων.

Οι βασικοί στόχοι της εργασίας είναι:

- Να αναλυθεί η συμπεριφορά των συστημάτων αναζητήσεων με λέξεις-κλειδιά σε σχεσιακές βάσεις δεδομένων.
- Να αναδειχθούν τα πλεονεκτήματα και μειονεκτήματά του κάθε συστήματος αναζήτησης.
- Να συγκριθούν τα συστήματα αναζήτησης σύμφωνα με τις επιδόσεις τους ως προς:
    - Τους χρόνους εκτέλεσης των ερωτημάτων ανεξάρτητα για την κάθε βάση δεδομένων.
    - Την τομή των απαντήσεων που επιστρέφουν μεταξύ τους.
    - Την ποιότητα των απαντήσεων που επιστρέφουν.
    - Τις αποτυχίες των απαντήσεων από το κάθε ένα.
- Η δημιουργία μιας μεθόδου για την μέτρηση της ποιότητας των απαντήσεων που επιστρέφονται για κάθε ερώτημα.
- Να βρεθούν οι συνδυασμοί των παραμέτρων σύμφωνα με τις οποίες τα συστήματα αναζήτησης έχουν τις καλύτερες επιδόσεις στην κάθε βάση δεδομένων καθώς και οι λόγοι που οδηγούν σε αυτές τις διαφορές.

- Η δημιουργία μιας παραλλαγής ενός από τα συστήματα αναζήτησης βασισμένο στις προσωπικές προτιμήσεις κάθε χρήστη και η σύγκρισή του με το αρχικό σύστημα.
- Να εξαχθούν συμπεράσματα από τη σύγκριση των συστημάτων βασισμένα σε λέξεις κλειδιά.
- Να επιβεβαιωθούν οι συμπεριφορές των συστημάτων αυτών συγκριτικά με τις πειραματικές προσεγγίσεις των αρχικών δημιουργών τους και άλλων ερευνητών του αντικειμένου.
- Να προταθούν μέθοδοι για τη βελτιστοποίηση της ποιότητας των απαντήσεων που επιστρέφουν τα συστήματα αναζήτησης.
- Να προταθούν νέες μελλοντικές προσεγγίσεις για βελτιστοποίηση των επιδόσεων των συστημάτων αναζήτησης που βασίζονται σε λέξεις-κλειδιά.

1.3. Δομή της εργασίας

Η υπόλοιπη εργασία δομείται από πέντε ακόμη κεφάλαια. Στο Κεφάλαιο 2 βλέπουμε τις μεθόδους που χρησιμοποιούν τα συστήματα αναζήτησης που εξετάζουμε για τη μοντελοποίηση των δεδομένων, των ερωτημάτων και των απαντήσεων. Βλέπουμε την απαραίτητη δομή των ερωτημάτων και ποιά χαρακτηριστικά πρέπει να έχει ένα σύνολο πλειάδες της βάσης δεδομένων για να θεωρείται απάντηση σε κάποιο ερώτημα.

Στο Κεφάλαιο 3 αναλύουμε τέσσερα από τα βασικότερα συστήματα που βασίζονται σε λέξεις-κλειδιά. Παρουσιάζουμε τα βασικά χαρακτηριστικά αλλά και τους μηχανισμούς λειτουργίας τους όπως τον τρόπο αναζήτησης, σύνθεσης και ταξινόμησης των απαντήσεων. Επιπρόσθετα δημιουργούμε την παραλλαγή ενός από αυτά τα συστήματα με την προσθήκη της δυνατότητας να επιστρέφει απαντήσεις οι οποίες βασίζονται στις προσωπικές προτιμήσεις των χρηστών.

Στο Κεφάλαιο 4 βλέπουμε κάποια στατιστικά στοιχεία των βάσεων δεδομένων που χρησιμοποιούμε για τις συγκρίσεις των συστημάτων αναζήτησης. Επίσης βλέπουμε το πλήθος των ερωτημάτων, τις πηγές ανάκτησης των ερωτημάτων που χρησιμοποιούμε για τη σύγκριση των συστημάτων αναζήτησης και τους λόγους που οδήγησαν στη χρήση των ερωτημάτων αυτών. Παρουσιάζουμε τα πειραματικά αποτελέσματα των μετρήσεων, κάνοντας σύγκριση των συστημάτων μεταξύ τους με

διάφορες μετρικές. Επίσης βλέπουμε τις αλλαγές που συμβαίνουν στις επιδόσεις των συστημάτων αναζήτησης αλλάζοντας κάποιες από τις παραμέτρους λειτουργίας τους.

Στο Κεφάλαιο 5 ανακεφαλαιώνουμε τονίζοντας τα συμπεράσματα τα οποία προκύπτουν από τις μετρήσεις. Επίσης θέτουμε νέους στόχους για μελλοντικές επεκτάσεις της εργασίας.

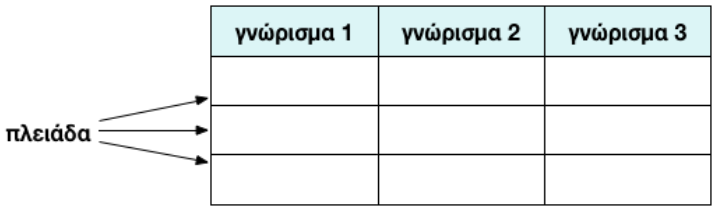
ΚΕΦΑΛΑΙΟ 2. ΜΟΝΤΕΛΟΠΟΙΗΣΗ

Στο κεφάλαιο αυτό ορίζουμε τη μοντελοποίηση των δεδομένων, των ερωτημάτων και των απαντήσεων. Συγκεκριμένα περιγράφουμε τον τρόπο αναπαράστασης μιας σχεσιακής βάσης δεδομένων σε μορφή γράφου δεδομένων ώστε να μπορεί ένα σύστημα αναζήτησης βασισμένο σε λέξεις-κλειδιά να εκτελεί αναζητήσεις στις πλειάδες της. Επίσης ορίζουμε τη βασική δομή και τα χαρακτηριστικά που χρησιμοποιούν τα συστήματα αναζήτησης που εξετάζουμε για την αναπαράσταση των ερωτημάτων που δέχονται και των απαντήσεων που επιστρέφουν. Τέλος, περιγράφουμε τα στοιχεία που καθιστούν μια απάντηση αντιπροσωπευτική προς ένα ερώτημα και τον τρόπο ταξινόμησης των απαντήσεων ανάλογα με το πόσο αντιπροσωπευτικές είναι.

2.1. Μοντελοποίηση δεδομένων

Μια σχεσιακή βάση δεδομένων είναι ένα σύνολο από πληροφορίες οργανωμένες έτσι ώστε η χρήση τους να είναι εύκολη, γρήγορη και αποτελεσματική. Η οργάνωση των πληροφοριών αυτών γίνεται σε πίνακες δυο διαστάσεων, με κάθε στήλη του πίνακα περιέχει ένα σύνολο συγκεκριμένων πεδίων που ονομάζονται *γνωρίσματα* με κάθε γνώρισμα έχει συγκεκριμένη ονομασία και μορφοποίηση ανάλογα με το είδος των πληροφοριών που αποθηκεύει. Κάθε γραμμή του πίνακα (εκτός από την επικεφαλίδα του) αποτελείται από ένα σύνολο γνωρισμάτων και ονομάζεται πλειάδα (tuple). Για να ξεχωρίζουν οι πλειάδες μεταξύ τους θα πρέπει να

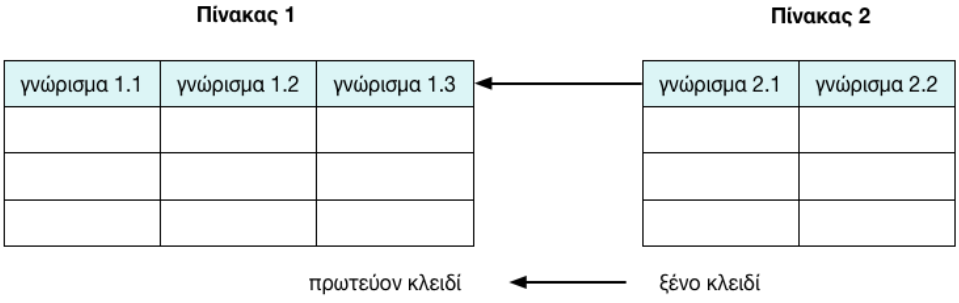
έχουν διαφορετικές τιμές τουλάχιστον σε ένα από τα γνωρίσματά τους και το γνώρισμα αυτό ονομάζεται *πρωτεύον κλειδί* του πίνακα. Στο Σχήμα 2.1 βλέπουμε ένα παράδειγμα αναπαράστασης ενός πίνακα μιας βάσης δεδομένων με τα γνωρίσματα και τις πλειάδες.



Σχήμα 2.1 Πίνακας με γνωρίσματα και πλειάδες μιας βάσης δεδομένων.

Ως *ξένο κλειδί* ονομάζουμε το γνώρισμα ενός πίνακα το οποίο αναφέρεται στο πρωτεύον κλειδί ενός άλλου πίνακα της βάσης δεδομένων. Η τιμή στις πλειάδες του γνωρίσματος ξένου κλειδιού ενός πίνακα θα εμφανίζονται σε κάποια από τις πλειάδες του γνωρίσματος πρωτεύοντος κλειδιού ενός άλλου πίνακα της βάσης δεδομένων. Στην απεικόνιση μιας βάσης δεδομένων η σχέση *πρωτεύοντος-ξένου κλειδιού* εμφανίζεται με τη μορφή μιας ακμής με κατεύθυνση από το γνώρισμα ξένου κλειδιού προς το γνώρισμα πρωτεύοντος κλειδιού του αντίστοιχου πίνακα.

Στο Σχήμα 2.2 βλέπουμε ως παράδειγμα δυο πίνακες μιας βάσης δεδομένων και μια σχέση πρωτεύοντος-ξένου κλειδιού μεταξύ δυο γνωρισμάτων τους.



Σχήμα 2.2 Απεικόνιση σχέσης πρωτεύον-ξένου κλειδιού βάσης δεδομένων.

Στη βάση δεδομένων του Σχήματος 2.2 η κατευθυνόμενη ακμή δηλώνει ότι ο πίνακας 1 έχει το γνώρισμα 1.3 ως *πρωτεύον κλειδί* και το γνώρισμα 2.1 του πίνακα 2 έχει οριστεί ως *ξένο κλειδί* του γνωρίσματος 1.3.

Σχήμα βάσης δεδομένων

Η αναπαράσταση μιας βάσης δεδομένων στη μορφή που είδαμε στο Σχήμα 2.2 ονομάζεται *σχήμα* της βάσης δεδομένων και η δημιουργία του ορίζεται ως εξής:

- Για βάση δεδομένων *R* με *n* πίνακες ($R_1, \ldots, R_n$), με κάθε πίνακα R_i να έχει m γνωρίσματα (${a_1}^i, \ldots, {a_m}^i$), θεωρούμε ως *σχήμα* της βάσης δεδομένων *R* τον *κατευθυνόμενο γράφο* $G_D(V, E)$.
- Το σύνολο *V* είναι το σύνολο των πινάκων $R_1, \ldots, R_n$ της βάσης δεδομένων *R* και δυο πίνακες R_i, R_j (με i, *j= 1,…, n*) συνδέονται με ακμή εάν ένα γνώρισμα του R_i συνδέεται με σχέση πρωτεύον-ξένου κλειδιού με ένα γνώρισμα του R_j.
- Το σύνολο *E* είναι το σύνολο των κατευθυνόμενων ακμών για τις σχέσεις πρωτεύοντος-ξένου κλειδιού μεταξύ των πινάκων. Υπάρχουν περισσότερες ακμές μεταξύ των R_i και R_j εάν υπάρχουν περισσότερα του ενός γνωρίσματα τα οποία έχουν σχέση πρωτεύον-ξένου κλειδιού μεταξύ τους και σε αυτή την περίπτωση συμβολίζονται $R_i \xrightarrow{X} R_j$ με *X* τα ονόματα των γνωρισμάτων.

Στην υπόλοιπη εργασία θα αναφερόμαστε στη *μη κατευθυνόμενη* μορφή του σχήματος G_D (χωρίς κατευθύνεις στις ακμές του) με τον συμβολισμό G_U. Στο Σχήμα 2.2 εκτός από τη σχέση πρωτεύοντος-ξένου κλειδιού βλέπουμε και το *σχήμα* της βάσης δεδομένων.

Αναπαράσταση βάσης δεδομένων σε μορφή γράφου δεδομένων

Τα συστήματα αναζήτησης που βασίζονται σε λέξεις κλειδιά δεν ανακτούν τις απαντήσεις με απευθείας αναζήτηση στη βάση δεδομένων αλλά χρησιμοποιούν μια αφαιρετική μορφή αναπαράστασης των δεδομένων της με σκοπό να γίνονται γρηγορότερα και ευκολότερα οι αναζητήσεις. Το βασικό μοντέλο αυτής της

αναπαράστασης γίνεται με τη μορφή ενός γράφου δεδομένων *G(V, E)* με το σύνολο κόμβων *V* να περιέχει πληροφορίες για τις πλειάδες της βάσης δεδομένων και το σύνολο ακμών *E* πληροφορίες για τις σχέσεις πρωτεύον-ξένου κλειδιού μεταξύ των πλειάδων αυτών. Τα μοντέλα αναπαράστασης του γράφου δεδομένων χωρίζονται σε δυο βασικές κατηγορίες ανάλογα με τον τρόπο λειτουργίας κάθε συστήματος αναζήτησης, στο *μοντέλο γράφου* και στο *μοντέλο σχήματος* και βλέπουμε τις κύριες διαφορές τους.

Το *μοντέλο γράφου* δημιουργεί τον κατευθυνόμενο γράφο δεδομένων *G(V, E)* ως εξής:

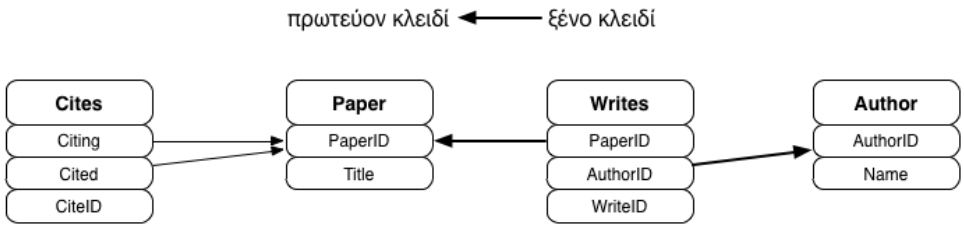
- Για κάθε πλειάδα t_i που υπάρχει στους πίνακες της βάσης δεδομένων δημιουργείται ένας κόμβος n_i στο σύνολο κόμβων *V* ο οποίος σχετίζεται με το πρωτεύον κλειδί της συγκεκριμένης πλειάδας.
- Για κάθε δυο κόμβους $u, v \in V$ δημιουργείται η κατευθυνόμενη ακμή $u \rightarrow v$ στο σύνολο ακμών *E* όταν ένα γνώρισμα της πλειάδας t_u αναφέρεται σε ένα γνώρισμα της πλειάδας t_v με σχέση πρωτεύον-ξένου κλειδιού.
- Στη συνέχεια αποδίδονται βάρη στους κόμβους και στις ακμές του γράφου δεδομένων με μεθόδους που ποικίλουν σε κάθε σύστημα αναζήτησης και βασίζονται κυρίως στο πλήθος των εισερχόμενων ακμών κάθε κόμβου.

Το *μοντέλο σχήματος* δημιουργεί τον γράφο δεδομένων ως εξής:

- Για βάση δεδομένων *R* με *n* πίνακες $R_1, \ldots, R_n$ και ερώτημα με *m* λέξεις-κλειδιά, δημιουργεί στο σύνολο κόμβων *n*m* κόμβους και κάθε κόμβος σχετίζεται μόνο με όσες πλειάδες ενός πίνακα R_j περιέχουν μια από τις *m* λέξεις-κλειδιά (*j=1,...,n*).
- Αφού δημιουργηθούν οι κόμβοι ενώνονται με άλλους κόμβους από το σύνολο *V* σύμφωνα με τη μή κατευθυνόμενη έκδοση του σχήματος της βάσης δεδομένων *Gu* δημιουργώντας δένδρα.

Η κύρια διαφορά μεταξύ των δυο μοντέλων είναι στο *μέγεθος* του γράφου δεδομένων που δημιουργούν και τα συστήματα αναζήτησης που εξετάζουμε στην παρούσα εργασία αυτή κάνουν χρήση και των δυο μοντέλων αναπαράστασης.

Σε αυτή την εργασία χρησιμοποιούμε δυο σχεσιακές βάσεις δεδομένων, μια με εγγραφές για επιστημονικά άρθρα (DBLP [43]) και μια με εγγραφές για κινηματογραφικές ταινίες (IMDB [44]). Στο Σχήμα 2.3 παίρνουμε ως παράδειγμα το σχήμα της βάσης δεδομένων DBLP από το οποίο δείχνουμε στο Σχήμα 2.4 ένα στιγμιότυπο των πλειάδων του κάθε πίνακα και στο Σχήμα 2.5 δείχνουμε την αναπαράσταση του στιγμιότυπου σε μορφή *γράφου δεδομένων* σύμφωνα με το *μοντέλο γράφου*.



Σχήμα 2.3 Το σχήμα της βάσης δεδομένων DBLP.

Paper

Paper ID	Title
p1	Publications of Hristidis
p2	Convert an XML database
p3	Match keywords in XML files
p4	Implementation of algorithms for XML files

Author

AuthorID	Name
a1	Vagelis Hristidis
a2	Yannis Papakonstantinou
a3	Kostas Hristidis

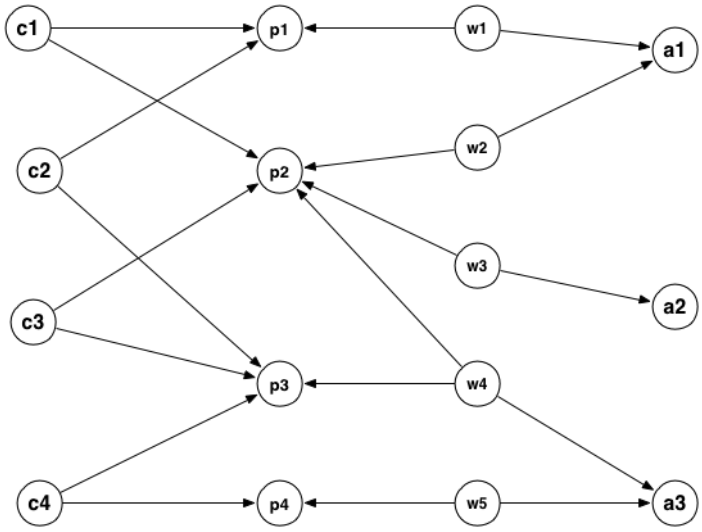
Writes

PaperID	AuthorID	WriteID
p1	a1	w1
p2	a1	w2
p2	a2	w3
p3	a3	w4
p4	a3	w5

Cites

Citing	Cited	CiteID
p2	p1	c1
p3	p1	c2
p2	p3	c3
p3	p4	c4

Σχήμα 2.4 Ένα στιγμιότυπό εγγραφών της βάσης δεδομένων DBLP.



Σχήμα 2.5 Αναπαράσταση του στιγμιότυπου της βάσης δεδομένων σε μορφή κατευθυνόμενου γράφου δεδομένων.

Βλέπουμε ότι για κάθε πλειάδα που υπάρχει στους τέσσερις πίνακες της βάσης δεδομένων (Σχήμα 2.4) δημιουργείται και ένας κόμβος στον γράφο δεδομένων στο Σχήμα 2.5 ο οποίος αντιπροσωπεύει το γνώρισμα πρωτεύοντος κλειδιού της πλειάδας αυτής. Για κάθε σχέση πρωτεύον-ξένου κλειδιού μεταξύ των γνωρισμάτων κάθε πλειάδας, δημιουργείται αντίστοιχα μια κατευθυνόμενη ακμή μεταξύ των κόμβων που σχετίζονται με τις πλειάδες αυτές (Σχήμα 2.4 και 2.5).

2.2. Μοντελοποίηση Ερωτημάτων

Ένα ερώτημα $Q = (k_1, k_2, \dots, k_m)$ αποτελείται συνολικά από *m* λέξεις-κλειδιά οι οποίες εισάγονται από τον χρήστη του συστήματος και ως *μέγεθος* ενός ερωτήματος ορίζουμε το πλήθος των λέξεων-κλειδιών που περιέχει. Οι λέξεις-κλειδιά του ερωτήματος θα πρέπει να διαχωρίζονται μεταξύ τους εισάγοντας κάποιον χαρακτήρα (κενό, κόμμα, εισαγωγικά) ο οποίος εξαρτάται από το κάθε σύστημα αναζήτησης. Όταν ένας κόμβος του γράφου δεδομένων περιέχει τουλάχιστον μια από τις ζητούμενες λέξεις-κλειδιά του ερωτήματος, καλείται «κόμβος λέξη-κλειδί» και θα αναφερόμαστε σε αυτόν στην υπόλοιπη εργασία με τη συντομογραφία «*kw-κόμβος*».

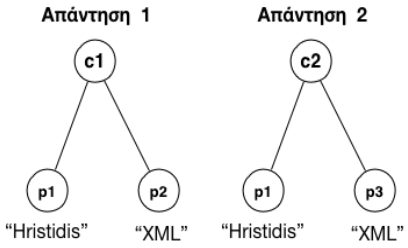
2.3. Μοντελοποίηση Απαντήσεων

Για ερώτημα με *m* λέξεις-κλειδιά $Q = (k_1, k_2, \dots, k_m)$ απάντηση θεωρείται οποιοδήποτε *συνδεδεμένο υπογράφημα* του γράφου δεδομένων που πληροί τις εξής προϋποθέσεις:

- Πρέπει κάθε λέξη-κλειδί του ερωτήματος να υπάρχει σε τουλάχιστον έναν από τους kw-κόμβους του.
- Πρέπει να είναι άκυκλο γράφημα δηλαδή να έχει δενδρική μορφή [20] και να είναι συνδεδεμένο ως ένα ενιαίο τμήμα [7].
- Πρέπει να μην υπάρχει κόμβος ο οποίος όταν αφαιρεθεί το υπογράφημα θα συνεχίσει να περιέχει όλες τις λέξεις-κλειδιά και να είναι συνδεδεμένο.
- Πρέπει να έχει έναν κοινό *κόμβο* ως *ρίζα* της απάντησης, από τον οποίο να υπάρχει μονοπάτι προς όλους τους kw-κόμβους και μπορεί να είναι και αυτός kw-κόμβος.
- Στα συστήματα που χρησιμοποιούν το μοντέλο γράφου το μονοπάτι πρέπει να έχει κατεύθυνση από τον κόμβο-ρίζα προς όλους τους kw-κόμβους της απάντησης.

Σύμφωνα με τα παραπάνω κριτήρια, στη γραφική αναπαράσταση μιας απάντησης οι kw-κόμβοι θα είναι τα φύλλα του δένδρου και ο κόμβος-ρίζα η κορυφή του. Κατά την αναζήτηση των απαντήσεων τα συστήματα αναζήτησης προσπαθούν να βρουν τις συντομότερες συνδέσεις μεταξύ των κόμβων της απάντησης και το πρόβλημα της εύρεσης του δένδρου με το ελάχιστο δυνατό κόστος ακμών που περιέχει όλους τους κόμβους ανάγεται στο πρόβλημα του steiner δένδρου. Έχει αποδειχθεί ότι ο υπολογισμός του *ελάχιστου steiner δένδρου* μέσα από ένα μεγάλο πλήθος δένδρων είναι NP-πλήρες πρόβλημα και αυτός είναι ο βασικός λόγος για τον οποίο δεν αναζητείται η μοναδική απάντηση με την καλύτερη σχετικότητα μέσα από ένα μεγάλο πλήθος απαντήσεων αλλά αναζητείται ένα πλήθος απαντήσεων με καλή σχετικότητα μεταξύ των υπολοίπων.

Στηριζόμενοι στον γράφο δεδομένων του Σχήματος 2.5 και στις εγγραφές του Σχήματος 2.4, βλέπουμε στο Σχήμα 2.6 για ερώτημα *Q=("Hristidis", "XML")* κάποιες απαντήσεις οι οποίες πληρούν τα κριτήρια που μόλις ορίσαμε:



Σχήμα 2.6 Απαντήσεις από τη βάση δεδομένων DBLP.

Όπως βλέπουμε στο Σχήμα 2.6 στην κορυφή των απαντήσεων υπάρχει ένας κόμβος-ρίζα, ως φύλλα είναι οι kw-κόμβοι οι οποίοι σχετίζονται με τις πλειάδες που περιέχουν τις ζητούμενες λέξεις-κλειδιά και πληρούν τις προϋποθέσεις των απαντήσεων. Για παράδειγμα απάντηση 2 δείχνει ότι το όνομα του συγγραφέα Hristidis υπάρχει στις αναφορές ενός paper που περιέχει στον τίτλο του τη λέξη *XML*. Η ανάκτηση αυτής της πληροφορίας γίνεται διότι δυο από τα γνωρίσματα της πλειάδας *c2* του πίνακα *Cites* σχετίζονται με σχέση πρωτεύον-ξένου κλειδιού με τα γνωρίσματα δυο πλειάδων του πίνακα *Paper*.

Τα περισσότερα συστήματα αναζήτησης με λέξεις-κλειδιά κάνουν χρήση της *top-k* απαντήσεων τεχνικής, σύμφωνα με την οποία σταματούν την αναζήτηση όταν ανακτηθούν οι πρώτες *k* απαντήσεις (*k* ακέραιος). Με αυτή την τεχνική υπάρχει η δυνατότητα να ορίζεται από τον χρήστη το πλήθος των απαντήσεων που θα ανακτώνται ανάλογα με τη βάση δεδομένων που απευθύνει τα ερωτήματα του. Συγκεκριμένα σε βάσεις δεδομένων με τεράστιο πλήθος πλειάδων πιθανόν να είναι αρκετά χρονοβόρα και μή αποδοτική η ανάκτηση πολλών απαντήσεων ταυτόχρονα.

2.3.1. Ταξινόμηση των απαντήσεων

Ως *μέγεθος* μιας απάντησης ορίζεται το πλήθος των κόμβων που περιέχει και σε σχετικές έρευνες [3, 6, 9, 20] έχει φανεί πως η σχετικότητα μιας απάντησης προς το ερώτημα είναι θετικά συνδεδεμένη με το μέγεθός της. Ο λόγος που οι ερευνητές καταλήγουν στο συμπέρασμα αυτό είναι ότι επειδή λίγοι κόμβοι συνεπάγονται σε λίγες ακμές, σε μια απάντηση με μικρό μέγεθος θα υπάρχουν μικρότερες αποστάσεις

μεταξύ των kw-κόμβων τους και μικρότερες αποστάσεις μεταξύ λέξεων-κλειδιών συνεπάγεται και σε αμεσότερες σχέσεις μεταξύ τους [1, 3, 20].

Συνήθως οι απαντήσεις που προκύπτουν από ένα ερώτημα σε μια μεγάλου όγκου σχεσιακή βάση δεδομένων είναι πάρα πολλές ενώ οι χρήστες ενδιαφέρονται για ένα μικρό πλήθος με τις σχετικότερες από αυτές. Όταν ένα σύστημα αναζήτησης με λέξεις-κλειδιά ανακτήσει τις απαντήσεις για ένα ερώτημα θα πρέπει να τις ταξινομήσει σύμφωνα με κάποια κριτήρια πριν τις εμφανίσει στον χρήστη αποδίδοντάς τους κάποιο βαθμό σχετικότητας με το ερώτημα. Έχουν αναπτυχθεί πολλές τεχνικές ταξινόμησης των απαντήσεων στα συστήματα αναζήτησης με λέξεις-κλειδιά και οι περισσότερες από αυτές αποδίδουν βαθμολογία η οποία βασίζεται στο μέγεθός τους, με σκοπό να ταξινομούνται πρώτες οι απαντήσεις που θεωρούνται αντιπροσωπευτικότερες.

Άλλες τεχνικές βαθμολόγησης απαντήσεων

Εκτός από το μέγεθος των απαντήσεων έχουν προταθεί και άλλες τεχνικές για την ταξινόμησή τους. Μια από αυτές είναι να δηλώνονται οι προσωπικές προτιμήσεις του κάθε χρήστη χρησιμοποιώντας επιπλέον συγκεκριμένες λέξεις-κλειδιά με σκοπό να προτιμώνται οι απαντήσεις που περιέχουν εκτός από τις λέξεις-κλειδιά του ερωτήματος επιπλέον και αυτές τις προτιμήσεις. Οι προτιμήσεις πρέπει να υπάρχουν στο περιεχόμενο των πλειάδων της βάσης δεδομένων και είναι ανεξάρτητες από το κάθε ερώτημα. Επίσης μπορεί να δηλωθεί μια σειρά προτεραιότητας μεταξύ των προτιμήσεών του χρήστη ώστε οι απαντήσεις που θα επιλέγονται να εμφανίζονται με την αντίστοιχη σειρά προτεραιότητας [36].

Το κέρδος με αυτή την τεχνική αναζήτησης και ταξινόμησης είναι ότι κατά τη διαδικασία αναζήτησης απορρίπτονται απαντήσεις για τις οποίες οι χρήστες δεν έχουν κάποιο πραγματικό ενδιαφέρον ενώ άλλα συστήματα αναζήτησης ίσως να εξετάσουν διεξοδικά ξοδεύοντας χρόνο και υπολογιστικούς πόρους.

2.4. Ευρετηριοποίηση

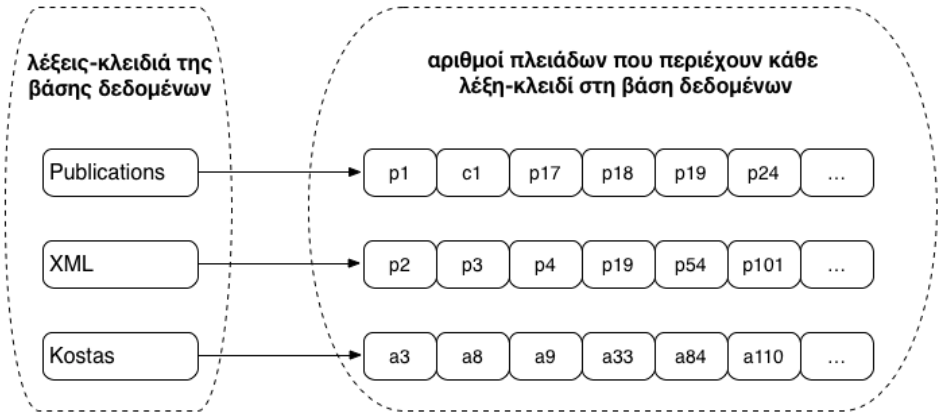
Η *ευρετηριοποίηση* είναι η διαδικασία δημιουργίας ενός συμπτυγμένου ευρετηρίου ενός συνόλου δεδομένων, με σκοπό την άμεση και γρήγορη έρευνα των εγγραφών του. Μια γνωστή τεχνική ευρετηριοποίησης είναι η δημιουργία μιας

ανεστραμμένης λίστας εγγραφών και χρησιμοποιείται από πολλά συστήματα αναζήτησης που βασίζονται σε λέξεις-κλειδιά. Τα βήματα δημιουργίας μίας τέτοιας λίστας είναι τα εξής [31]:

- Αρχικά γίνεται η συλλογή των δεδομένων που πρόκειται να εισαχθούν στο ευρετήριο και διαχωρίζονται σε αυτόνομες λέξεις-κλειδιά.
- Στη συνέχεια γίνεται γλωσσική προ-επεξεργασία των λέξεων-κλειδιών για την κανονικοποίησή τους ώστε να γίνεται ευκολότερη η αναζήτηση εξαλείφοντας ασήμαντες διαφορές μεταξύ τους όπως σημεία στίξης. Για παράδειγμα η λέξη-κλειδί «déjà vu» να μπορεί να αναζητηθεί και σαν «deja vu».
- Τέλος, γίνεται η καταγραφή των πλειάδων της βάσης δεδομένων στις οποίες υπάρχει η κάθε λέξη-κλειδί.

Στο Σχήμα 2.7 βλέπουμε ένα παράδειγμα αναπαράστασης ενός ευρετηρίου με τις λέξεις-κλειδιά κάποιων πλειάδων του πίνακα *Papers* από τη βάση δεδομένων DBLP στο οποίο συγκρατούνται δυο στοιχεία:

- Μία λίστα με τις διαφορετικές λέξεις-κλειδιά, η οποία περιέχει μια γραμμή για κάθε μια λέξη-κλειδί.
- Μια λίστα με αριθμούς που δείχνουν σε ποιά πλειάδα της βάσης δεδομένων υπάρχει η κάθε λέξη-κλειδί.



Σχήμα 2.7 Ευρετήριο με τις λέξεις-κλειδιά και τις πλειάδες που υπάρχουν στη βάση δεδομένων.

Έχουν προταθεί πολλές τεχνικές για τη χρήση του ευρετηρίου από διάφορα συστήματα αναζήτησης που βασίζονται σε λέξεις-κλειδιά. Ένας απλός τρόπος για να γίνει η αναζήτηση κάποιων συγκεκριμένων λέξεων-κλειδιών στο ευρετήριο του Σχήματος 2.7 είναι αρχικά να ελεγχθεί εάν υπάρχουν καταχωρημένες στο πεδίο «*λέξεις-κλειδιά*». Επόμενο βήμα είναι να ελεγχθεί το πεδίο «*αριθμοί πλειάδων στη βάση δεδομένων*» για να βρεθεί σε ποιες εγγραφές της βάσης δεδομένων υπάρχουν. Ένας συνηθισμένος τρόπος χρήστης του ευρετηρίου από τα συστήματα αναζήτησης με λέξεις-κλειδιά είναι να βρίσκουν σε ποιες πλειάδες της βάσης δεδομένων υπάρχουν οι λέξεις-κλειδιά του ερωτήματος και να αναζητούν τις απαντήσεις μέσω του γράφου δεδομένων με διάφορες τεχνικές.

ΚΕΦΑΛΑΙΟ 3. ΣΥΣΤΗΜΑΤΑ ΑΝΑΖΗΤΗΣΗΣ ΜΕ ΛΕΞΕΙΣ-ΚΛΕΙΔΙΑ

Σε αυτό το κεφάλαιο περιγράφουμε τα συστήματα αναζήτησης με λέξεις-κλειδιά που εξετάζουμε στην παρούσα εργασία και αναλύουμε τους βασικούς μηχανισμούς λειτουργίας τους όπως τις μεθόδους αναζήτησης, αναπαράστασης και ταξινόμησης των απαντήσεων. Επίσης βλέπουμε για το κάθε σύστημα αναζήτησης τον τρόπο αναπαράστασης της βάσης δεδομένων σε μορφή γράφου δεδομένων καθώς και των ιδιαίτερων χαρακτηριστικών του. Επιπρόσθετα δημιουργούμε μια παραλλαγή ενός από τα συστήματα αυτά με την προσθήκη να λαμβάνει υπόψιν συγκεκριμένες προτιμήσεις χρηστών για την εύρεση και ταξινόμηση των απαντήσεων.

3.1. BANKS

Το σύστημα αναζήτησης Banks [3] παρουσιάστηκε από ερευνητές του Τμήματος Μηχανικών Η/Υ και Πληροφορικής του Indian Institute of Technology της Bombay το 2002 και θεωρείται το σύστημα το οποίο έδωσε το έναυσμα για τη δημιουργία και εξέλιξη νέων συστημάτων αναζήτησης με λέξεις-κλειδιά. Για τη αναπαράσταση του γράφου δεδομένων χρησιμοποιεί την προσέγγιση του μοντέλο γράφου την οποία είδαμε στο Κεφάλαιο 2.1, σύμφωνα με το οποίο όλες οι πλειάδες στης βάσης δεδομένων μοντελοποιούνται σε κόμβους και οι σχέσεις πρωτεύοντος-

ξένου κλειδιού μεταξύ τους μετατρέπονται σε κατευθυνόμενες ακμές μεταξύ των κόμβων.

Το ευρετήριο που χρησιμοποιεί βασίζεται στη μέθοδο ευρετηριοποίησης που είδαμε στο Κεφάλαιο 2.4 το οποίο χαρτογραφεί σε ποιες πλειάδες της βάσης δεδομένων ανήκει η κάθε λέξη-κλειδί που περιέχει. Εκτός από τον γράφο δεδομένων αποθηκεύεται στην κύρια μνήμη και ένα ακόμη ευρετήριο για την αντιστοίχιση των πλειάδων του ευρετηρίου με τους κόμβους του γράφου δεδομένων.

Το σύστημα Banks δημιουργεί τον γράφο δεδομένων αποδίδοντας βάρη στις ακμές και στους κόμβους ως εξής:

- Σε κάθε κόμβο *u* που δημιουργείται στον γράφο δεδομένων αποδίδεται βάρος από τον τύπο $prestige(u) = indegree(u)$, με *indegree(u)* το πλήθος των εισερχόμενων ακμών του κόμβου *u*.
- Σε κάθε ακμή e που δημιουργείται στον γράφο δεδομένων με κατεύθυνση προς τον κόμβο *u* αποδίδεται βάρος *w(e)* ίσο με 1.
- Για κάθε εισερχόμενη ακμή προς τον κόμβο u δημιουργείται μια νέα ακμή $backward(e)$ με αντίθετη κατεύθυνση από την αρχική (αντίστροφη ακμή).

  Το βάρος της αντίστροφης ακμής αποδίδεται από τον τύπο $backward(w(e)) = prestige(u)$, δηλαδή είναι ίσο με το βάρος του κόμβου u από το οποίο ξεκινάει.

Στη συνέχεια τα αρχικά βάρη των ακμών και των κόμβων κανονικοποιούνται ως εξής:

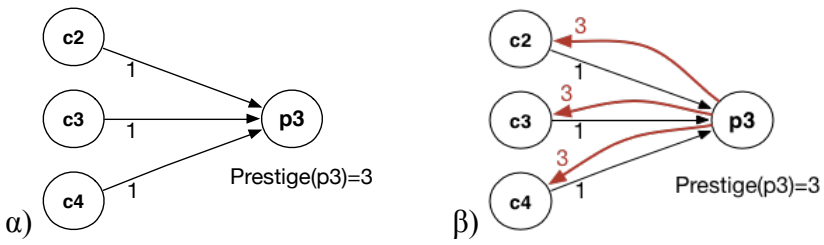
- Το αρχικό βάρος *w(e)* κάθε ακμής κανονικοποιείται σύμφωνα με τον τύπο $Escore(e) = log(1 + w(e))$, με $Escore(e)$ το τελικό βάρος της ακμής *e*.
- Το αρχικό βάρος $prestige(u)$ κάθε κόμβου *u* του γράφου δεδομένων κανονικοποιείται σύμφωνα με τον τύπο $Nscore(u) = log\left(1 + \frac{prestige(u)}{N_{max}}\right)$

με N_{max} το μεγαλύτερο βάρος κόμβου του γράφου δεδομένων και $Nscore(u)$ το τελικό βάρος του κόμβου *u*.

Το βάρος κάθε κόμβου δείχνει το πόσο δημοφιλής είναι μεταξύ των υπολοίπων και η λογική απόδοσης βαρών στους κόμβους βασίζεται στο ότι οι δημοφιλέστεροι κόμβοι είναι πιθανότερο να οδηγήσουν γρήγορα στην εύρεση μιας αντιπροσωπευτικής απάντησης επειδή συνδέονται με πολλούς άλλους κόμβους. Όταν προς έναν συγκεκριμένο κόμβο *u* υπάρχουν κατευθυνόμενες ακμές από ένα μεγάλο πλήθος κόμβων *N*, τότε στις λέξεις-κλειδιά μεταξύ των πλειάδων που σχετίζονται με τους κόμβους του συνόλου *N* είναι πιθανότερο να υπάρχουν μεγάλες αποστάσεις. Επομένως το βάρος των αντίστροφων ακμών $backward(e)$ ορίζεται για να αποδοθεί κάποιος βαθμός σχετικότητας μεταξύ του συνόλου κόμβων *N* που έχουν κατευθυνόμενη ακμή προς τον κόμβο *u*.

Στο Σχήμα 2.3 ο πίνακας "*Writes*" έχει κατευθυνόμενες ακμές προς τους πίνακες *Paper* και *Author* $[Paper] \leftarrow [Writes] \rightarrow [Author]$. Εάν για τον σχηματισμό μιας συγκεκριμένης απάντησης χρειαστεί ακμή με κατεύθυνση από τον πίνακα *Paper* προς τον πίνακα *Author* αυτό θα ήταν εφικτό μόνο εάν ο γράφος δεδομένων ήταν μη κατευθυνόμενος. Όμως αυτή η προσέγγιση θα δημιουργούσε προβλήματα στην απόδοση βαθμού σχετικότητας των απαντήσεων και για αυτό τον λόγο δημιουργούνται οι αντίστροφες ακμές μέσω των οποίων θα μπορεί να σχηματιστεί κάποια απάντηση στο ερώτημα.

Στο Σχήμα 3.1 βλέπουμε ένα παράδειγμα δημιουργίας γράφου δεδομένων στο σύστημα αναζήτησης Banks, το οποίο στηρίζεται σε ένα τμήμα του γράφου δεδομένων του Σχήματος 2.5.



Σχήμα 3.1 Blinks - Απόδοση βάρους στον κόμβο *a3* και στις αντίστροφες ακμές του.

Σύμφωνα με τη μέθοδο απόδοσης βαρών που είδαμε, στο Σχήμα 3.1 οι κόμβοι *c2, c3* και *c4* έχουν ακμή με κατεύθυνση προς τον κόμβο *p3*. Επομένως:

- Ο κόμβος p3 θα πάρει βάρος $prestige(p3) = indegree(p3) = 3$ (Σχήμα 3.1(α)).
- Για κάθε εισερχόμενη ακμή στον κόμβο *p3* θα δημιουργηθεί μια αντίστροφη ακμή με βάρος $backward(e) = indegree(p3) = 3$ (Σχήμα 3.1(β)).

Πλέον η αναζήτηση μπορεί να γίνεται και προς τις δυο κατευθύνσεις με τον τρόπο που περιγράφουμε στην *εύρεση των απαντήσεων*.

Εύρεση των απαντήσεων

Είσοδος στο σύστημα είναι ένα ερώτημα $Q = (\mathrm{k}_1, \ldots, \mathrm{k}_m)$ με *m* λέξεις-κλειδιά και έξοδος είναι οι *top-k* απαντήσεις.

Για να θεωρείται απάντηση κάθε υπογράφημα του γράφου δεδομένων θα πρέπει να πληροί τις προϋποθέσεις των απαντήσεων που τέθηκαν στο Κεφάλαιο 2.3.

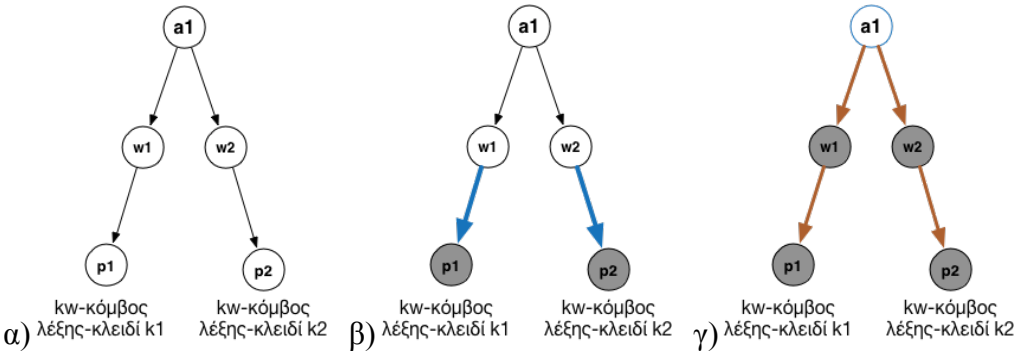
Για την εύρεση των απαντήσεων γίνονται οι παρακάτω λειτουργίες:

- Δημιουργείται το ευρετήριο και ο γράφος δεδομένων με τον τρόπο που είδαμε στο Σχήμα 2.7 και 3.1.
- Για κάθε ζητούμενη λέξη-κλειδί k_i του ερωτήματος (για *i=1,...,m*) βρίσκονται μέσω του ευρετηρίου οι πλειάδες της βάσης δεδομένων που την περιέχουν και δημιουργούνται τα σύνολα S_i των κόμβων του γράφου δεδομένων που σχετίζονται με τις πλειάδες αυτές οι οποίοι θεωρούνται kw-κόμβοι.
- Δημιουργούνται συνολικά $S = \cup\, \mathrm{S}_i$ kw-κόμβοι και αντίστοιχα *S* αντίγραφα του Dijkstra Single Source Shortest Path (Dijkstra SSSP) αλγορίθμου [13].
- Κάθε αντίγραφό του Dijkstra SSSP διασχίζει τον γράφο δεδομένων σε αντίθετη κατεύθυνση από αυτή που έχουν οι ακμές του (ένα από τον κάθε kw-κόμβο) για να βρει τα συντομότερα μονοπάτια από έναν kw-κόμβο προς έναν κόμβο-ρίζα αντιστοιχίζοντας έναν iterator [46] για κάθε αντίγραφο.

- Κατά την εύρεση των συντομότερων μονοπατιών από τον Dijkstra SSSP κάθε φορά που εξετάζεται κάποιος κόμβος για πρώτη φορά ελέγχεται εάν αποτελεί τη ρίζα κάποιας απάντησης εξετάζοντας εάν έχει κατευθυνόμενο μονοπάτι προς τουλάχιστον έναν kw-κόμβο από κάθε σύνολο S_i σύμφωνα με την κανονική κατεύθυνση των ακμών.
- Κάθε φορά που βρίσκεται ένας κόμβος-ρίζα το υπογράφημα στο οποίο συμμετέχει θεωρείται απάντηση στο ερώτημα και εφόσον δεν έχει ξαναβρεθεί η ίδια απάντηση προστίθεται στη λίστα απαντήσεων.
- Όταν βρεθούν οι top-k απαντήσεις ταξινομούνται σύμφωνα με το βαθμό σχετικότητας που τους αποδίδεται από τη *συνάρτηση απόδοσης βαθμολογίας* την οποία βλέπουμε στην επόμενη παράγραφο.
- Τέλος, αναζητούνται στη βάση δεδομένων με τη σειρά ταξινόμησης τα δένδρα των πλειάδων που σχετίζονται με τους kw-κόμβους που βρέθηκαν και τα αποτελέσματα επιστρέφονται στον χρήστη.

Για N το πλήθος κόμβων του γράφου δεδομένων και N iterators (έναν για κάθε αντίγραφο του Dijkstra SSSP), το ευρετήριο θα χρειάζεται χρόνο για τη δημιουργία του στη χειρότερη περίπτωση $O(N^2)$. Στην πραγματικότητα όμως κάθε κόμβος του γράφου δεδομένων δε συνδέεται με κάθε άλλον και επομένως είναι αρκετά μικρότερος ο χρόνος από την χειρότερη περίπτωση.

Στο Σχήμα 3.2 βλέπουμε ένα παράδειγμα αναζήτησης και εύρεσης μιας απάντησης, το οποίο βασίζεται σε ένα τμήμα του Σχήματος 2.5.



Σχήμα 3.2 Αναζήτηση απάντησης στο σύστημα Banks.

Στο Σχήμα 3.2.α βλέπουμε ότι για ερώτημα *Q=(k1, k2)* αρχικά εξετάζονται οι kw-κόμβοι *p1* και *p2* εάν αποτελούν τη ρίζα μιας απάντησης. Στη συνέχεια η αναζήτηση κατευθύνεται σε αντίστροφη κατεύθυνση και φτάνει στους κόμβους *w1* και *w2* αντίστοιχα ελέγχοντάς τους πάλι με τη σειρά που εξετάστηκαν για το αν αποτελούν κόμβο-ρίζα (Σχήμα 3.2.β). Δεν βρίσκεται απάντηση και αναζήτηση κατευθύνεται ξανά αντίστροφα στον κόμβο *a1* από τον οποίο βρίσκεται το μονοπάτι προς έναν kw-κόμβο από κάθε ζητούμενη λέξη-κλειδί και επομένως βρίσκεται μια απάντηση στο ερώτημα *Q* (Σχήμα 3.2.γ).

Ταξινόμηση των απαντήσεων:
Για την ταξινόμηση της κάθε απάντησης χρησιμοποιούνται το τελικό βάρος των κόμβων της απάντησης (*Nscore*) και το συνολικό βάρος των ακμών της απάντησης (*Escore*) ως εξής:

- Το τελικό βάρος των κόμβων *Nscore* ισούται με τον μέσο όρο των βαρών των kw-κόμβων και του κόμβου-ρίζας της απάντησης. Ορίζεται από τον τύπο $Nscore = \frac{prestige(u)+\sum_1^n prestige\left(kw(u)\right)_n}{n+1}$, με $prestige(u)$ το βάρος του κόμβου-ρίζα *u*, $prestige(kw(u))$ το βάρος του kw-κόμβου *kw(u)* και *n* το πλήθος των kw-κόμβων.
- *Escore:* Για $Escore(e)$ το βάρος κάθε ακμής e που υπάρχει σε κάθε απάντηση, το συνολικό βάρος των ακμών μιας απάντησης προκύπτει από τον τύπο $Escore = \frac{1}{(1+\sum_e Escore(e))}$. Ο τύπος *Escore* αποδίδει χαμηλότερο συνολικό βάρος ακμών σε απαντήσεις με πολλές ακμές.

Επίσης λαμβάνεται υπόψιν ο *παράγοντας βαρύτητας* λ ο οποίος δέχεται τιμές στο εύρος $[0-1]$ και καθορίζει το ποσοστό που θα ληφθεί υπόψιν το *Nscore* και το *Escore* στη συνάρτηση απόδοσης βαθμολογίας.

Συνάρτηση απόδοσης βαθμολογίας: Κάθε απάντηση που δημιουργείται βαθμολογείται από τον συνδυασμό των *Escore* και *Nscore* μέσω του *παράγοντα βαρύτητας* λ και η τελική βαθμολογία της υπολογίζεται από τον τύπο

$$TotalScore = ((1 - \lambda) * Escore) + (\lambda * Nscore)$$

Η συνάρτηση απόδοσης βαθμολογίας του συστήματος Banks λαμβάνει υπόψιν κατά ένα ποσοστό της σημαντικότητα των ακμών και των κόμβων της ανάλογα με την τιμή του παράγοντα βαρύτητας λ.

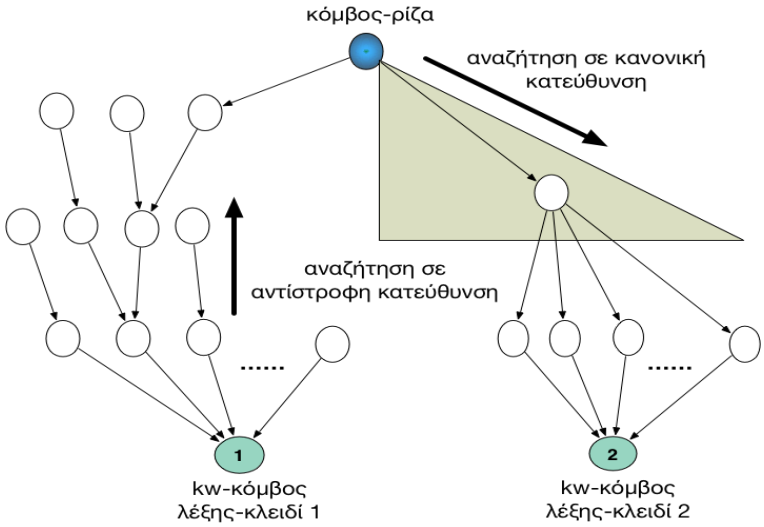
3.2. BIDIRECTIONAL

Το σύστημα αναζήτησης Bidirectional [23] παρουσιάστηκε από ερευνητές του Τμήματος Μηχανικών Η/Υ και Πληροφορικής του Indian Institute of Technology της Bombay το έτος 2005, πολλοί από τους οποίους κάποιοι συμμετείχαν και στην ανάπτυξη του συστήματος Banks [3]. Χαρακτηρίζεται ως η εξέλιξη του συστήματος Banks επειδή βασίζει σε αυτόν ένα τμήμα της λειτουργίας του, εκτελώντας αναζητήσεις σύμφωνα με την αντίθετη και την κανονική κατεύθυνση των ακμών για να βρει τον κόμβο-ρίζα μιας απάντησης. Οι λόγοι που οδήγησαν στην δημιουργία του Bidirectional είναι η ανάγκη για μείωση των υπολογιστικών πόρων που χρησιμοποιούνται κατά την αναζήτηση από το σύστημα Banks. Η κεντρική ιδέα για την εύρεση των απαντήσεων βασίζεται στο ότι οι δημοφιλέστερες απαντήσεις θα πρέπει να περιέχουν τους δημοφιλέστερους κόμβους του γράφου δεδομένων.

Στο σύστημα αναζήτησης Banks χρησιμοποιείται ένας iterator [36] για κάθε αντίγραφο του Dijkstra SSSP (έναν για κάθε kw-κόμβο). Για τόσα αντίγραφα του Dijkstra SSSP όσα και τα σύνολα kw-κόμβων, θα παραχθεί ένα μεγάλο πλήθος από iterators με αποτέλεσμα να χρειαστούν πολλοί υπολογιστικοί πόροι. Η λύση που εφαρμόζεται στο σύστημα αναζήτησης Bidirectional είναι να ταξινομούνται οι κόμβοι αποδίδοντάς τους έναν βαθμό δημοτικότητας τον οποίο βλέπουμε αναλυτικά στον τρόπο αναζήτησης των απαντήσεων. Με αυτή τη μέθοδο οι κόμβοι εξετάζονται με τη σειρά δημοτικότητάς τους, στοχεύοντας στις περισσότερες πιθανότητες που δίνει ένας αρκετά δημοφιλής κόμβος να οδηγήσει στην εύρεση μιας απάντησης νωρίτερα. Δίνεται έτσι η δυνατότητα να ελέγχεται εάν κάποιος κόμβος αποτελεί τη

ρίζα μιας απάντησης χωρίς να πρέπει να ελεγχθούν όλοι οι γειτονικοί του κόμβοι στον γράφο δεδομένων και επομένως η εμπρόσθια αναζήτηση μπορεί να γίνεται γρηγορότερα σε σχέση με τον Banks.

Για την αναπαράσταση του γράφου δεδομένων, την ευρετηριοποίηση και την αναπαράσταση των απαντήσεων χρησιμοποιεί την ίδια μέθοδο που χρησιμοποιεί και το σύστημα Banks στο Κεφάλαιο 3.1. Στο Σχήμα 3.3 βλέπουμε μια περίπτωση για αναζήτηση σε ερώτημα με δυο λέξεις-κλειδιά και έναν κόμβο-ρίζα.



Σχήμα 3.3 Περίπτωση ανάγκης αναζήτησης στην κανονική κατεύθυνση των ακμών.

Με βάση το Σχήμα 3.3 ο τρόπος αναζήτησης του συστήματος Banks θα δαπανήσει αρκετό χρόνο και πολλούς υπολογιστικούς πόρους επειδή θα ξεκινήσει την αναζήτηση από τον kw-κόμβο 1 και kw-κόμβο 2 και θα πρέπει πρώτα να εξετάσει σχεδόν όλους τους ενδιάμεσους κόμβους μέχρι να εντοπίσει τον κόμβο-ρίζα ώστε να σχηματιστεί μια απάντηση. Το πρόβλημα των αυξημένων πόρων μεγαλώνει όσο αυξάνονται οι κόμβοι ανάμεσα στις λέξεις-κλειδιά και στον κόμβο-ρίζα ή όσο αυξάνονται οι kw-κόμβοι για κάθε λέξη-κλειδί. Με τη μέθοδο αναζήτησης του συστήματος Bidirectional είναι πιθανότερο η αναζήτηση να φτάσει νωρίτερα στον κόμβο-ρίζα και να ανακαλύψει νωρίτερα την ίδια απάντηση.

Ταξινόμηση κόμβων

Το σύστημα αναζήτησης Bidirectional χρησιμοποιεί συνολικά δυο iterators, τους *incoming* και *outgoing,* με τον *incoming* iterator να χρησιμεύει για την αντίστροφη αναζήτηση η οποία ξεκινά από τους kw-κόμβους. Ο *outgoing* iterator χρησιμεύει για την αναζήτηση σύμφωνα με την κανονική κατεύθυνση των ακμών καθώς οι κόμβοι που περιέχει εξετάζονται για το αν αποτελούν τη ρίζα μιας απάντησης. Με αυτή την τεχνική μειώνεται το μέγεθος των απαιτούμενων υπολογιστικών πόρων καθώς κατά την αναζήτηση επιλέγεται κάθε φορά για εξέταση ο δημοφιλέστερος κόμβος ανάμεσα στους δυο iterators.

Πριν ξεκινήσει η διαδικασία αναζήτησης των απαντήσεων, οι κόμβοι ταξινομούνται στον incoming iterator σύμφωνα με την προτεραιότητα που τους αποδίδεται από τη διαδικασία βαθμολόγησης της δημοτικότητάς τους, ξεκινώντας από τους kw-κόμβους ως εξής:

- Υποθέτουμε ερώτημα *Q=(k1,...,km)* με *m* λέξεις-κλειδιά, S_i το σύνολο των kw-κόβμων της λέξης-κλειδί *i* και kw_i κάθε kw-κόμβο του συνόλου S_i (για *i=1,...,m*).
- Ο βαθμός δημοτικότητας του kw_i προκύπτει από τον τύπο $a_{kwi} = \frac{prestige\,(a_{kwi})}{|S_i|}$, με το $prestige(kw_i)$ το πλήθος των εισερχόμενων ακμών του κόμβου και $|S_i|$ το πλήθος των kw-κόμβων του συνόλου S_i.

Όταν επιλεγεί για πρώτη φορά κάποιος kw-κόμβος για να συνεχιστεί η αναζήτηση από αυτόν, τότε μεταδίδει τον βαθμό δημοτικότητας που έχει στους γειτονικούς του κόμβους σύμφωνα με τον *παράγοντα διάδοσης μ* ως εξής:
Για *παράγοντα διάδοσης μ* με τιμές στο πεδίο [0-1] και a_{kwi} το βαθμό δημοτικότητας του kw-κόμβου kw_i:

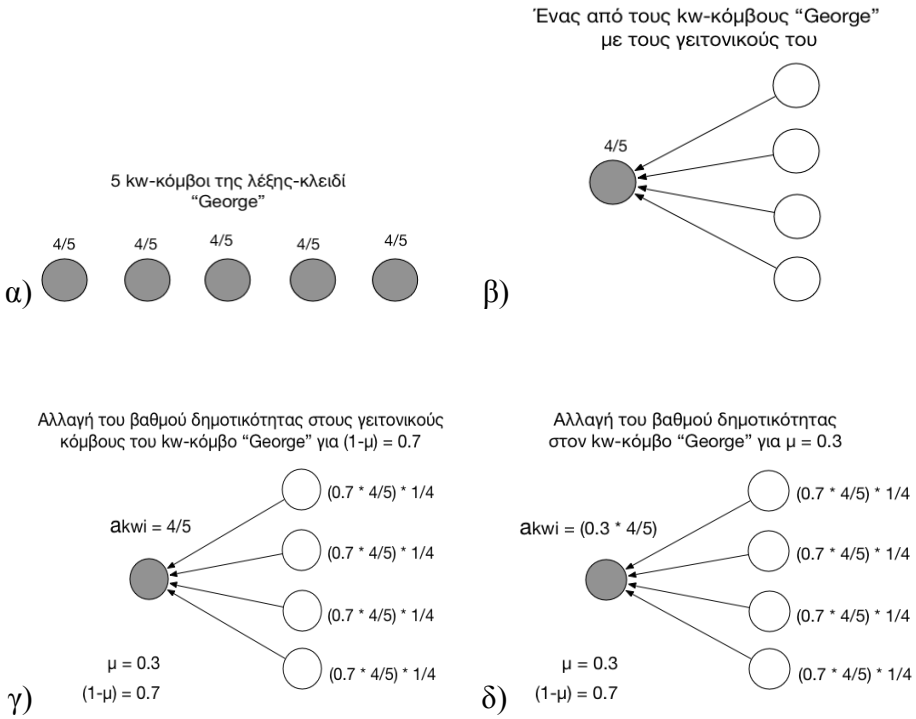
- Μεταδίδεται στους γειτονικούς του κόμβους το ποσοστό *μ* από το βαθμό a_{kwi} του kw_i και στη συνέχεια το αρχικό a_{kwi} μειώνεται σε ποσοστό *1-μ*.

Μέχρι να έρθει η σειρά ενός απλού κόμβου *u* (όχι kw-κόμβου) να εξεταστεί ενδέχεται να λάβει περισσότερους του ενός βαθμούς δημοτικότητας οι οποίοι προέρχονται από kw-κόμβους διαφορετικών συνόλων *Si* (διαφορετικής λέξης-κλειδί).

- Ο συνολικός βαθμός δημοτικότητας ενός απλού κόμβου *u* σύμφωνα με τον οποίο θα ταξινομηθεί μεταξύ των υπολοίπων κόμβων ισούται με το σύνολο όλων των βαθμών δημοτικότητας που λαμβάνει και προέρχεται από kw-κόμβο ξεχωριστού σύνολο *Si*.
- Εάν ο κόμβος *u* λάβει βαθμό δημοτικότητας από άλλον kw-κόμβο του ίδιου συνόλου *Si*, τότε κρατάει τον μεγαλύτερο βαθμό από τους δυο.

Επομένως κάποιος απλός κόμβος (όχι kw-κόμβος) που βρίσκεται κοντά σε πολλούς kw-κόμβους είναι πιθανό να αποτελέσει τη ρίζα μιας απάντησης καθώς θα ελεγχθεί γρήγορα διότι θα ταξινομηθεί σε υψηλή θέση έναντι άλλων απλών κόμβων, επειδή θα λάβει βαθμούς δημοτικότητας από πολλούς kw-κόμβους διαφορετικών συνόλων *Si*. Η μετάδοση του βαθμού δημοτικότητας συνεχίζεται στους υπόλοιπους κόμβους μέχρι να ολοκληρωθεί η αναζήτηση.

Στο Σχήμα 3.4 παίρνουμε ως παράδειγμα ένα τμήμα της βάσης δεδομένων κινηματογραφικών ταινιών IMDB και βλέπουμε ένα παράδειγμα απόδοσης βαθμού δημοτικότητας στους kw-κόμβους καθώς και της μετάδοσης του βαθμού δημοτικότητας ενός kw-κόμβου στους γειτονικούς του κόμβους.



Σχήμα 3.4 Απόδοση και μετάδοση του βαθμού δημοτικότητας μεταξύ των κόμβων.

Αρχικά υποθέτουμε ότι υπάρχουν 5 kw-κόμβοι για τη λέξη-κλειδί "George" και κάθε ένας έχει 4 εισερχόμενες ακμές (Σχήμα 3.3.α και Σχήμα 3.3.β). Επομένως κάθε kw-κόμβος αρχικά θα πάρει βαθμό δημοτικότητας $a_{kwi} = \frac{prestige\,(a_{kwi})}{|S_i|} = \frac{4}{5}$.

Με βαθμό διάδοσης μ=0.3 μεταδίδεται στους γειτονικούς κόμβους το ποσοστό *(1-μ)=0.7* του αρχικού βαθμού δημοτικότητάς a_{kwi} του kw-κόμβου "George" το οποίο οι γειτονικοί κόμβοι μοιράζονται ανάλογα με το πλήθος τους. Δηλαδή για 4 γειτονικούς κόμβους θα πάρει ο κάθε ένας τους από ¼ του βαθμού δημοτικότητας που μεταδίδει ο kw-κόμβος a_{kwi} (Σχήμα 3.3.γ).

Στη συνέχεια μειώνεται ο βαθμός δημοτικότητας του kw-κόμβου "George" στο ποσοστό *μ=0.3* της αρχικής του τιμής και οι κόμβοι ταξινομούνται ανάλογα με το βαθμό δημοτικότητας που έχουν (Σχήμα 3.3.δ).

Αναζήτηση απαντήσεων

Ως είσοδος στο σύστημα είναι ένα ερώτημα $Q = (k_1, \ldots, k_m)$ με *m* λέξεις-κλειδιά και έξοδος από το σύστημα είναι οι *top-k* απαντήσεις.

Τα βήματα αναζήτησης των απαντήσεων είναι τα ακόλουθα:

- Αρχικά δημιουργείται το ευρετήριο και ο γράφος δεδομένων και βρίσκονται τα σύνολα S_i των kw-κόμβων με την ίδια μέθοδο που είδαμε στο σύστημα Banks στο Κεφάλαιο 3.1. Επομένως θα δημιουργηθεί συνολικά $S = \cup S_i$ πλήθος kw-κόμβων.
- Όλοι οι kw-κόμβοι ταξινομούνται στον incoming iterator σύμφωνα με τον μηχανισμό απόδοσης δημοτικότητας και όσο οι δυο iterators δεν είναι άδειοι επιλέγεται μεταξύ τους ο κόμβος με τον καλύτερο βαθμό δημοτικότητας για να συνεχιστεί η αναζήτηση από αυτόν.
- Εάν επιλεγεί κόμβος από τον *incoming iterator* τότε μεταδίδει ένα ποσοστό του βαθμού δημοτικότητάς του στους γειτονικούς του κόμβους σύμφωνα με τον παράγοντα διάδοσης μ και οι γειτονικοί του κόμβοι ταξινομούνται στον

incoming iterator ανάλογα με βαθμό δημοτικότητας που θα λάβουν. Κάθε φορά που επιλέγεται κόμβος από τον *incoming iterator* επαναλαμβάνεται η διαδικασία μετάδοσης του βαθμού δημοτικότητας στους γειτονικούς κόμβους.

- Οι κόμβοι που εξετάζονται σε αντίστροφη κατεύθυνση από τον *incoming iterator* στη συνέχεια εισάγονται στον *outgoing iterator* μειώνοντας το βαθμό δημοτικότητάς τους σύμφωνα με τον παράγοντα διάδοσης μ.
- Εάν επιλεγεί κόμβος από τον outgoing iterator τότε γίνεται αναζήτηση στην κανονική κατεύθυνση των ακμών ελέγχοντας αν ο κόμβος αυτός έχει κατευθυνόμενο μονοπάτι τουλάχιστον προς έναν kw-κόμβο από κάθε σύνολο *Si* ώστε να αποτελέσει τη ρίζα μιας απάντησης.
- Κάθε κόμβο που βρίσκει να πληροί τις προϋποθέσεις ως ρίζα μιας απάντησης τον εισάγει στη λίστα απαντήσεων μαζί με το υπογράφημα που συμμετέχει αφού πρώτα ελέγξει αν έχει ξαναβρεθεί η ίδια απάντηση νωρίτερα.
- Όταν βρεθούν οι top-k απαντήσεις ταξινομούνται σύμφωνα με το βαθμό σχετικότητας που τους αποδίδεται από τη *συνάρτηση απόδοσης βαθμολογίας* την οποία βλέπουμε παρακάτω.
- Τέλος, αναζητούνται στη βάση δεδομένων με τη σειρά ταξινόμησης τα δένδρα των πλειάδων που σχετίζονται με τους kw-κόμβους που βρέθηκαν και τα αποτελέσματα επιστρέφονται στον χρήστη..

Τα κέρδη που πετυχαίνει το σύστημα αναζήτησης Bidirectional με τον μηχανισμό ταξινόμησης των κόμβων μέσω της απόδοσης προτεραιοτήτων είναι:

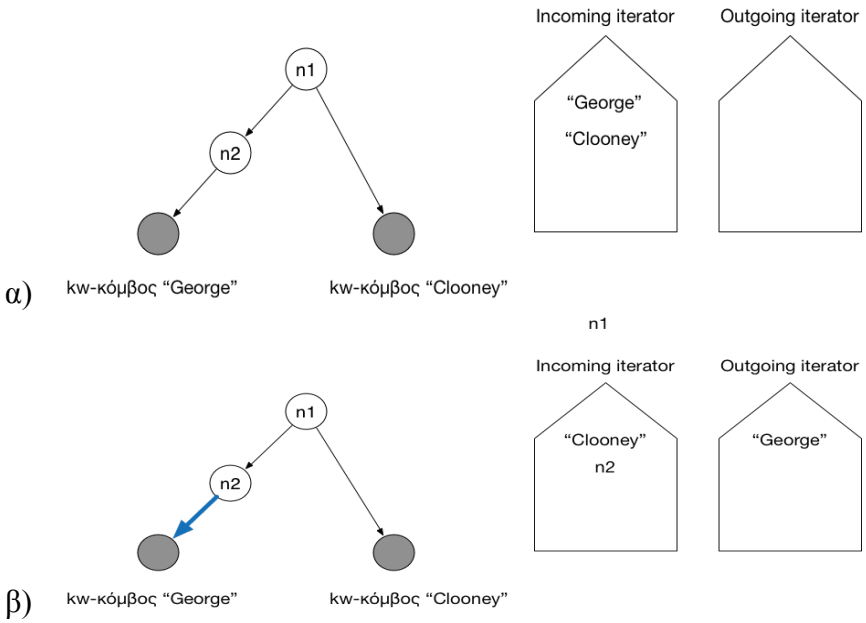
- Οι κόμβοι με μεγάλο πλήθος εισερχόμενων ακμών να έχουν και μεγαλύτερη προτεραιότητα για να συνεχιστεί η αναζήτηση από αυτούς διότι είναι αρκετά δημοφιλής.
- Η δημοτικότητα που μεταφέρεται στους απλούς κόμβους αντικατοπτρίζει το μήκος του μονοπατιού από έναν απλό κόμβο προς έναν kw-κόμβο.

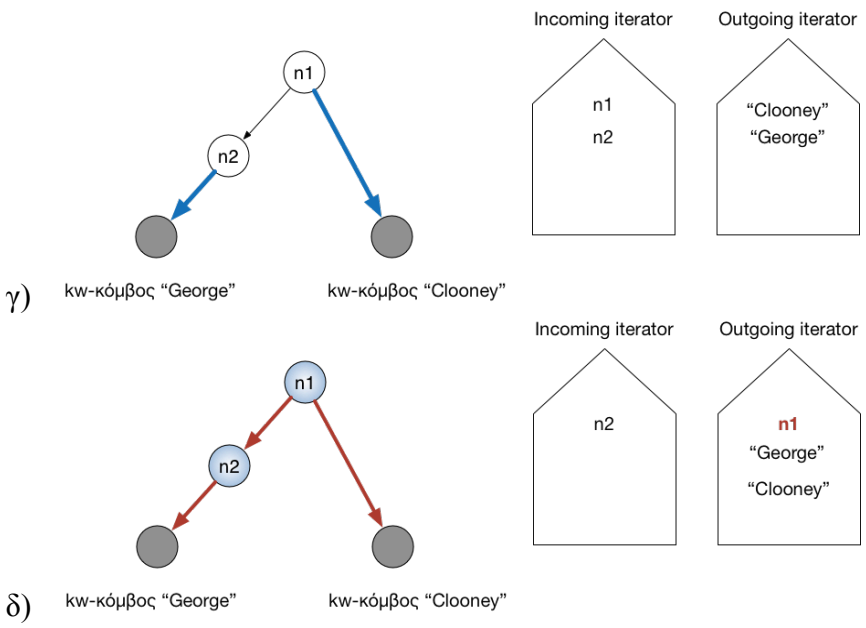
Ταξινόμηση των απαντήσεων:
Για την απόδοση βαθμολογίας σε κάθε απάντηση το σύστημα Bidirectional χρησιμοποιεί τα εξής στοιχεία:

- Το συνολικό βάρος των κόμβων της απάντησης *Nscore*, το συνολικό βάρος των ακμών της απάντησης *Escore* και ο παράγοντας βαρύτητας λ τα οποία προκύπτουν με την ίδια μέθοδο που είδαμε στο σύστημα Banks στο Κεφάλαιο 3.1.
- Η βαθμολογία κάθε απάντησης T προκύπτει από τη συνάρτηση απόδοσης βαθμολογίας $TotalScore(T) = Escore * Nscore^{\lambda}$.

Ως αποτέλεσμα της συνάρτησης απόδοσης βαθμολογίας $TotalScore(T)$ είναι οι απαντήσεις με μεγάλο μέγεθος να έχουν πιθανότατα χαμηλή βαθμολογία και οι απαντήσεις με μικρό μέγεθος να έχουν καλύτερη βαθμολογία καθώς θεωρούνται αντιπροσωπευτικότερες.

Στο Σχήμα 3.5 παίρνουμε ως παράδειγμα ένα τμήμα της βάσης δεδομένων κινηματογραφικών ταινιών IMDB για να δούμε τη διαδικασία εύρεσης μιας απάντησης σε ερώτημα με δυο λέξεις-κλειδιά Q=(“George”, “Clooney”).





Σχήμα 3.5 Bidirectional – Βήματα εύρεσης μιας απάντησης.

Τα βήματα που εκτελούνται στην αναζήτηση απαντήσεων στο Σχήμα 3.5 είναι τα εξής:

- Αρχικά αποδίδεται ο βαθμός δημοτικότητας στους kw-κόμβους που σχετίζονται με τη λέξη-κλειδί "George" και με τη λέξη-κλειδί "Clooney" και ταξινομούνται σύμφωνα με αυτόν το βαθμό στον incoming iterator (Σχήμα 3.5.α). Υποθέτουμε ότι οι κόμβοι ταξινομούνται με τη σειρά που βλέπουμε.
- Ο βαθμός δημοτικότητάς του kw-κόμβου "George" μεταδίδεται στον γειτονικό κόμβο *n2* σε ποσοστό *1-μ,* καθώς έχει τον καλύτερο βαθμό ανάμεσα στους δυο iterators. Ο κόμβος *n2* ταξινομείται στον incoming iterator και ο kw-κόμβος "George" μεταφέρεται στον outgoing iterator με μειωμένο τον αρχικό βαθμό δημοτικότητας σε ποσοστό *μ* για να ελεγχθεί εάν αποτελεί τη ρίζα μιας απάντησης (Σχήμα 3.5.β).
- Στο επόμενο βήμα ο κόμβος με τον καλύτερο βαθμό δημοτικότητας μεταξύ των δυο iterators είναι ο kw-κόμβος "Clooney" και μεταδίδει τον βαθμό δημοτικότητάς του στον γειτονικό του κόμβο *n1* σε ποσοστό *1-μ*. Στη συνέχεια ο kw-κόμβος " Clooney " μεταφέρεται στον outgoing iterator με μειωμένο τον αρχικό βαθμό δημοτικότητας σε ποσοστό *μ* (Σχήμα 3.5.γ).

Υποθέτουμε ότι ο βαθμός δημοτικότητας των κόμβων τους αποδίδει τη σειρά ταξινόμησης που βλέπουμε στους δυο *iterators*.

- Εφόσον ούτε ο kw-κόμβος " Clooney" στον outgoing iterator δεν αποτελεί τη ρίζα μιας απάντησης, επαναλαμβάνεται η διαδικασία μετάδοσης του βαθμού δημοτικότητας στους γειτονικούς κόμβους του κόμβου *n1* και τοποθετείται στον outgoing iterator. Με τον κόμβο *n1* να έχει πλέον τον μεγαλύτερο βαθμό δημοτικότητας μεταξύ των πρώτων κόμβων των δυο iterators, ελέγχεται για το αν αποτελεί τη ρίζα μιας απάντησης σύμφωνα με την κανονική κατεύθυνση των ακμών και βρίσκεται το κατευθυνόμενο μονοπάτι προς τους δυο kw-κόμβους (Σχήμα 3.5.δ).

Η απάντηση που βρίσκεται αποτελείται από τους κόμβους *"George"*_←_*n2*_←_*n1*_→*"Clooney"* με κόμβο-ρίζα τον κόμβο *n1*.

3.3. DISCOVER

Το σύστημα αναζήτησης Discover [20] παρουσιάστηκε από Έλληνες ερευνητές του Τμήματος Πληροφορικής του Πανεπιστημίου της California το έτος 2002. Για την αναπαράσταση του γράφου δεδομένων χρησιμοποιεί το *μοντέλο σχήματος* το οποίο είδαμε στο Κεφάλαιο 2 το οποίο χρησιμοποιεί τη μη κατευθυνόμενη έκδοσή του *σχήματος Gu*.

Για την αναπαράσταση των απαντήσεων δίνουμε τους παρακάτω ορισμούς:

Ορίζουμε ως ερώτημα το $Q = (k_1, \dots, k_m)$, βάση δεδομένων R με σχήμα G_U με n πίνακες $(R_1, \dots, R_n)$ και κάθε πίνακας R_i να έχει m πλειάδες ${t_1}^i, \dots, {t_m}^i$ και $t_i \bowtie t_j$ τη συνένωση μεταξύ των πλειάδων t_i και t_j.

Πίνακας πλειάδων λέξης-κλειδί TS (Tuple Set): Το σύνολο $R_i(k_j)$ με *j=1,…,m* είναι το γνήσιο υποσύνολο πλειάδων του πίνακα R_i οι οποίες περιέχουν την λέξη-κλειδί k_j

του ερωτήματος Q με $j=1,\ldots,m$. Στην υπόλοιπη εργασία θα αναφερόμαστε στον *πίνακα πλειάδων* $R_i(k_j)$ με τη συντομογραφία *kw-πίνακας*. Υπάρχει η περίπτωση το $R_i(k_j)$ να είναι κενό όταν ο πίνακας R_i δεν περιέχει σε καμία πλειάδα του τη λέξη-κλειδί k_j και καλείται *άδειος kw-πίνακας* [20, 38].

Συνδεδεμένο δίκτυο kw-πινάκων JNTS (*Joining Network of Tuple Sets*): Είναι ένα συνδεδεμένο δένδρο kw-πινάκων στο οποίο δυο kw-πίνακες $R_i(k_1)$ και $R_j(k_2)$ συνδέονται με ακμή όταν υπάρχει ακμή μεταξύ των πινάκων R_i και R_j στο σχήμα G_u.

Υποψήφια απάντηση (candidate network): Είναι ένα συνδεδεμένο δίκτυο kw-πινάκων JNTS το οποίο πρέπει να ικανοποιεί δυο συνθήκες:

- *Total:* Πρέπει κάθε λέξη-κλειδί του ερωτήματος Q να υπάρχει τουλάχιστον σε έναν kw-πίνακα (να μην υπάρχει κενός kw-πίνακας).
- *Minimal:* Θα πάψει να είναι total αν αφαιρεθεί οποιοσδήποτε kw-πίνακας.

Πρόκειται για ένα *συνδεδεμένο δίκτυο kw-πινάκων JNTS* το οποίο δεν περιέχει κάποιον κενό kw-πίνακα (χωρίς λέκεις-κλειδιά).

Συνδεδεμένο δίκτυο πλειάδων JNT (*Joining Network of Tuples*): Είναι ένα συνδεδεμένο δένδρο πλειάδων στο οποίο δυο πλειάδες $t_i \in R_i$ και $t_j \in R_j$ συνδέονται με ακμή όταν μπορούν να συνενωθούν σύμφωνα με τη σχέση πρωτεύον-ξένου κλειδιού των πινάκων R_i και R_j στο σχήμα G_u $\left((t_i \bowtie t_j) \in (R_i \bowtie R_j)\right)$. Θα πρέπει να είναι ταυτόχρονα:

- *Total*: Κάθε ζητούμενη λέξη-κλειδί του ερωτήματος Q πρέπει να περιέχεται τουλάχιστον σε μια από τις πλειάδες του *(Total JNT)*.
- *Minimal*: Δε μπορεί όταν αφαιρεθεί κάποια από τις πλειάδες του να συνεχίζει να είναι *Total (Minimal Total JNT)*.

Διαισθητικά μια *υποψήφια απάντηση* είναι ένα δίκτυο συνδεδεμένων πλειάδων από το οποίο ίσως να προκύψει μια ή και περισσότερες τελικές απαντήσεις στο

ερώτημα *Q* οι οποίες πληρούν αυτές τις προϋποθέσεις του *Minimal Total JNT*. Στην υπόλοιπη εργασία θα αναφερόμαστε σε αυτές με τη συντομογραφία *MTJNT*. Επειδή μια τελική απάντηση δε θεωρείται πολύ αντιπροσωπευτική όταν υπάρχει πολύ μεγάλη απόσταση μεταξύ δυο πλειάδων της (πολλές συνενώσεις), υπάρχει η παράμετρος *Tmax* η οποία καθορίζει το μέγιστο πλήθος πλειάδων που θα έχει η απάντηση με σκοπό να κρατηθεί ένα επίπεδο σχετικότητας στις απαντήσεις που θα ανακτηθούν.

Αναζήτηση των απαντήσεων

Για σχεσιακή βάση δεδομένων *R* και ερώτημα $Q = (k_1, \dots, k_m)$ τα βασικά στάδια της λειτουργίας του συστήματος Discover είναι τα εξής:

- Με είσοδο το ερώτημα *Q* παράγονται τα σύνολα των kw-πινάκων $R_i^{k_1}, \dots, R_i^{k_m}$ από κάθε πίνακα R_i της βάσης δεδομένων *R*, με κάθε kw-πίνακα $R_i^{k_j}$ να αφορά τη λέξη-κλειδί k_j του ερωτήματος (για *j=1,..,m*)
- Με τη λίστα kw-πινάκων *Qr* και το σχήμα G_u δημιουργείται ο γράφος των kw-πινάκων *Gts* (*Graph of tuple sets*) ως εξής:
  - Δημιουργείται ένας κόμβος R_i^k για κάθε ένα μη κενό kw-πίνακα R_i^k.
  - Δημιουργείται μια ακμή μεταξύ των kw-πινάκων R_i^k και R_j^k εάν υπάρχει η αντίστοιχη ακμή μεταξύ των πινάκων R_i και R_j στο G_u.
- Κάθε φορά επιλέγει *τυχαία* μια λέξη-κλειδί k_j και προσθέτει σε μια ουρά *Qr* τους kw-πίνακες που την περιέχουν.
- Στη συνέχεια επιλέγει το πρώτο *δένδρο kw-πλειάδων JNTS* της ουράς *Qr* και ελέγχει εάν πληροί τις προϋποθέσεις μιας *τελικής απάντησης MTJNT*.
  - Εάν δεν πληροί τις απαραίτητες προϋποθέσεις για την δημιουργία μιας απάντησης, απορρίπτεται από την ουρά.

- Εάν πληροί τις προϋποθέσεις και συμπεριλαμβάνει όλες τις λέξεις-κλειδιά του ερωτήματος, τότε επιλέγει τις πλειάδες του δένδρου που συνενώνονται βάσει ενός άπληστου αλγόριθμου τον οποίο βλέπουμε παρακάτω.
- Διαφορετικά ελέγχει τον γράφο των kw-πινάκων *Gts* για να ενώσει το *δένδρο kw-πλειάδων JNTS* με το γειτονικό του εφόσον ενώνονται στον γράφο των kw-πινάκων *Gts* και προσθέτει το νέο *δένδρο kw-πλειάδων JNTS* στην ουρά *Qr* για να ελεγχθεί ξανά.

- Για τη δημιουργία μιας απάντησης από ένα δένδρο kw-πινάκων επιλέγει το δένδρο πλειάδων βάσει ενός άπληστου αλγόριθμου τον οποίο βλέπουμε στην επομενη παράγραφο.
- Οι top-k τελικές απαντήσεις *MTJNT* εξάγονται από τις υποψήφιες απαντήσεις και ταξινομούνται σε αύξουσα σειρά σύμφωνα με το πλήθος των συνενώσεών τους.

Άπληστος αλγόριθμος επιλογής πλειάδων

Έχει αποδειχθεί στο [20] ότι το πρόβλημα της βέλτιστης δημιουργίας και αξιολόγησης τελικών απαντήσεων μέσα από ένα μεγάλο πλήθος υποψήφιων απαντήσεων είναι NP-complete πρόβλημα.

Για το λόγο αυτό οι τελικές απαντήσεις που επιλέγονται μέσα από ένα πλήθος υποψήφιων απαντήσεων βασίζονται σε έναν άπληστο αλγόριθμο ο οποίος λειτουργεί ως εξής:

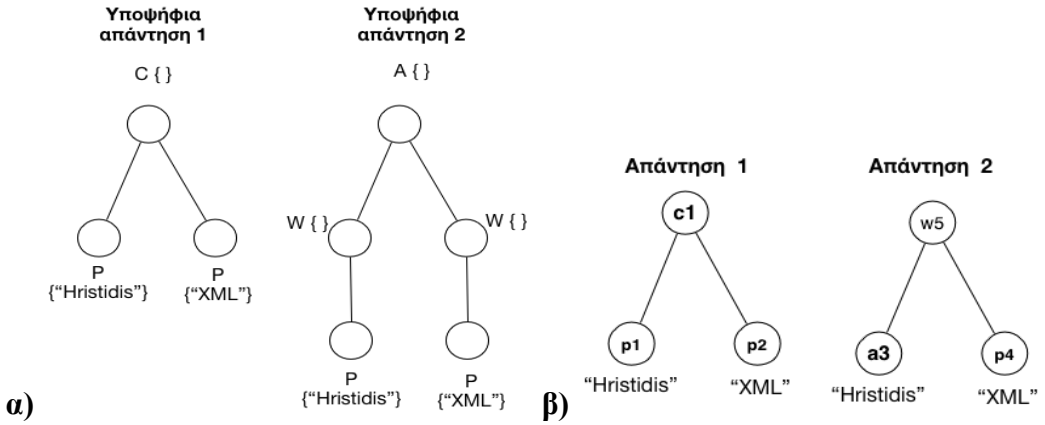
- Μέσα από ένα σύνολο kw-πινάκων επιλέγεται κάθε φορά η συνένωση *J* με τις περισσότερες εμφανίσεις μεταξύ δυο kw-πινάκων.
- Στη συνέχεια επιλέγονται οι τελικές απαντήσεις μέσα από το ελάχιστο πλήθος πλειάδων το οποίο συνενώνει η *J*.

Με *S* το πλήθος των υποψήφιων απαντήσεων μεγέθους $\leq T$, ο άπληστος αλγόριθμος έχει χρόνο χειρότερης εκτέλεσης $O((S\ T)^2)$, επειδή όλες οι υποψήφιες απαντήσεις θα ελεγχθούν το πολύ $S * T$ φορές και το πολύ άλλες τόσες φορές θα ενωθούν με άλλα δένδρα πλειάδων [20].

Ταξινόμηση των απαντήσεων

Για $Size(T)$ το πλήθος των συνενώσεών μιας απάντησης, η συνάρτηση απόδοσης βαθμολογίας στην απάντηση είναι η $Score(T) = \frac{1}{Size(T)}$ και αποδίδει καλύτερη βαθμολογία σε απαντήσεις με λιγότερες συνενώσεις επειδή θεωρούνται αντιπροσωπευτικότερες.

Σύμφωνα με τους παραπάνω ορισμούς βλέπουμε στο Σχήμα 3.6 ένα παράδειγμα αναπαράστασης υποψήφιων και τελικών απαντήσεων βασιζόμενοι στο Σχήμα 2.3 και 2.4 για ερώτημα *Q=("Hristidis", "XML")* και παράμετρο *Tmax=5*.



Σχήμα 3.6 Discover - Υποψήφιες και τελικές Απαντήσεις.

Στο Σχήμα 3.6.α βλέπουμε δυο διαφορετικές υποψήφιες απαντήσεις, με *C{}, A{}* στο επάνω μέρος τους να αφορούν τους πίνακες *Cites* και *Author* της βάσης δεδομένων DBLP. Το σύμβολο *P{"Hristidis"}* στο κάτω μέρος κάθε πίνακα σημαίνει την επιλογή όλων των πλειάδων του πίνακα *Paper* οι οποίες περιέχουν τη λέξη-κλειδί *"Hristidis"*.

Η τελική απάντηση 2 του Σχήματος 3.6.β είναι η *a3*⋈*w5*⋈*p4* και σημαίνει ότι ο συγγραφέας της πλειάδας *a3* που περιέχει τη λέξη-κλειδί "*Hristidis*" έγραψε ένα άρθρο το οποίο υπάρχει στην πλειάδα *p4* και περιέχει τη λέξη-κλειδί "*XML*".
Με τις εγγραφές που υπάρχουν στο στιγμιότυπο του Σχήματος 2.4, από την υποψήφια *απάντηση 1 = P{"Hristidis"}⋈C{}⋈P{"XML")* στο σχήμα Σχήμα 3.6.α προκύπτει η απάντηση 1 που βλέπουμε στο Σχήμα 3.6.β.

3.4. BLINKS

Το σύστημα αναζήτησης Blinks [19] παρουσιάστηκε το έτος 2007 από ερευνητές του Duke University και του IBM Thomas J. Watson Research Center. Χρησιμοποιεί το *μοντέλο γράφου* για την αναπαράσταση του γράφου δεδομένων το οποίο είδαμε στο Κεφάλαιο 2.1.

Κατά τη δημιουργία του γράφου δεδομένων εκτελεί τις εξής λειτουργίες:

- Σε κάθε ακμή *e* που εξέρχεται από έναν κόμβο *u* του γράφου δεδομένων αποδίδεται βάρος από τον τύπο $Escore(e) = log(1 + out\_degree(u))$ με $out\_degree(u)$ το πλήθος των εξερχόμενων ακμών του κόμβου *u*.
- Σε κάθε κόμβο *u* του γράφου δεδομένων αποδίδεται βάρος ίσο με το πλήθος των εισερχόμενων ακμών του (*indegree(u)*) λαμβάνοντας έτσι υπόψιν το πόσο δημοφιλής είναι μεταξύ των υπόλοιπων κόμβων.
- Στη συνέχεια χωρίζει τον γράφο δεδομένων σε τμήματα (*blocks*) και δημιουργεί ένα ευρετήριο δυο επιπέδων με το πρώτο επίπεδο να έχει πληροφορίες σχετικά με ποιο block ανήκει ο κάθε κόμβος και το δεύτερο επίπεδο να έχει λεπτομερής πληροφορίες για τα συντομότερα μονοπάτια μεταξύ των κόμβων που ανήκουν σε κάθε block.

Τμηματοποίηση του γράφου δεδομένων

Για την τμηματοποίηση του γράφου δεδομένων χρησιμοποιείται ο αλγόριθμος *Breath-First Search (BFS)* [49] ή ο αλγόριθμος *METIS* [24].

Breath-First Search τμηματοποίηση: Για τη δημιουργία ενός νέου block, ξεκινάει από εάν κόμβο που δεν έχει επισκεφτεί ξανά και εφαρμόζει διαπέραση BFS. Αν τελειώσει η διαπέραση χωρίς να έχει φτάσει στο επιθυμητό μέγεθος block (μικρό block), επιλέγεται άλλος κόμβος και επαναλαμβάνεται η διαδικασία.
METIS-Based Partitioning: Ο αλγόριθμος METIS [5] είναι ένας αλγόριθμος που δημιουργήθηκε για να τμηματοποιεί γράφους και στοχεύει στη μείωση του συνολικού πλήθους των συνοριακών κόμβων.

Κατά την τμηματοποίηση γίνεται η προσπάθεια τα blocks να έχουν σχεδόν ίδιο πλήθος κόμβων μεταξύ τους και η δημιουργία του κάθε ενός ξεκινάει από κάποιον κόμβο που δεν έχει ελεγχθεί ξανά. Αν τελειώσει η διαπέραση χωρίς να έχει φτάσει στο επιθυμητό μέγεθος block (μικρό block), επιλέγεται άλλος κόμβος και επαναλαμβάνεται η διαδικασία. Οι δυο αλγόριθμοι τμηματοποίησης δεν έχουν τις ίδιες επιδόσεις σε κάθε βάση δεδομένων και βλέπουμε τις διαφορές τους στις πειραματικές συγκρίσεις στο Κεφάλαιο 4.

Δημιουργία ευρετηρίου δυο επιπέδων

Ορίζουμε ως *portal-κόμβους* τους συνοριακούς κόμβους οι οποίοι συνδέονται ταυτόχρονα με δυο block. Όταν κάποιος κόμβος έχει ακμή η οποία εισέρχεται σε ένα block ονομάζεται *in-portal κόμβος* για αυτό το block και όταν έχει ακμή η οποία εξέρχεται από το block ονομάζεται *out-portal* κόμβος αυτού του block.

Το ευρετήριο δυο επιπέδων δημιουργείται με τα εξής βήματα:

- Αρχικά χωρίζεται ο γράφος δεδομένων σε blocks και καταγράφεται το block στο οποίο ανήκει ο κάθε κόμβος (1ο επίπεδο ευρετηρίου).
- Υπολογίζονται τα συντομότερα μονοπάτια μεταξύ των kw-κόμβων και των απλών κόμβων τα οποία δεν εξέρχονται από το block εκτελώντας τόσα αντίγραφα του αλγορίθμου *Dijsktra SSSP* όσοι είναι και οι κόμβοι του κάθε block. Στη συνέχεια υπολογίζονται τα συντομότερα μονοπάτια (με *Dijsktra SSSP*) από τους *out-portal* του κάθε block προς κάθε κόμβο εντός του block (2ο επίπεδο ευρετηρίου).

Πιο συγκεκριμένα υπολογίζονται τα εξής:

- Για την αντίστροφη αναζήτηση αποθηκεύονται σε μια λίστα τα σύνολα των κόμβων *u* που μπορούν να φτάσουν σε κάθε kw-κόμβο εντός του block ταξινομημένα σύμφωνα με τη συντομότερη απόσταση. Για κάθε out-portal κόμβο *p* ενός block αποθηκεύονται σε μια λίστα τα σύνολα των κόμβων εντός του block που μπορούν να φτάσουν στον *p,* ταξινομημένα σύμφωνα με τη συντομότερη απόσταση.
- Για την εμπρόσθια αναζήτηση αποθηκεύονται σε έναν πίνακα οι συντομότερες αποστάσεις από κάθε κόμβο *u* προς κάθε kw-κόμβο εντός του block. Για κάθε κόμβο *u* εντός του block αποθηκεύονται σε έναν πίνακα η συντομότερη απόσταση προς κάθε out-portal κόμβο.

Με τη βοήθεια του ευρετηρίου μπορεί να γίνεται αναζήτηση σύμφωνα με την αντίστροφη και την κανονική κατεύθυνση των ακμών, όταν υπάρχουν οι πληροφορίες για τα συντομότερα μονοπάτια μεταξύ των κόμβων με αποτέλεσμα να γίνονται γρηγορότερες αναζητήσεις. Όσο συντομότερα επισκεφτεί αντίστροφα η αναζήτηση κάποιον κόμβο, τόσο γρηγορότερα γίνεται γνωστό εάν βρέθηκε κάποια ρίζα απάντησης, ελέγχοντας την απόσταση που έχει προς τους άλλους kw-κόμβους. Για *N* το πλήθος των κόμβων του γράφου δεδομένων και k_b το πλήθος των λέξεων-κλειδιών εντός του block το μέγεθος το ευρετηρίου θα είναι στη χειρότερη περίπτωση $O(Nk_b)$. Στην πραγματικότητα όμως το μέγεθος είναι κατά πολύ μικρότερο καθώς δε συνδέεται άμεσα κάθε ένας κόμβος του γράφου δεδομένων και κάθε λέξη-κλειδί της βάσης δεδομένων.

Το σύστημα ευρετηρίου δυο επιπέδων πρέπει να αντιμετωπίσει το πρόβλημα του πότε ένα μονοπάτι μεταξύ δυο κόμβων είναι και το συντομότερο μέσα σε όλο τον γράφο δεδομένων εντός του block και δεν υπάρχει άλλο συντομότερο μονοπάτι που διέρχεται από άλλο block. Για τον έλεγχο αυτής της λειτουργίας εφαρμόζονται οι εξής τεχνικές:

- Στην αντίστροφη αναζήτηση θεωρείται ότι ο πρώτος kw-κόμβος της λέξης-κλειδί k_j που θα φτάσει στον out-portal κόμβο *u* θα έχει τη συντομότερη

απόσταση μεταξύ όλων των kw-κόμβων της λέξης-κλειδί k_j και οποιαδήποτε μετέπειτα πρόσβαση από kw-κόμβο της λέξης-κλειδί k_j θα παραλείπεται.

- Στην εμπρόσθια αναζήτηση ελέγχεται μέσω του αντίστοιχου πίνακα η συντομότερη απόσταση ενός κόμβου *u* που συμμετέχει στην απάντηση με τον κοντινότερο out-portal κόμβο. Εάν είναι μεγαλύτερη από την κοντινότερη απόσταση του κόμβου *u* με τον κάθε kw-κόμβο εντός του block, τότε θεωρείται ως συντομότερη απόσταση αυτή εντός του block.

Αναζήτηση απαντήσεων

Το σύστημα αναζήτησης Blinks βασίζει τη λειτουργία του στο ευρετήριο μέσω του οποίου ελέγχει κάθε φορά τις συντομότερες αποστάσεις μεταξύ των κόμβων και των block που ανήκουν. Για ερώτημα $Q = (k_1, \dots, k_m)$, ως απάντηση στο σύστημα Blinks ορίζεται κάθε σύνολο κόμβων του γράφου δεδομένων το οποίο πληροί τις προϋποθέσεις που τέθηκαν στο Κεφάλαιο 2.3 και τα βασικά στάδια αναζήτησης απαντήσεων είναι τα εξής:

- Αρχικά δημιουργείται ο γράφος δεδομένων, χωρίζεται σε τμήματα και δημιουργείται το ευρετήριο δυο επιπέδων.
- Για κάθε ζητούμενη λέξη-κλειδί k_i του ερωτήματος $Q = (k_1, \dots, k_m)$ δημιουργούνται τα σύνολα των kw-κόμβων S_i.
- Η αναζήτηση ξεκινά σύμφωνα με την αντίστροφη κατεύθυνση των ακμών από τους kw-κόμβους και κάθε φορά που εντοπίζεται κάποιος συνοριακός κόμβος *p,* συνεχίζεται σε αντίστροφη κατεύθυνση προς τα τμήματα που έχουν τον κόμβο *p* ως out-portal, επιλέγοντας ως επόμενο τμήμα αυτό με το μικρότερο πλήθος κόμβων που ελέγχθηκαν μέχρι εκείνη την στιγμή.
- Αφού επιλεγεί το κατάλληλο τμήμα στη συνέχεια επιλέγεται προς εξέταση ο κόμβος του τμήματος με συντομότερο μονοπάτι προς τον κόμβο *p*.
- Κάθε φορά που η αναζήτηση συναντά κάποιον κόμβο για πρώτη φορά, ελέγχεται στο ευρετήριο εάν υπάρχει μονοπάτι σύμφωνα με την κανονική

κατεύθυνση των ακμών προς κάθε ένα σύνολο kw-κόμβων S_i, ώστε να αποτελέσει τη ρίζα μιας απάντησης.

- Μέχρι να βρεθούν οι top-k απαντήσεις, κάθε απάντηση που βρίσκεται βαθμολογείται από τη *συνάρτηση απόδοσης βαθμολογίας* την οποία βλέπουμε στην επόμενη παράγραφο. Μεταξύ των απαντήσεων που έχουν κοινό κόμβο-ρίζα επιλέγεται αυτή με την καλύτερη βαθμολογία για να συμπεριληφθεί στις *top-k*.
- Όταν βρεθούν οι top-k απαντήσεις ταξινομούνται σύμφωνα με τη βαθμολογία τους, αναζητούνται στη βάση δεδομένων τα δένδρα των πλειάδων που σχετίζονται με τους kw-κόμβους που βρέθηκαν και τα αποτελέσματα επιστρέφονται στον χρήστη.

Ταξινόμηση απαντήσεων

Για ερώτημα $Q = (k_1, k_2, \dots, k_m)$ και απάντηση T, με r τον κόμβο-ρίζα και $n_1, n_2, \dots, n_m$ τους kw-κόμβους η συνάρτηση απόδοσης βαθμολογίας στις απαντήσεις λαμβάνει υπόψιν τα εξής στοιχεία:

- $\bar{S}_r(r)$: Το βάρος του κόμβου-ρίζα r το οποίο ισούται με το πλήθος των εισερχόμενων ακμών του (*indegree(r)*).
- $\bar{S}_n(n_i, k_i)$: Το μέσο όρο των βαρών των kw-κόμβων n_i που υπάρχουν στην απάντηση για κάθε λέξη-κλειδί k_i $\bar{S}_n(n_i, k_i) = \frac{\sum_1^z prestige(n_i)_z}{|n_i|}$, με z το πλήθος των kw-κόμβων n_i που υπάρχουν στην απάντηση για κάθε λέξη-κλειδί k_i.
- $\bar{S}_p(r, n_i)$: Το άθροισμα βαρών των ακμών της απάντησης με κάθε ακμή να μετριέται τόσες φορές όσα είναι και τα μονοπάτια που συμμετέχει.

Η συνάρτηση απόδοσης βαθμολογίας ορίζεται ως εξής:

$$S(T) = \bar{S}_r(r) + \sum_{i=1}^{m} \bar{S}_n(n_i, k_i) + \sum_{i=1}^{m} \bar{S}_p(r, n_i)$$

Σχήμα 3.7 Η συνάρτηση απόδοσης βαθμολογίας του Blinks.

Η συνάρτηση απόδοσης βαθμολογίας του συστήματος Blinks λαμβάνει υπόψιν τη δομή και το περιεχόμενό του γράφου δεδομένων καθώς τα $\bar{S}_r, \bar{S}_n$ και $\bar{S}_p$ αντιπροσωπεύουν μετρήσεις βασισμένες στο βάρος κόμβων και στη σχετικότητα των συνδέσεων μεταξύ τους.

3.5. DISCOVER με προτιμήσεις χρηστών

Δημιουργήσαμε μια παραλλαγή ενός συστήματος αναζήτησης, συνδυάζοντάς τον βασικό τρόπο λειτουργίας του συστήματος Discover [20] με τη μέθοδο ανάκτησης και ταξινόμησης απαντήσεων του συστήματος PERK [36]. Το σύστημα PERK παρουσιάστηκε από ερευνητές του Τμήματος Μηχανικών Η/Υ και Πληροφορικής του Πανεπιστημίου Ιωαννίνων τον Μάριο του 2010. Όπως και το σύστημα Discover χρησιμοποιεί το *μοντέλο σχήματος* για την αναπαράσταση του γράφου δεδομένων και η κύρια διαφορά μεταξύ των δυο συστημάτων είναι ότι στο σύστημα PERK η μέθοδος ταξινόμησης των απαντήσεων βασίζεται στις προσωπικές προτιμήσεις του κάθε χρήστη.

Οι προτιμήσεις αυτές δηλώνονται χρησιμοποιώντας συγκεκριμένες λέξεις-κλειδιά ως ανεξάρτητη είσοδος στο σύστημα από τις λέξεις-κλειδιά του ερωτήματος και αντιπροσωπεύουν τα προσωπικά ενδιαφέροντα του κάθε χρήστη, με σκοπό χρήστες με διαφορετικά ενδιαφέροντα να ανακτούν και διαφορετικές απαντήσεις. Επίσης υπάρχει η δυνατότητα να δηλωθεί κάποια σειρά προτεραιότητας μεταξύ των προτιμήσεων με σκοπό να ανακτώνται πρώτα οι απαντήσεις που περιέχουν στις πλειάδες τους τις λέξεις-κλειδιά μεγαλύτερης προτεραιότητας.

Ταξινόμηση προτιμήσεων

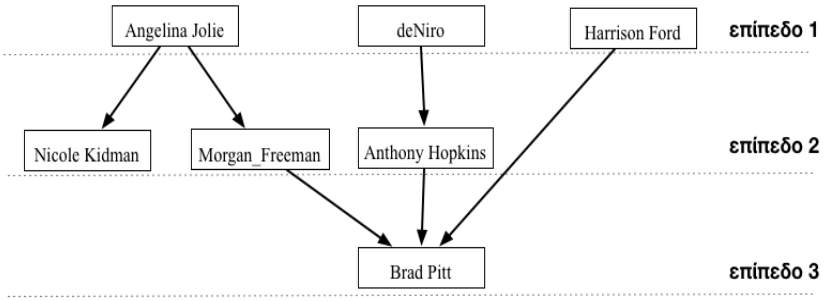
Για να γίνει η αναζήτηση των απαντήσεων αρχικά ταξινομούνται οι προτιμήσεις των χρηστών σε επίπεδα, σύμφωνα με τη σειρά προτεραιότητας που υπάρχει μεταξύ τους. Στη λίστα προτιμήσεων το σύμβολο «>» δηλώνει ότι η προτίμηση που βρίσκεται αριστερά του θα έχει προτεραιότητα έναντι της προτίμησης

που βρίσκεται δεξιά του. Η ταξινόμηση γίνεται μέσω της δημιουργίας ενός κατευθυνόμενου γράφου προτιμήσεων *Gp* ως εξής:

- Για την κάθε προτίμηση ενός χρήστη δημιουργείται αντίστοιχα στον γράφο προτιμήσεων *Gp* και ένας κόμβος με το περιεχόμενο της προτίμησης αυτής.
- Όταν μια προτίμηση έχει προτεραιότητα έναντι κάποιας άλλης, δημιουργείται μια κατευθυνόμενη ακμή μεταξύ των κόμβων τους στον γράφο προτιμήσεων με κατεύθυνση που ξεκινάει από τον κόμβο της ακμής που προηγείται.

Βασιζόμενοι στη βάση δεδομένων κινηματογραφικών ταινιών IMDB βλέπουμε στο Σχήμα 3.8 ένα παράδειγμα προτιμήσεων δηλωμένες με σειρά προτεραιότητας μεταξύ τους και τον αντίστοιχο γράφο προτιμήσεων *Gp* που δημιουργείται από αυτές.

Angelina Jolie > Nicole Kidman
Angelina Jolie > Morgan Freeman
Morgan Freeman > Brad Pitt
Anthony Hopkins > Brad Pitt
deNiro > Anthony Hopkins
Harrison Ford > Brad Pitt



Σχήμα 3.8 Οι προτιμήσεις και γράφος των προτιμήσεων Gp.

Οι προτιμήσεις του *επιπέδου 1* έχουν προτεραιότητα έναντι των προτιμήσεων του *επιπέδου 2* και *3* και οι προτιμήσεις του *επιπέδου 2* έχουν προτεραιότητα έναντι των προτιμήσεων του *επιπέδου 3*. Μέχρι να βρεθούν οι top-k απαντήσεις, πρώτες θα

επιλεγούν όσες απαντήσεις βρεθούν να περιέχουν στις πλειάδες τους εκτός από τις λέξεις-κλειδιά του ερωτήματος και κάποια από τις προτιμήσεις του επιπέδου 1, θα ακολουθήσουν οι απαντήσεις που θα βρεθούν να περιέχουν κάποια από τις προτιμήσεις του επιπέδου 2 και στη συνέχεια του επιπέδου 3. Το κέρδος με αυτή τη μέθοδο ταξινόμησης των απαντήσεων είναι ότι κατά πρώτον επιστρέφονται αποτελέσματα σχετικά με τις προτιμήσεις του χρήστη καλύπτοντας έτσι κατά έναν μεγάλο βαθμό τις απαιτήσεις του και κατά δεύτερον εξοικονομείται χρόνος αποφεύγοντας περιττές αναζητήσεις σε περιεχόμενο που δεν σχετίζεται καθόλου με κάποια από τις προτιμήσεις.

Βλέπουμε ένα παράδειγμα ταξινόμησης απαντήσεων βασισμένη στις προτιμήσεις χρήστη. Για ερώτημα $Q = (k_1)$ με μια λέξη-κλειδί και με $pref_i$ τις προτιμήσεις χρήστη (με *i= 1,…,4*) και σειρά προτεραιότητας:

$pref_1 > pref_2$

$pref_2 > pref_3$

$pref_4 > pref_2$

Για απαντήσεις στο ερώτημα Q την $T_1 = (k_1, pref_1, pref_3)$ και την $T_2 = (k_1, pref_2)$:

- Η απάντηση T_1 περιέχει την προτίμηση $pref_3$ η οποία είναι λιγότερο προτιμώμενη από την $pref_2$ που υπάρχει στην απάντηση T_2.
- Όμως η απάντηση T_1 θα ανακτηθεί πριν από την απάντηση T_2 επειδή περιέχει επιπλέον την προτίμηση $pref_1$ η οποία είναι περισσότερο προτιμώμενη από την προτίμηση $pref_2$.

Αναπαράσταση απαντήσεων

Για την αναπαράσταση των απαντήσεων χρησιμοποιεί τις ίδιες προδιαγραφές με του συστήματος Discover, σύμφωνα με το οποίο για να θεωρείται μια υποψήφια απάντηση ως τελική απάντηση θα πρέπει να πληροί τις προϋποθέσεις του Minimal Total Joining Network of Tuples (*MTJNT*).

Μοναδικότητα απαντήσεων

Μια σημαντική λειτουργία του συστήματος PERK είναι η ύπαρξη ποικιλίας των αποτελεσμάτων (diversity) αποφεύγοντας απαντήσεις με την ίδια πληροφορία καθώς η ανάκτηση απαντήσεων με διαφορετικό περιεχόμενο μεταξύ τους φαίνεται ότι αυξάνει την ικανοποιησιμότητα των χρηστών [36]. Πριν μια απάντηση συμπεριληφθεί στη λίστα τελικών απαντήσεων ελέγχεται για την ομοιομορφία της με όλες όσες έχουν ήδη βρεθεί και η μέτρησή της βασίζεται σε έναν Jaccard-based [47] ορισμό απόστασης ως εξής:

- Για δυο απαντήσεις τις *Ti* και *Tj* οι οποίες αποτελούνται από τα σύνολα εγγραφών *A* και *B* αντίστοιχα, η απόσταση μεταξύ των συνόλων εγγραφών τους υπολογίζεται από τον τύπο $d(T_i, T_j) = 1 - \left(\frac{|A \cap B|}{|A \cup B|}\right)$.

- Εάν η απόσταση μεταξύ δυο απαντήσεων είναι μηδενική τότε το περιεχόμενό τους θα είναι ίδιο και σε αυτή την περίπτωση η απάντηση που βρίσκεται δεύτερη θα απορρίπτεται.

Αναζήτηση απαντήσεων

Ο τρόπος αναζήτησης των απαντήσεων βασίζεται στον τρόπο λειτουργίας του συστήματος Discover που είδαμε στο Κεφάλαιο 3.3.

Οι λέξεις-κλειδιά με τις οποίες ξεκινάει η αναζήτηση δεν επιλέγονται τυχαία όπως στο σύστημα Discover αλλά σύμφωνα με τη σειρά προτεραιότητάς τους από τον γράφο προτιμήσεων *Gp*.

Η αναζήτηση των απαντήσεων γίνεται ως εξής:

- Με είσοδο το ερώτημα $Q = (\mathrm{k_1}, \dots, \mathrm{k_j})$ και σχήμα G_u παράγονται τα σύνολα των kw-πινάκων $\mathrm{R_i}^{\mathrm{k_1}}, \dots, \mathrm{R_i}^{\mathrm{k_m}}$ με την ίδια μέθοδο που είδαμε στο σύστημα Discover.
- Στη συνέχεια δημιουργείται ο γράφος προτιμήσεων *Gp* με τη μέθοδο που είδαμε στο Σχήμα 3.8.
- Μέχρι να βρεθούν οι top-k απαντήσεις επιλέγεται κάθε φορά η *προτίμηση λέξη-κλειδί* από τον κόμβο του γράφου προτιμήσεων *Gp* που δεν έχει

εισερχόμενη ακμή για να διατηρηθεί η σειρά προτεραιότητας των προτιμήσεων.

- Τα σύνολα των kw-πινάκων $R_i^{k_j}$ που περιέχουν την προτίμηση που επιλέγεται από τον γράφο προτιμήσεων προστίθενται στην ουρά υποψήφιων απαντήσεων *Qr*.
- Ο έλεγχος εύρεσης των τελικών απαντήσεων μέσα από τις υποψήφιες απαντήσεις γίνεται με την ίδια διαδικασία του συστήματος Discover.

Στην πειραματική σύγκριση των δυο συστημάτων βλέπουμε τις διαφορές τους στις επιδόσεις.

ΚΕΦΑΛΑΙΟ 4. ΠΕΙΡΑΜΑΤΙΚΗ ΣΥΓΚΡΙΣΗ

Σε αυτό το κεφάλαιο βλέπουμε τα αποτελέσματα των πειραματικών δοκιμών των συστημάτων αναζήτησης που εξετάσαμε τα οποία συγκρίνουμε σε διάφορους τομείς. Βλέπουμε τις πηγές ανάκτησης των ερωτημάτων και τους λόγους που επιλέξαμε τα συγκεκριμένα ερωτήματα για τις πειραματικές δοκιμές αλλά και κάποια στατιστικά στοιχεία σχετικά με το μέγεθος των βάσεων δεδομένων που χρησιμοποιήσαμε. Επίσης βλέπουμε κάποιους περιορισμούς των συστημάτων αναζήτησης που εξετάσαμε αλλά και τις αλλαγές που συμβαίνουν στην απόδοσή τους όταν αλλάζουμε κάποιες από τις παραμέτρους λειτουργίας τους. Τέλος, συγκρίνουμε τις επιδόσεις της παραλλαγής του συστήματος που δημιουργήσαμε σε σχέση με το αρχικό του σε διάφορους τομείς.

4.1. Τα δεδομένα

Οι σχεσιακές βάσεις δεδομένων που χρησιμοποιούμε στις πειραματικές συγκρίσεις της εργασίας είναι η DBLP [43] και η IMDB [44] και τα αντίγραφά τους

ανακτήθηκαν το 2015. Οι καταχωρήσεις της βάσης δεδομένων DBLP έχουν ξεκινήσει το έτος 1936 και περιλαμβάνουν στοιχεία σχετικά με δημοσιεύσεις επιστημονικών άρθρων συνεδρίων και περιοδικών. Η βάση δεδομένων IMDB περιλαμβάνει καταχωρήσεις σχετικά με κινηματογραφικές ταινίες, προσωπικό και στούντιο και οι εγγραφές της χρονολογούνται από το έτος 1874 [45]. Στον Πίνακα 1 βλέπουμε τα στοιχεία των ακμών και κόμβων των βάσεων δεδομένων και στο παράρτημα της εργασίας υπάρχουν αναλυτικά στατιστικά και ιστορικά στοιχεία για την κάθε μια.

Πίνακας 1 Πίνακας εγγραφών βάσεων δεδομένων.

Βάση δεδομένων	Σύνολο κόμβων *(V)*	Σύνολο ακμών *(E)*
DBLP	4.926.329	16.986.618
IMDB	6.637.432	23.802.122

4.2. Ρυθμίσεις πειραμάτων

Ο πηγαίος κώδικας των συστημάτων αναζήτησης που εξετάζουμε διατέθηκε από τους δημιουργούς τους, οι οποίοι τα υλοποίησαν σε γλώσσα προγραμματισμού Java και η εκτέλεσή τους έγινε στο περιβάλλον προγραμματισμού Eclipse. Για κάθε ερώτημα ανακτήθηκαν οι top-10 απαντήσεις καθώς θεωρείται από τους περισσότερους ερευνητές του χώρου ένα αντιπροσωπευτικό πλήθος απαντήσεων για την αξιολόγηση των συστημάτων αναζήτησης.

Ποιότητα και πηγές ανάκτησης των ερωτημάτων

Σε σχετικές έρευνες [8, 12] φάνηκε ότι καθοριστικό ρόλο στην αντικειμενική αξιολόγηση των συστημάτων αναζήτησης που βασίζονται σε λέξεις–κλειδιά αποτελεί η χρήση ερωτημάτων πραγματικού κόσμου αντί για τη δημιουργία ερωτημάτων επιλέγοντας τυχαία τις λέξεις-κλειδιά καθώς και η εκτέλεση ενός πλήθους τουλάχιστον 50 ερωτημάτων [39] ή και περισσότερων [37] για κάθε ένα σύστημα. Για τους λόγους αυτούς στις πειραματικές δοκιμές της εργασίας χρησιμοποιήθηκαν

ερωτήματα πραγματικού κόσμου και συγκεντρώθηκαν για κάθε μια από τις βάσεις δεδομένων τέσσερα διαφορετικά μεγέθη ερωτημάτων, από 2 μέχρι και 5 λέξεις-κλειδιά με 20 διαφορετικά ερωτήματα για κάθε μέγεθος. Δηλαδή χρησιμοποιούμε συνολικά 80 διαφορετικά ερωτήματα πραγματικού κόσμου σε κάθε βάση δεδομένων. Στο παράρτημα Α υπάρχουν τα ερωτήματα χωρισμένα ανά μέγεθος και για κάθε βάση δεδομένων, μαζί με το πλήθος των πλειάδων για την κάθε λέξη-κλειδί που περιέχουν.

Πηγές ανάκτησης των ερωτημάτων

Τα ερωτήματα πραγματικού κόσμου που χρησιμοποιήθηκαν επιλέχθηκαν από τις πηγές που έδωσαν οι ίδιοι οι συγγραφείς των συστημάτων αναζήτησης και από ερωτήματα αληθινών χρηστών τα οποία ανακτήθηκαν από τα αρχεία log δυο από των γνωστότερων διαδικτυακών μηχανών αναζήτησης (AOL και BING) [22, 40]. Τα ερωτήματα των διαδικτυακών μηχανών αναζήτησης συγκεντρώθηκαν από ερευνητές του αντίστοιχου ερευνητικού πεδίου και βρίσκονται διαθέσιμα στο διαδίκτυο για ερευνητικούς σκοπούς [22, 40]. Επίσης τα ερωτήματα που χρησιμοποιήθηκαν ελέγχθηκαν ώστε να είναι σχετικά με τις βάσεις δεδομένων που χρησιμοποιούμε.

4.3. Έλεγχος εσωτερικών μηχανισμών συστημάτων αναζήτησης

Εξετάζουμε τους εσωτερικούς μηχανισμούς αναζήτησης των συστημάτων που συγκρίνουμε ώστε να δούμε τις αλλαγές που θα συμβούν στις επιδόσεις των αναζητήσεων των ερωτημάτων.

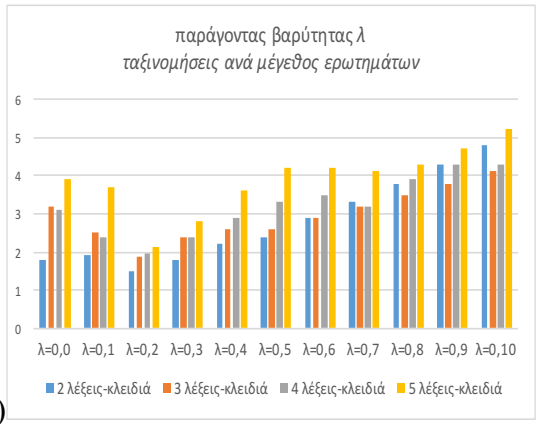
Σύστημα Banks και Bidirectional

Όπως είδαμε στα προηγούμενα κεφάλαια, πριν ένα σύστημα αναζήτησης επιστρέψει τις απαντήσεις που βρίσκει, τις ταξινομεί αποδίδοντάς τους κάποιο βαθμό σχετικότητας προς το ερώτημα με σκοπό οι απαντήσεις με μικρό μέγεθος να έχουν καλύτερη βαθμολογία. Όμως υπάρχουν περιπτώσεις στις οποίες σε κάποιες απαντήσεις με μεγάλο μέγεθος αποδίδεται καλύτερη βαθμολογία από κάποιες άλλες με μικρότερο μέγεθος και ονομάζουμε αυτή την περίπτωση «*λανθασμένη ταξινόμηση*» απαντήσεων. Ο παράγοντας βαρύτητας λ με τιμές στο πεδίο [0-1] καθορίζει το ποσοστό που θα λαμβάνει υπόψιν η συνάρτηση βαθμολόγησης των απαντήσεων την

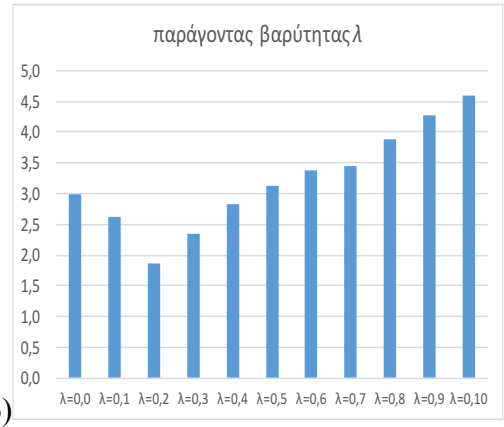
τελική βαθμολογία των κόμβων *(Nscore)* και την τελική βαθμολογία των ακμών *(Escore)* κάθε απάντησης.

Στο Σχήμα 4.1.α βλέπουμε για διαφορετικές τιμές του παράγοντα βαρύτητας λ τις περιπτώσεις στις οποίες αυξάνονται ή μειώνονται οι λανθασμένες ταξινομήσεις των απαντήσεων για κάθε μέγεθος ερωτημάτων και στο Σχήμα 4.1.β βλέπουμε το μέσο όρο των λάθος ταξινομήσεων απαντήσεων.

α)



β)



Σχήμα 4.1 Πλήθος λάθος ταξινομημένων απαντήσεων.

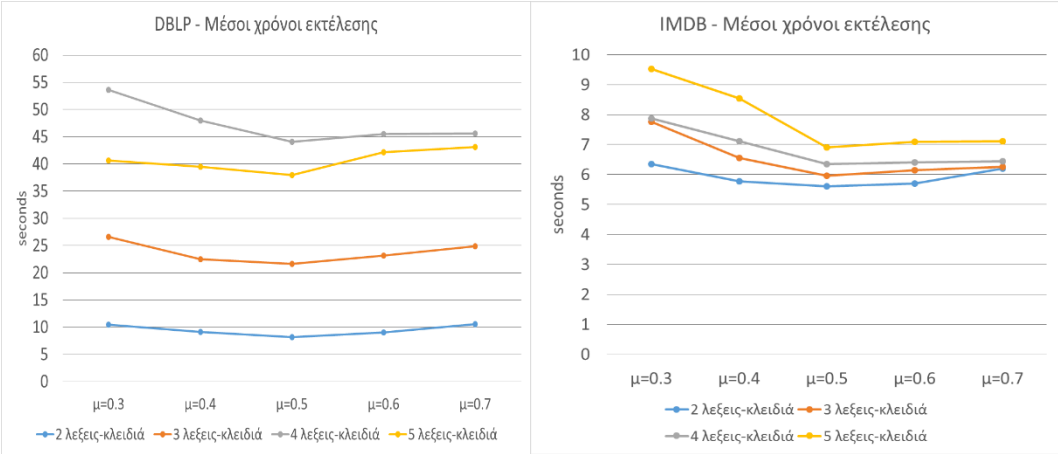
Βλέπουμε ότι ο παράγοντας βαρύτητας αποδίδει τα καλύτερα αποτελέσματα για τιμή $\lambda = 0.2$. Αυτό συμβαίνει επειδή υπάρχουν περισσότερες σωστά ταξινομημένες απαντήσεις (σύμφωνα με το μέγεθός τους) όταν στη συνάρτηση απόδοσης βαθμολογίας ληφθεί περισσότερο υπόψιν η σημαντικότητα των ακμών μεταξύ των κόμβων και λιγότερο υπόψιν το βάρος των κόμβων της απάντησης.

Σύστημα Bidirectional

Στο σύστημα αναζήτησης Bidirectional οι κόμβοι ταξινομούνται σύμφωνα με το σύστημα απόδοσης προτεραιοτήτων το οποίο είδαμε στο Κεφάλαιο 4.2 βάσει του οποίου αποδίδεται αρχικά κάποιος βαθμός δημοτικότητας στον κάθε kw-κόμβο και ο καθένας μεταδίδει το βαθμό αυτό στους γειτονικούς του κόμβους. Αν ο κόμβος από τον οποίο λάβουν τον βαθμό προτεραιότητας είναι αρκετά δημοφιλής τότε θα μεταδοθεί στους γειτονικούς του κόμβους ένα ποσοστό της δημοτικότητάς του οι οποίοι θα ταξινομηθούν αντίστοιχα και είναι πιθανό να οδηγήσουν στην ανάκτηση μιας αντιπροσωπευτικής απάντησης. Το ποσοστό δημοτικότητας που θα μεταδοθεί

καθορίζεται από τον παράγοντα μετάδοσης μ και αλλάζοντας τις τιμές του θα αλλάξει και η σειρά με την οποία ταξινομούνται οι επόμενοι κόμβοι αλλά και ο αρχικός βαθμός προτεραιότητας των kw-κόμβων. Επομένως θα αλλάξει η πορεία της αναζήτησης και κατά συνέπεια οι απαντήσεις οι οποίες θα ανακτηθούν.

Στο Σχήμα 4.2 βλέπουμε τις αλλαγές που συμβαίνουν στις επιδόσεις του συστήματος Bidirectional για διαφορετικές τιμές του παράγοντα μετάδοσης προτεραιότητας μ.



Σχήμα 4.2 Χρόνοι εκτέλεσης - Παράγοντας μετάδοσης προτεραιότητας μ.

Από τις μετρήσεις που έγιναν βλέπουμε ότι υπάρχει μια συγκεκριμένη τιμή για την οποία ο βαθμός μετάδοσης προτεραιοτήτων αποδίδει τον καλύτερο μέσο χρόνο εκτέλεσης και εξηγούμε τους λόγους που οδηγούν σε αυτές τις αλλαγές:

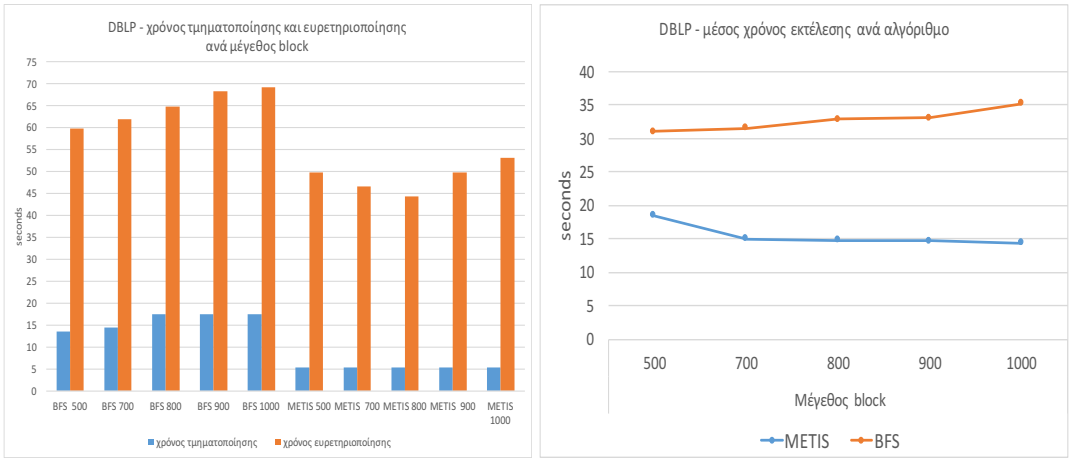
- Εάν μεταδοθεί ένα μικρό ποσοστό του βαθμού προτεραιότητας ενός κόμβου τότε οι γειτονικοί του κόμβοι θα ταξινομηθούν σε χαμηλές θέσεις έναντι των υπολοίπων κόμβων. Ως αποτέλεσμα αυτής της περίπτωσης είναι να αργήσει να βρεθεί μια απάντηση καθώς οι αρχικοί κόμβοι θα επανεξετάζονται συχνά για το αν αποτελούν τη ρίζα μιας απάντησης διότι θα έχουν κρατήσει ένα μεγάλο ποσοστό της δημοτικότητάς τους.
- Εάν μεταδοθεί μεγάλο ποσοστό του βαθμού προτεραιότητας ενός κόμβου τότε οι γειτονικοί του κόμβοι θα ταξινομηθούν σε υψηλές θέσεις έναντι των υπολοίπων αφήνοντας ένα μικρό μέρος του βαθμού προτεραιότητας στους αρχικούς κόμβους. Αποτέλεσμα θα είναι οι kw-κόμβοι να αργούν να

ελεγχθούν για το αν αποτελούν τη ρίζα μιας απάντησης και να αυξάνονται οι χρόνοι εκτέλεσης.

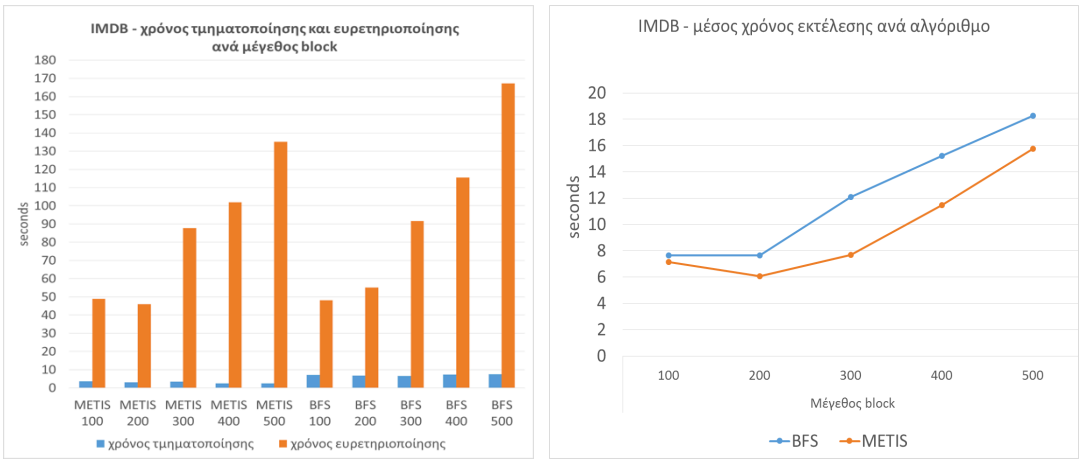
Υπήρξαν κάποια ερωτήματα για τα οποία με βαθμό $\mu=0.4$ ή $\mu=0.6$ οι αναζητήσεις έγιναν γρηγορότερα όμως η πλειοψηφία των αναζητήσεων έδειξε ότι γίνεται σε καλύτερο χρόνο για $\mu=0.5$.

Σύστημα Blinks

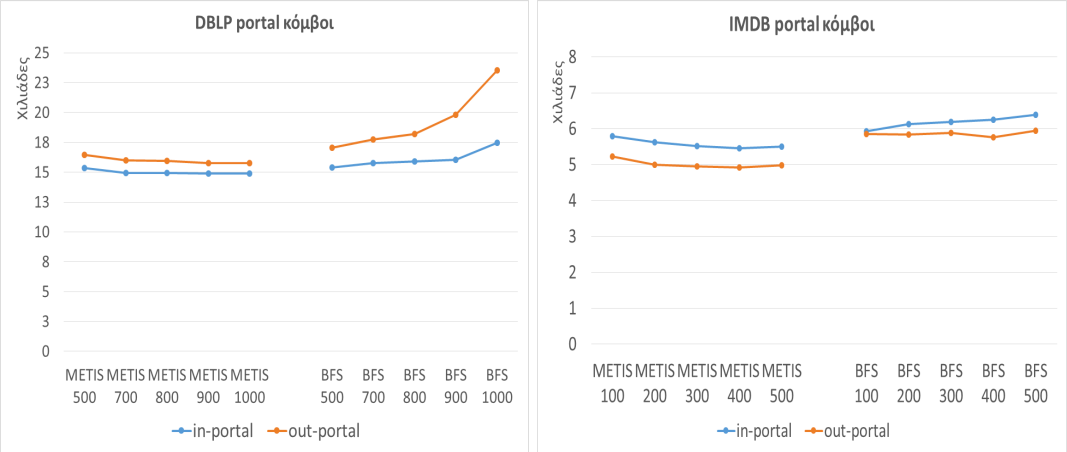
Για το σύστημα αναζήτησης Blinks βλέπουμε τους χρόνους που χρειάζεται για να γίνουν οι διαδικασίες της τμηματοποίησης και της ευρετηριοποίησης του γράφου δεδομένων με τους δυο διαφορετικούς αλγόριθμους τμηματοποίησης.



Σχήμα 4.3 DBLP – Τμηματοποίηση-ευρετηριοποίηση και χρόνοι εκτέλεσης.



Σχήμα 4.4 IMDB – Τμηματοποίηση-ευρετηριοποίηση και χρόνοι εκτέλεσης.

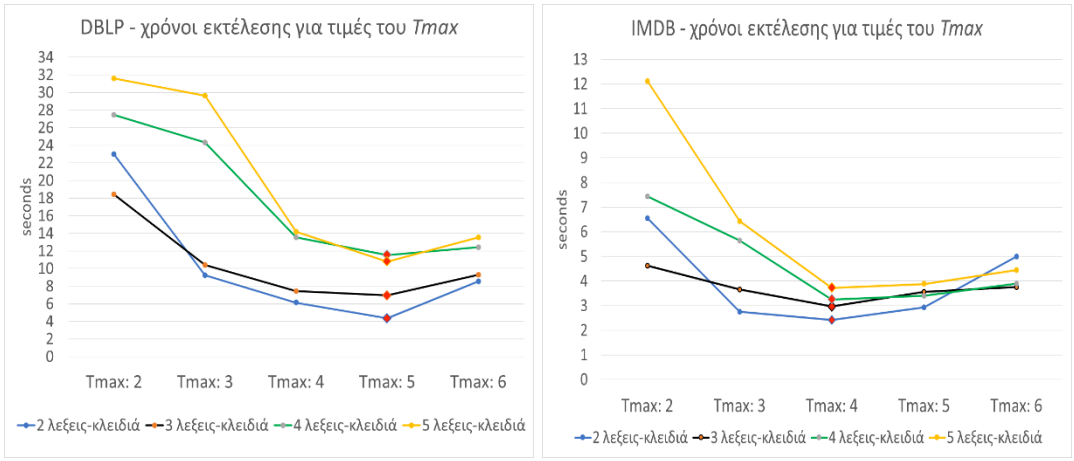


Σχήμα 4.5 Πλήθος portal κόμβων για κάθε μέγεθος block.

Παρατηρούμε στις περισσότερες περιπτώσεις ότι όσο αυξάνεται το μέγεθος των block τόσο μεγαλώνει ο χρόνος ευρετηριοποίησης και οι εγγραφές του ευρετηρίου για κάθε block. Στη βάση δεδομένων DBLP η προετοιμασία του γράφου δεδομένων ολοκληρώνονται γρηγορότερα για μέγεθος block 800 κόμβων και στη βάση δεδομένων IMDB ολοκληρώνονται γρηγορότερα για μέγεθος block 200. Υπεύθυνο για τις διαφορετικές ρυθμίσεις μέσω των οποίων πετυχαίνουμε τους καλύτερους χρόνους σε κάθε περίπτωση είναι η διαφορετική δομή της κάθε βάσης δεδομένων [19].

Σύστημα Discover

Στο σύστημα αναζήτησης Discover μπορεί να οριστεί το μέγιστο πλήθος πλειάδων που μπορεί να έχει κάθε απάντηση για να γίνεται αποδεκτή. Ο ορισμός του μεγέθους γίνεται μέσω της παραμέτρου T_{max} η οποία δέχεται για τιμές ακέραιους αριθμούς. Εξετάζουμε τη συμπεριφορά του συστήματος Discover για διαφορετικές τιμές της παραμέτρου T_{max} και στο Σχήμα 4.6 βλέπουμε τις αλλαγές που επιφέρει στις επιδόσεις του για τις διαφορετικές τιμές του.



Σχήμα 4.6 Discover – Χρόνοι εκτέλεσης για διαφορετικές τιμές του T_{max}.

Βλέπουμε ότι υπάρχει μια συγκεκριμένη τιμή για την παράμετρο T_{max} στην οποία οι αναζητήσεις των περισσότερων ερωτημάτων κάθε μεγέθους γίνονται στον καλύτερο δυνατό χρόνο. Παρατηρούμε ότι μεταξύ των δυο βάσεων δεδομένων, στη DBLP χρειάζεται μεγαλύτερο μέγεθος απαντήσεων για να υπάρξουν οι καλύτεροι χρόνοι εκτέλεσης των αναζητήσεων. Όπως βλέπουμε στις υπόλοιπες μετρήσεις των πειραματικών συγκρίσεων, αυτή η διαφορά οφείλεται στο είδος των εγγραφών που υπάρχει στη κάθε βάση δεδομένων.

Αποτελέσματα συγκρίσεων των παραμέτρων των συστημάτων αναζήτησης

Μετά από τους ελέγχους των εσωτερικών μηχανισμών των συστημάτων που εξετάσαμε, βλέπουμε στον Πίνακα 2 τις τιμές των παραμέτρων που χρησιμοποιούμε, με τις οποίες το κάθε σύστημα αναζήτησης έχει την καλύτερη απόδοση.

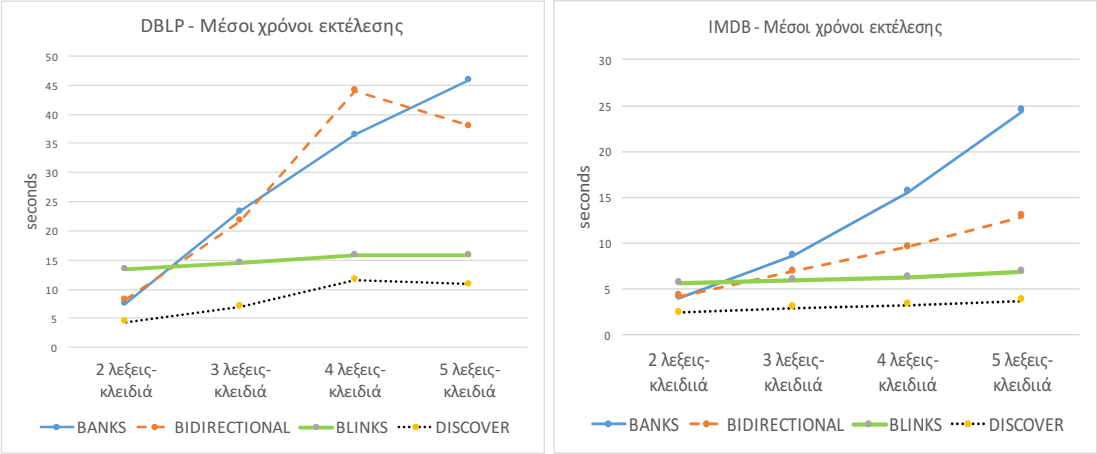
Πίνακας 2 Πίνακας παραμέτρων και τιμών των συστημάτων αναζήτησης.

Σύστημα αναζήτησης	Παράμετρος	Τιμή	Περιγραφή

Banks και Bidirectional	λ	0.2	Ορίζει το ποσοστό στο οποίο θα ληφθεί υπόψιν το βάρος των κόμβων (Nscore) και το βάρος ακμών (Escore) στην τελική βαθμολογία κάθε απάντησης. Στην τιμή 0.2 πετυχαίνει την καλύτερη απόδοση ταξινόμησης των απαντήσεων ανάλογα με το μέγεθός τους.
Bidirectional	μ	0.5	Ορίζει το ποσοστό μετάδοσης του βαθμού προτεραιότητας που διαδίδεται από έναν κόμβο προς τους γειτονικούς του κόμβους. Δέχεται τιμές στο εύρος [0-1], μεταδίδεται το ποσοστό *1-μ* και το υπόλοιπο της αφαίρεσης (*μ*) αποδίδεται στον αρχικό κόμβο.
Blinks	DBLP Μέγεθος block: 800 Τμηματοποίηση: METIS		Το μέγεθος block ορίζει το ιδανικό πλήθος κόμβων του κάθε block σύμφωνα με το οποίο επιτυγχάνονται οι καλύτεροι χρόνοι για την προετοιμασία του γράφου δεδομένων σε συνδυασμό με τον αλγόριθμο τμηματοποίησης.
	IMDB Μέγεθος block: 200 Τμηματοποίηση: METIS		
Discover	T_{max}	DBLP: 5	Ορίζει το μέγιστο επιτρεπόμενο πλήθος ενώσεων που μπορούν να έχουν οι τελικές απαντήσεις που θα ανακτηθούν.
		IMDB: 4	

4.4. Χρόνοι εκτέλεσης

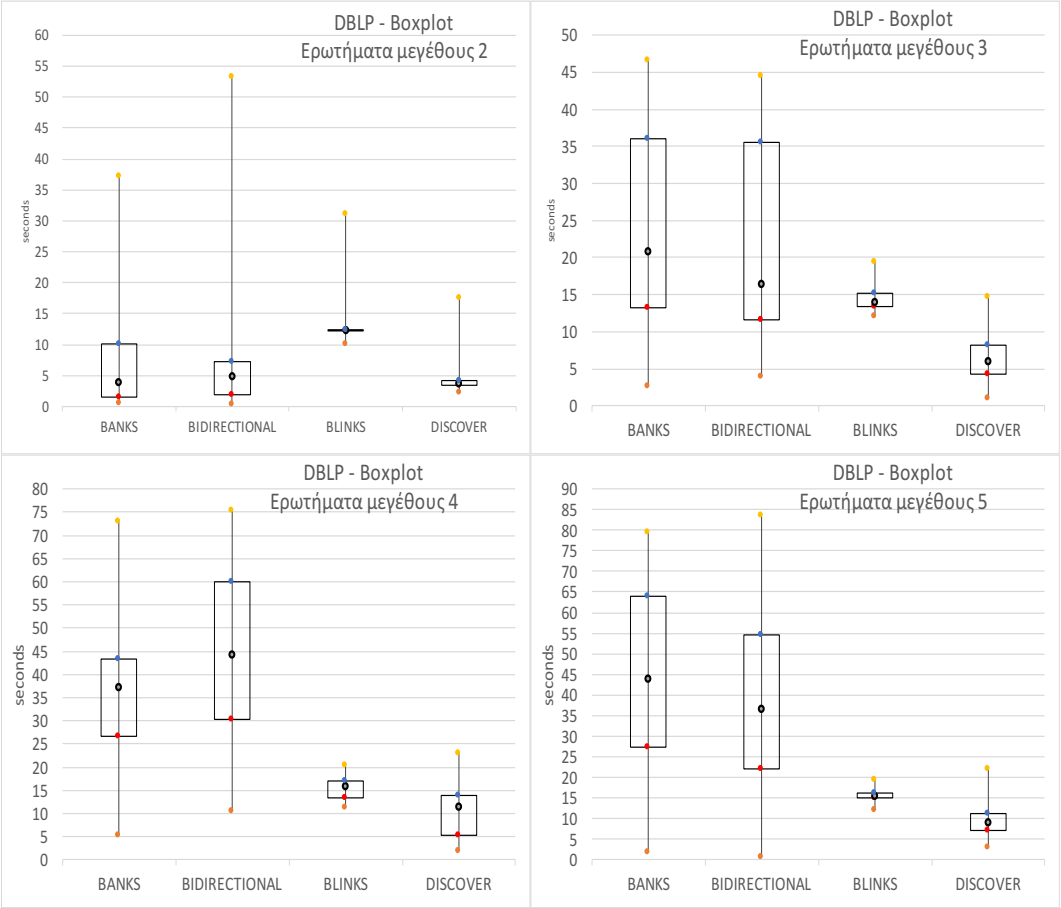
Σημαντική παράμετρο για την αξιολόγηση ενός συστήματος αναζήτησης αποτελεί ο χρόνος εκτέλεσης των αναζητήσεων των ερωτημάτων. Στο Σχήμα 4.7 βλέπουμε το μέσο χρόνο εκτέλεσης των αναζητήσεων των συστημάτων που εξετάσαμε για κάθε μήκος ερωτημάτων στις δυο βάσεις δεδομένων.



Σχήμα 4.7 DBLP - Μέσοι χρόνοι εκτέλεσης.

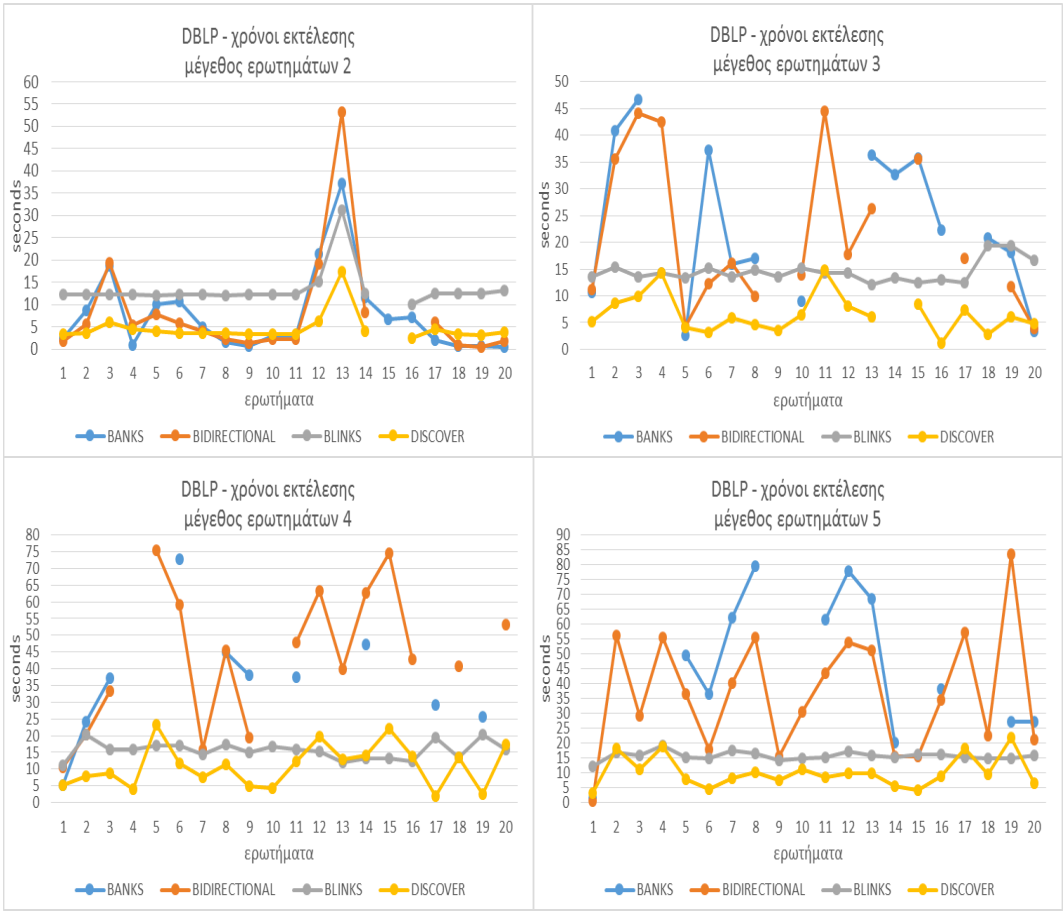
Τα δυο πρώτα συστήματα Banks και Bidirectional έχουν αυξημένους χρόνους εκτέλεσης συγκριτικά με τα συστήματα Blinks και Discover και αυτό οφείλεται στη διαφορετική μέθοδο αναζήτησης των απαντήσεων που χρησιμοποιούν. Το σύστημα Bidirectional στις περισσότερες περιπτώσεις εκτελεί τα ερωτήματα γρηγορότερα από το σύστημα Banks και θεωρείται αναμενόμενο επειδή κάποιες φορές ανακαλύπτει νωρίτερα απαντήσεις με πολλούς ενδιάμεσους και kw-κόμβους.

Στο Σχήμα 4.8 βλέπουμε τον χρόνο εκτέλεσης των συστημάτων αναζήτησης σε μορφή boxplot ξεχωριστά για κάθε μέγεθος ερωτημάτων για τη βάση δεδομένων DBLP.



Σχήμα 4.8 DBLP - χρόνοι εκτέλεσης σε μορφή boxplot.

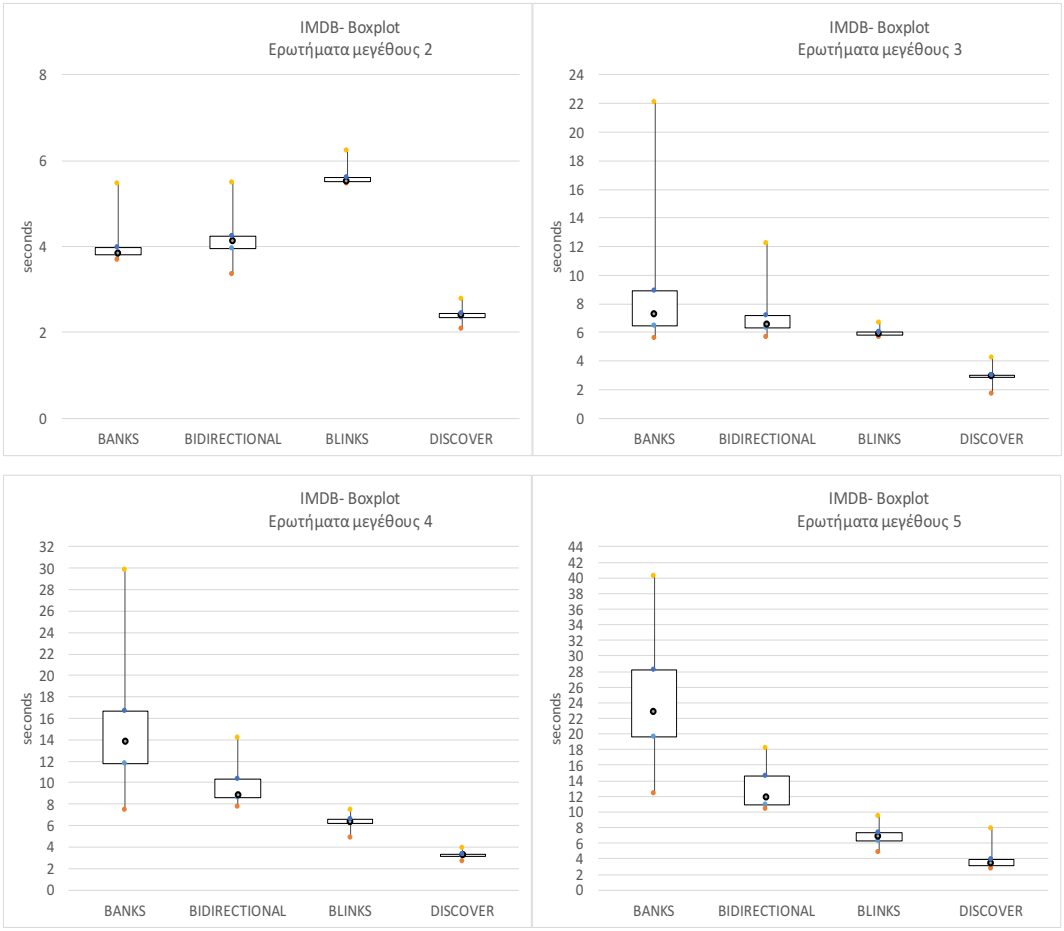
Στο Σχήμα 4.9 βλέπουμε αναλυτικά τους χρόνους εκτέλεσης για κάθε ερώτημα από κάθε σύστημα αναζήτησης στη βάση δεδομένων DBLP. Στα κενά σημεία είναι οι αδυναμίες επιστροφής απαντήσεων.



Σχήμα 4.9 DBLP – Αναλυτικοί χρόνοι εκτέλεσης για κάθε σύστημα αναζήτησης.

Βλέπουμε ότι ο μειωμένος όγκος του γράφου δεδομένων που χρησιμοποιεί το σύστημα Discover παίζει καθοριστικό ρόλο στον χρόνο εύρεσης των απαντήσεων καθώς έχει τους καλύτερους χρόνους εκτέλεσης στις περισσότερες περιπτώσεις. Αυτό γίνεται αντιληπτό και από το πλήθος των κόμβων που εξετάζονται σε κάθε ερώτημα (το αναλύουμε περισσότερο στο Κεφάλαιο 4.8) καθώς στις περισσότερες περιπτώσεις σε κάθε ερώτημα αναλογούν από μερικές δεκάδες έως και κάποιες χιλιάδες kw-κόμβοι για όλες τις λέξεις-κλειδιά που περιέχει. Το μέγεθος αυτό είναι σημαντικά μικρότερο από το να πρέπει αυτό το πλήθος κόμβων να εξεταστεί μέσα σε έναν γράφο δεδομένων που περιέχει εκατομμύρια κόμβους και ενώσεις.

Στο Σχήμα 4.10 βλέπουμε τον χρόνο εκτέλεσης των συστημάτων αναζήτησης σε boxplot για κάθε μήκος ερωτημάτων για τη βάση δεδομένων IMDB.



Σχήμα 4.10 IMDB - χρόνοι εκτέλεσης σε μορφή boxplot.

Στο Σχήμα 4.11 βλέπουμε αναλυτικά τους χρόνους εκτέλεσης για κάθε ερώτημα από κάθε σύστημα αναζήτησης στη βάση δεδομένων IMDB. Στα κενά σημεία είναι οι αδυναμίες επιστροφής απαντήσεων.

Σχήμα 4.11 IMDB – Αναλυτικοί χρόνοι εκτέλεσης για κάθε σύστημα αναζήτησης.

Συγκρίνοντας τα πειραματικά αποτελέσματα των συστημάτων αναζήτησης που εξετάσαμε με τα πειραματικά αποτελέσματα αντίστοιχων ερευνών [8] βλέπουμε ότι στις περισσότερες περιπτώσεις υπάρχει συμφωνία στις επιδόσεις τους ως προς τους χρόνους εκτέλεσης αλλά και ως προς τη συμπεριφορά των συστημάτων αναζήτησης και για τις δυο βάσεις δεδομένων. Επίσης βλέπουμε ότι οι μέσοι χρόνοι εκτέλεσης αυξάνονται σταδιακά όσο μεγαλώνει το μέγεθος των ερωτημάτων. Το σύστημα αναζήτησης Banks έχει τους μεγαλύτερους χρόνους εκτέλεσης με το σύστημα αναζήτησης Bidirectional να ακολουθεί και το σύστημα Discover εξακολουθεί να έχει τους καλύτερους χρόνους εκτέλεσης από τα υπόλοιπα.

Όπως παρατηρούμε μεταξύ των δυο βάσεων δεδομένων στην IMDB οι απαντήσεις εξάγονται σε καλύτερους χρόνους και αυτό είναι μια ένδειξη για το ρόλο που παίζουν στην αναζήτηση πληροφοριών το είδος των εγγραφών μιας βάσης

δεδομένων. Για παράδειγμα η βάση δεδομένων IMDB φιλοξενεί στοιχεία σχετικά με κινηματογραφικές ταινίες οι περισσότερες από τις οποίες διαφημίζονται με αποτέλεσμα οι χρήστες να γνωρίζουν ήδη κάποιο τμήμα της απάντησης που αναζητούν. Επομένως είναι πιθανότερο να συντάξουν ερωτήματα με λέξεις-κλειδιά από τα οποία θα ανακτηθούν απαντήσεις από πλειάδες που έχουν μικρές αποστάσεις μεταξύ τους που συνεπάγεται σε μικρό μέγεθος και κατά συνέπεια βρίσκονται σε συντομότερο χρονικό διάστημα. Σημαντική παρατήρηση αποτελεί επίσης ότι και για τις δυο βάσεις δεδομένων, στα συστήματα Banks και Bidirectional αυξάνονται σταδιακά οι χρόνοι των αναζητήσεων όσο αυξάνεται το μέγεθος των ερωτημάτων ενώ τα συστήματα Banks και Discover αποδίδουν εξίσου καλά αυξάνοντας ελάχιστα τους χρόνους εκτέλεσης των αναζητήσεων.

4.5. Αποτυχίες αναζήτησης

Κατά την εκτέλεση των αναζητήσεων κάποιες φορές τα συστήματα αναζήτησης αδυνατούν να επιστρέψουν απαντήσεις. Υπεύθυνα για την αποτυχία αυτή είναι η αδυναμία σχηματισμού απαντήσεων ανάλογα με τη μέθοδο αναζήτησης του κάθε συστήματος αναζήτησης. Στον Πίνακα 3 και Πίνακα 4 βλέπουμε αναλυτικά τα ποσοστά αποτυχίας απάντησης κάθε συστήματος για όλα τα μεγέθη ερωτημάτων.

Πίνακας 3 DBLP – Ποσοστά αποτυχιών αναζήτησης.

DBLP -Συνολικές αποτυχίες				
Μέγεθος ερωτημάτων	Banks	Bidirectional	Blinks	Discover
2 λέξεις-κλειδιά	0	2	1	1
3 λέξεις-κλειδιά	5	4	0	1
4 λέξεις-κλειδιά	10	4	0	1
5 λέξεις-κλειδιά	8	0	0	0
Συνολικές αποτυχίες	23	10	1	3
Ποσοστό απαντήσεων	71,25%	87,50%	98,75%	96,25%

Πίνακας 4 IMDB - Ποσοστά αποτυχιών αναζήτησης.

IMDB -Συνολικές αποτυχίες				
Μέγεθος ερωτημάτων	Banks	Bidirectional	Blinks	Discover
2 λέξεις-κλειδιά	0	0	0	0
3 λέξεις-κλειδιά	0	0	0	0
4 λέξεις-κλειδιά	0	0	0	0
5 λέξεις-κλειδιά	1	0	0	0
Συνολικές αποτυχίες	1	0	0	0
Ποσοστό απαντήσεων	98,75%	100,00%	100,00%	100,00%

Βλέπουμε ότι παρά το γεγονός ότι χρησιμοποιούμε ερωτήματα πραγματικού κόσμου, υπάρχουν αποτυχίες εύρεσης απαντήσεων από τα συστήματα αναζήτησης με τα συστήματα Banks και Bidirectional να έχουν τις περισσότερες. Παρατηρούμε ότι οι περισσότερες αποτυχίες απάντησης συμβαίνουν στη βάση δεδομένων DBLP, στην οποία συμβαίνει οι μέσοι χρόνοι εκτέλεσης να είναι αρκετά μεγαλύτεροι από τη βάση δεδομένων IMDB. Μεταξύ των συστημάτων αναζήτησης στη βάση δεδομένων DBLP, το σύστημα Banks, ακολουθεί το σύστημα Bidirectional και τέλος τα συστήματα Blinks και Discover με ελάχιστα ποσοστά αποτυχιών.

4.6. Τομή απαντήσεων

Στον Πίνακα 5 και Πίνακα 6 βλέπουμε το μέσο όρο της τομής των απαντήσεων συγκρίνοντας ανά ζεύγη τα συστήματα αναζήτησης. Στο παράρτημα της εργασίας υπάρχει η αναλυτική τομή των απαντήσεων για κάθε ερώτημα.

Πίνακας 5 Τομή απαντήσεων DBLP.

	Banks	Bidirectional	Blinks	Discover

Banks		29%	15%	12%
Bidirectional	29%		14%	11%
Blinks	15%	14%		15%
Discover	12%	11%	15%	

Πίνακας 6 Τομή απαντήσεων IMDB.

	Banks	Bidirectional	Blinks	Discover
Banks		28%	14%	13%
Bidirectional	28%		16%	15%
Blinks	14%	16%		18%
Discover	13%	15%	18%	

Παρατηρούμε ότι τα συστήματα μεταξύ τους επιστρέφουν απαντήσεις με χαμηλό ποσοστό όμοιων αποτελεσμάτων και αυτό οφείλεται στον διαφορετικό τρόπο με τον οποίο ανακαλύπτονται οι kw-κόμβοι και οι κόμβοι-ρίζα των απαντήσεων από το κάθε ένα. Το μεγαλύτερο ποσοστό τομής απαντήσεων υπάρχει μεταξύ των συστημάτων Banks και Bidirectional καθώς βασίζονται κατά ένα μέρος στην ίδια μέθοδο αναζήτησης.

4.7. Ποιότητα απαντήσεων

Στα προηγούμενα κεφάλαια είδαμε ότι το μέγεθος κάθε απάντησης δείχνει το πόσο στενά συνδεδεμένοι είναι μεταξύ τους οι kw-κόμβοι της και το πόσο αντιπροσωπευτική είναι η κάθε μια προς το ερώτημα του χρήστη. Για επιστροφή των 10 καλύτερων απαντήσεων, η απάντηση που θεωρείται η καλύτερη ταξινομείται στην $1^{η}$ θέση κατάταξης και η χειρότερη στη $10^{η}$ θέση. Προτείνουμε μια νεα μέθοδο μέτρησης της ποιότητας των απαντήσεων που επιστρέφουν τα συστήματα

αναζήτησης για κάθε ένα ερώτημα, μετρώντας το μέγεθος της κάθε απάντησης σε συνδυασμό με τη θέση κατάταξής της ως εξής:

Για ερώτημα Q και απάντηση *T* βλέπουμε τη συνάρτηση μέτρησης ποιότητας των απαντήσεων που επιστρέφονται στο ερώτημα *Q* στο Σχήμα 4.12 και είναι η εξής:

$$quality(Q) = \sum_1^k \frac{1}{SeqNum\,(T_k) * AnswerSize\,(T_k)}$$

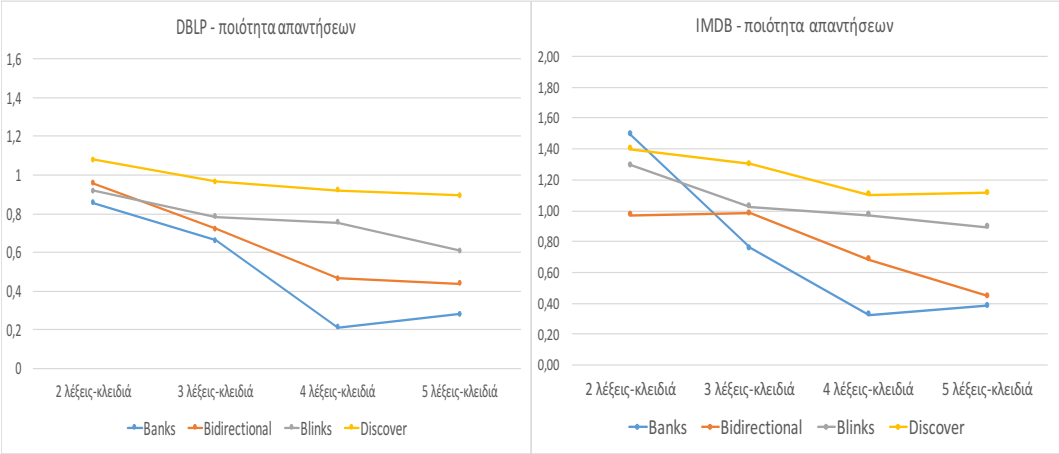
Σχήμα 4.12 Συνάρτηση μέτρησης ποιότητας των απαντήσεων κάθε ερωτήματος.

Στην συνάρτηση λαμβάνουμε υπόψιν τη θέση κατάταξης της κάθε απάντησης και το μέγεθος της και τα πεδία που χρησιμοποιούμε είναι τα εξής:

- Για k από την 1η μέχρι και την 10η σειρά κατάταξης κάθε απάντησης.
- $SeqNum(T_k)$ η σειρά κατάταξης της κάθε απάντησης *T*.
- $AnswerSize(T)$ το μέγεθος (πλήθος κόμβων) της απάντησης *T*.

Με τη συνάρτηση μέτρησης ποιότητας *quality(Q)* συνδυάζει το μέγεθος κάθε απάντησης που επιστρέφεται για ένα ερώτημα σε σχέση με τη σειρά κατάταξής της και επομένως όσο καλύτερη σειρά κατάταξης έχει κάποια απάντηση τόσο περισσότερο λαμβάνεται υπόψιν το μέγεθός της στη βαθμολογία της ποιότητας κάθε ερωτήματος.

Στο Σχήμα 4.13 βλέπουμε το μέσο όρο των της ποιότητας των ερωτημάτων για όλα τα μεγέθη ερωτημάτων στις δυο βάσεις δεδομένων.



Σχήμα 4.13 Μέσος όρος ποιότητας απαντήσεων.

Βλέπουμε ότι τα χαμηλότερα επίπεδα σημειώθηκαν στα συστήματα Banks και Bidirectional, στα οποία οι απαντήσεις που επιστράφηκαν είχαν πολύ μεγαλύτερο μέγεθος σε σχέση με τα υπόλοιπα συστήματα και στις δυο βάσεις δεδομένων. Η ποιότητα απαντήσεων στο σύστημα Banks και στις δυο περιπτώσεις παρότι ξεκινάει από ένα σχετικά καλό επίπεδο, καταλήγει στην τελευταία θέση. Ακολουθούν τα συστήματα Bidirectional και Banks με το σύστημα Discover να έχει την καλύτερη απόδοση.

Από τη διαφορά του δείκτη ποιότητας μεταξύ των δυο βάσεων δεδομένων συμπεραίνουμε ότι από τη βάση δεδομένων IMDB ανακτώνται απαντήσεις μικρότερου μεγέθους από τη βάση δεδομένων DBLP. Το γεγονός αυτό συνδέεται με τους μεγαλύτερους χρόνους εκτέλεσης που υπάρχουν για τη βάση δεδομένων DBLP (Κεφάλαιο 4.4) από τα περισσότερα συστήματα αναζήτησης διότι για την εύρεση απαντήσεων μεγαλύτερου μεγέθους χρειάζεται περισσότερος χρόνος.

4.8. Αθροιστική διαφορά kw-κόμβων

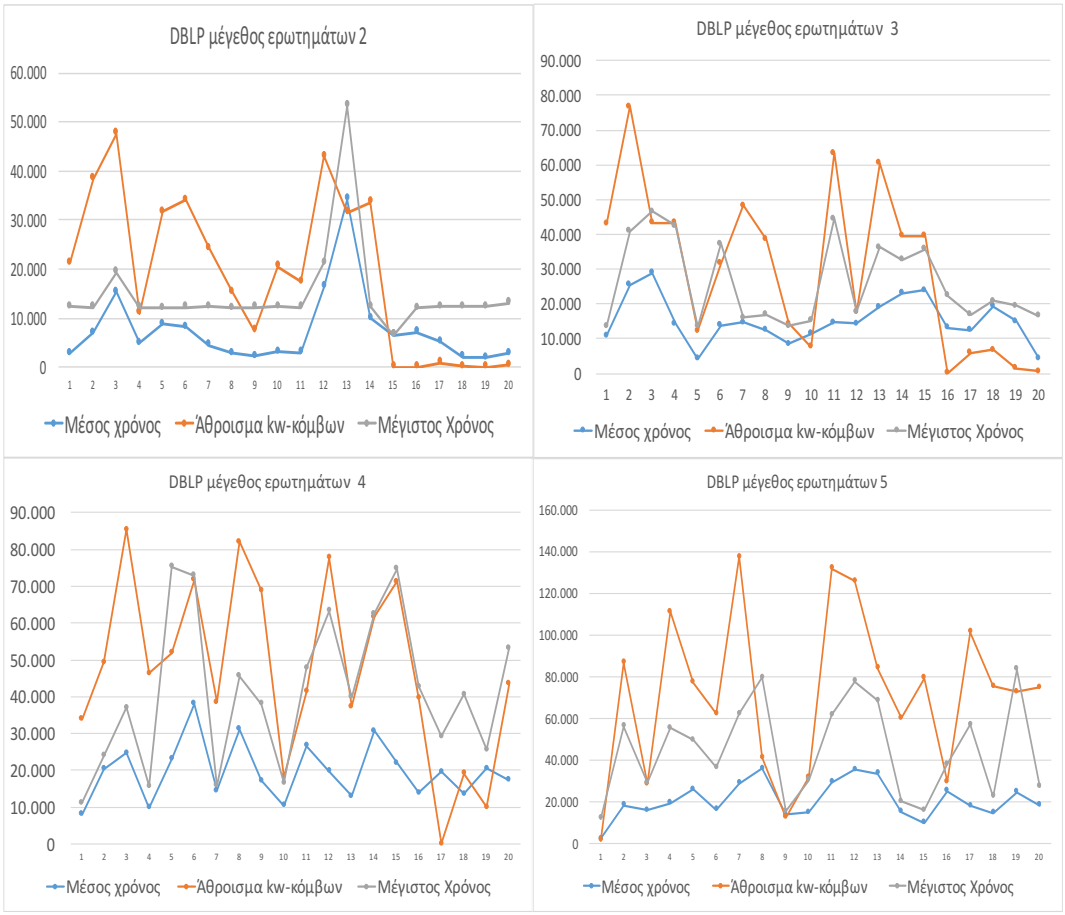
Κάνουμε μια προσπάθεια για να βρεθεί κάποιος βαθμός συσχέτισης της δυσκολίας ορισμένων ερωτημάτων στο να απαντηθούν γρήγορα. Συγκρίνουμε το πλήθος των kw-κόμβων των απαντήσεων του κάθε ερωτήματος σε σχέση με τους χρόνους εκτέλεσης όλων των συστημάτων ταυτόχρονα. Στο Σχήμα 4.14 βλέπουμε τις

μετρήσεις για τη βάση δεδομένων DBLP δείχνοντας σε κάθε σχήμα και το διαφορετικό μήκος ερωτημάτων.

Ο κάθετος άξονας αφορά ταυτόχρονα δυο μονάδες μέτρησης:

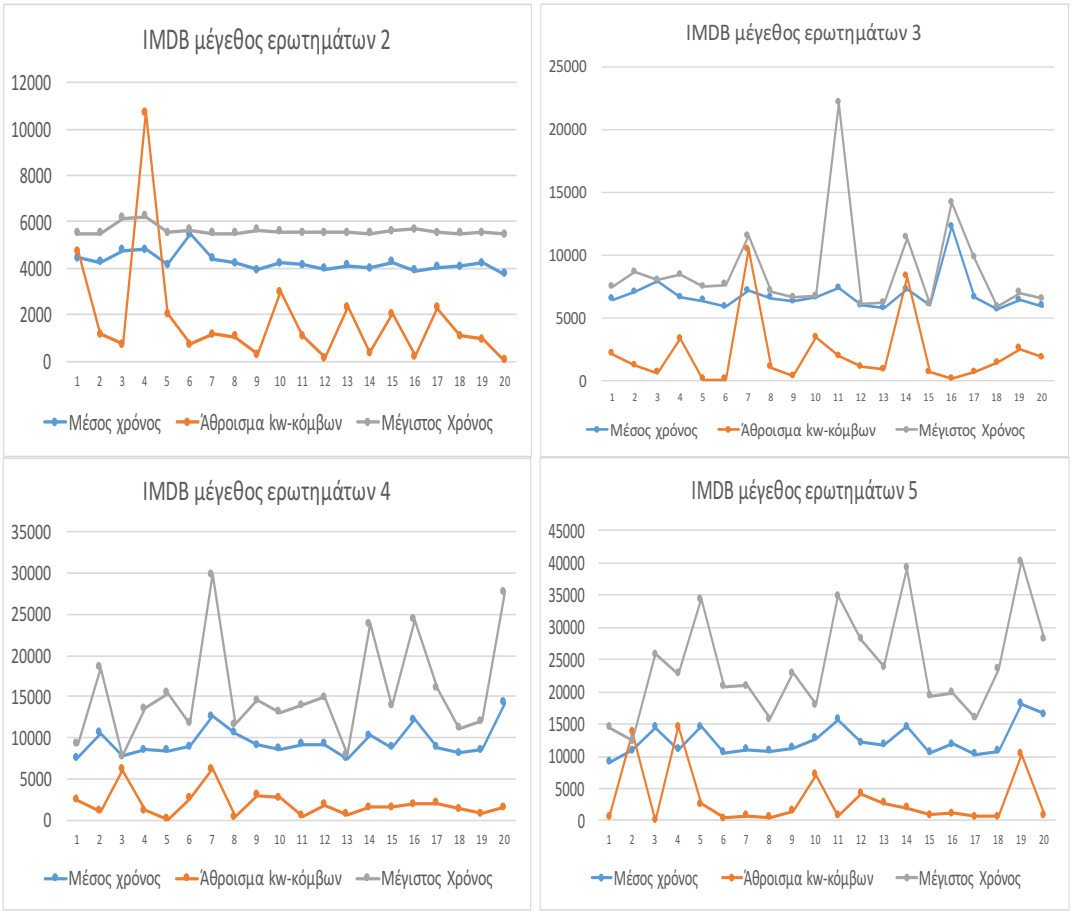
α) Για τον μέσο και μέγιστο χρόνο (γκρι και μπλε γραμμή) δείχνει το χρόνο εκτέλεσης σε χιλιάδες milliseconds

β) Για το άθροισμα των kw-κόμβων δείχνει το πλήθος των εγγραφών τους στη βάση δεδομένων.



Σχήμα 4.14 DBLP – Σύγκριση μέσου χρόνου εκτέλεσης.

Στο Σχήμα 4.15 βλέπουμε τις μετρήσεις για τη βάση δεδομένων IMDB δείχνοντας σε κάθε σχήμα και το διαφορετικό μήκος ερωτημάτων.



Σχήμα 4.15 IMDB – Σύγκριση μέσου χρόνου εκτέλεσης.

Παρατηρούμε ότι στις περισσότερες περιπτώσεις στις οποίες σε ερωτήματα ίδιου μεγέθους υπάρχουν κάποια που έχουν μεγαλύτερους χρόνους εκτέλεσης στα οποία αντιστοιχεί και αυξημένο πλήθος kw-κόμβων. Το ίδιο συμβαίνει και στις δυο βάσεις δεδομένων και είναι περισσότερο εμφανές στη βάση δεδομένων DBLP στην οποία το πλήθος των kw-κόμβων των περισσότερων ερωτημάτων είναι αρκετά μεγαλύτερο από αυτό της DBLP. Η παρατήρηση αυτή επιβεβαιώνει την υποψία ότι όσο αυξάνεται ο χώρος αναζήτησης (searching space) τόσο αυξάνεται και ο απαιτούμενος χρόνος εκτέλεσης των ερωτημάτων. Επίσης η διαφορά αυτή είναι μια ένδειξη ότι στη βάση δεδομένων DBLP υπάρχουν πολλές λέξεις-κλειδιά οι οποίες επαναλαμβάνονται σε μεγάλο βαθμό, κάτι που συνεπάγεται σε αυξημένο χώρο αναζήτησης και επομένως δικαιολογεί τους διαφορετικούς χρόνους εκτέλεσης που

χρειάζονται τα συστήματα αναζήτησης για να εξάγουν απαντήσεις μεταξύ των δυο βάσεων δεδομένων.

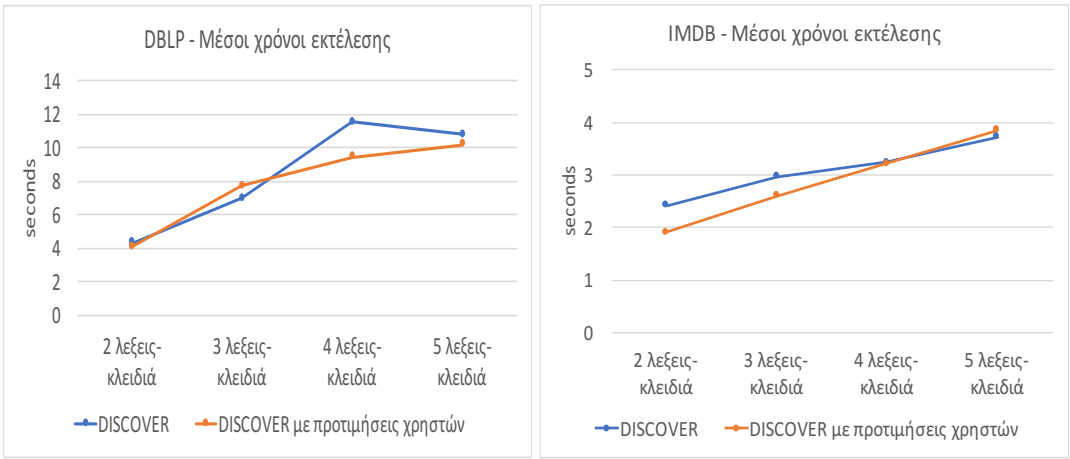
Με αυτό τον τρόπο μπορεί να χαρακτηριστεί ένα ερώτημα ως πιθανόν χρονοβόρο να εκτελεστεί καθώς ο έλεγχος των kw-κόμβων του γίνεται στο ευρετήριο της βάσης δεδομένων πριν από την εκτέλεσή του. Επομένως ανάλογα με τη βάση δεδομένων που εξετάζουμε κάθε φορά, μπορεί σε μελλοντικές προσεγγίσεις να οριστεί κάποιο όριο πλήθους kw-κόμβων με το οποίο το σύστημα αναζήτησης να χαρακτηρίζει ένα ερώτημα ως πιθανόν χρονοβόρο στο να απαντηθεί και να προταθούν στον χρήστη διορθώσεις στις λέξεις-κλειδιά του ερωτήματος ή να υπάρξουν διαφορετικές προσεγγίσεις.

4.9. Σύγκριση Discover και Discover με προτιμήσεις

Συγκρίνουμε το σύστημα Discover με το σύστημα Discover με προτιμήσεις χρηστών που δημιουργήσαμε σε διάφορα επίπεδα απόδοσης, όπως τους χρόνους εκτέλεσης, την με προτιμήσεις τομή και τις αποτυχίες των απαντήσεων και μετράμε την ποιότητα των απαντήσεων που επιστρέφουν βάσει της μετρικής που θέσαμε στο Κεφάλαιο 4.7.

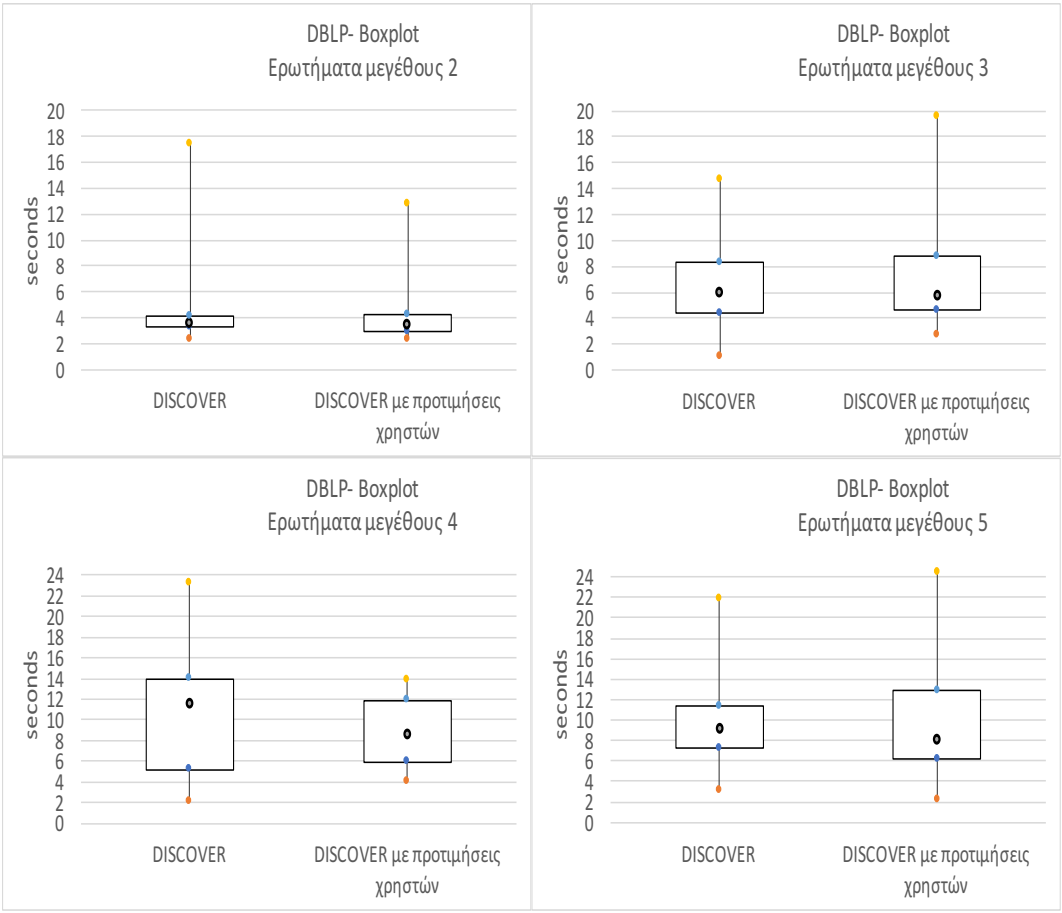
Χρόνοι εκτέλεσης

Στο Σχήμα 4.16 βλέπουμε τους μέσους χρόνους εκτέλεσης των ερωτημάτων για όλα τα μεγέθη ερωτημάτων και για τις δυο βάσεις δεδομένων.



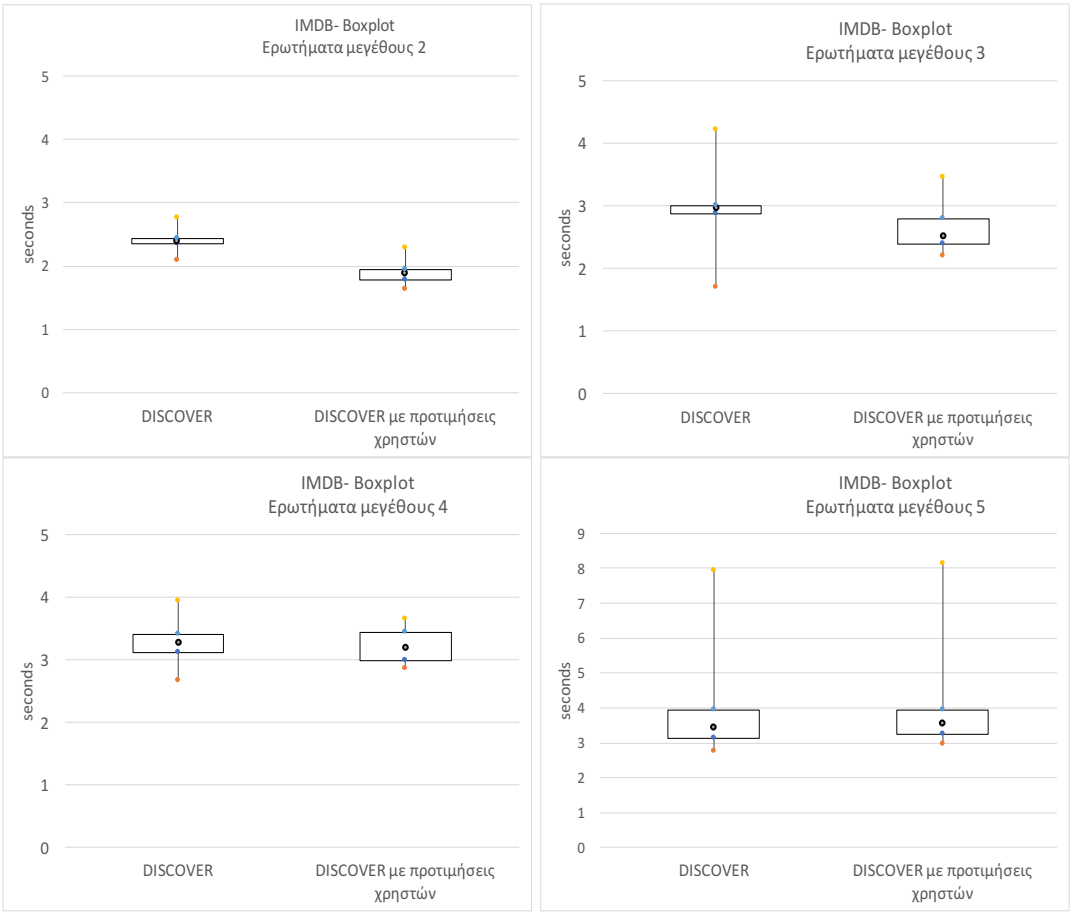
Σχήμα 4.16 Μέσοι χρόνοι εκτέλεσης Discover και Discover με προτιμήσεις χρηστών.

Στο Σχήμα 4.17 βλέπουμε τους χρόνους εκτέλεσης σε μορφή boxplot για κάθε μήκος ερωτημάτων για τη βάση δεδομένων DBLP.



Σχήμα 4.17 DBLP –Boxplot χρόνων εκτέλεσης Discover και Discover με προτιμήσεις χρηστών.

Στο Σχήμα 4.18 βλέπουμε τους χρόνους εκτέλεσης σε μορφή boxplot για κάθε μήκος ερωτημάτων για τη βάση δεδομένων IMDB.



Σχήμα 4.18 IMDB – Χρόνοι εκτέλεσης Discover και Discover με προτιμήσεις χρηστών.

Όπως είδαμε από τους χρόνους εκτέλεσης των αναζητήσεων και στις δυο βάσεις δεδομένων τα δυο συστήματα έχουν μικρές διαφορές μεταξύ τους. Αυτό οφείλεται στο γεγονός ότι χρησιμοποιούν παρόμοιο τρόπο λειτουργίας αλλά διαφορετικό τρόπο για την επιλογή των λέξεων-κλειδιών με τις οποίες ξεκινάνε την αναζήτηση και ταξινόμησης των τελικών απαντήσεων.

Τομή απαντήσεων

Στον Πίνακα 7 βλέπουμε την τομή των απαντήσεων μεταξύ των δυο συστημάτων αναζήτησης.

Πίνακας 7 Τομή απαντήσεων Discover και Discover με προτιμήσεις χρηστών.

τομή απαντήσεων	DBLP	IMDB
2 λέξεις-κλειδιά	27%	38%
3 λέξεις-κλειδιά	23%	41%
4 λέξεις-κλειδιά	23%	40%
5 λέξεις-κλειδιά	21%	37%
Μέσος όρος τομής	24%	39%

Παρατηρούμε ότι τα δυο συστήματα αναζήτησης δεν επιστρέφουν υψηλό ποσοστό όμοιων απαντήσεων μεταξύ τους. Το γεγονός αυτό οφείλεται στη διαφορετική μέθοδο με την οποία ξεκινά την αναζήτηση των απαντήσεων το κάθε σύστημα αναζήτησης και στο ότι το σύστημα *Discover με προτιμήσεις χρηστών* κάνει χρήση μοναδικότητας απαντήσεων (diversity) απορρίπτοντας απαντήσεις με όμοιες εγγραφές.

Αποτυχίες ανάκτησης απαντήσεων

Στον Πίνακα 8 βλέπουμε τις αποτυχίες των απαντήσεων μεταξύ των δυο συστημάτων αναζήτησης και για τις δυο βάσεις δεδομένων.

Πίνακας 8 Αποτυχίες απαντήσεων Discover και Discover με προτιμήσεις χρηστών.

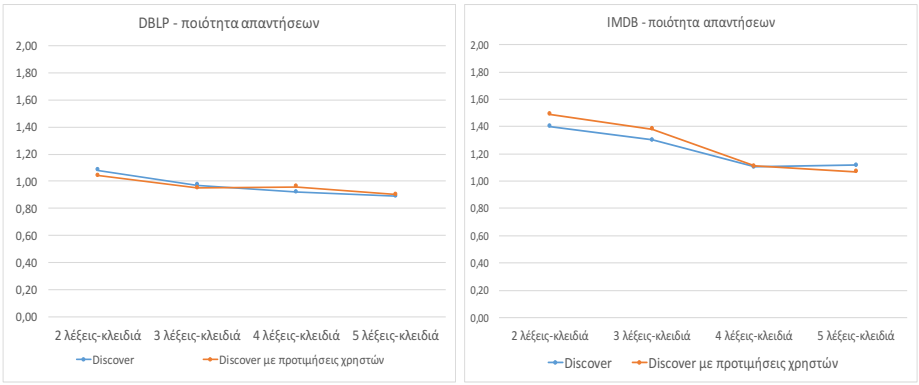
	DBLP		IMDB	
Μέγεθος ερωτημάτων	Discover	Discover με προτιμήσεις χρηστών	Discover	Discover με προτιμήσεις χρηστών
2 λέξεις-κλειδιά	1	0	0	0
3 λέξεις-κλειδιά	1	1	0	0

4 λέξεις-κλειδιά	1	1	0	0
5 λέξεις-κλειδιά	0	0	0	0
Συνολικές αποτυχίες	3	2	0	0
Ποσοστό αποτυχιών	96,25%	97,50%	100,00%	100,00%

Τα ποσοστά αποτυχιών κυμαίνονται σε πολύ χαμηλά επίπεδα και στις δυο βάσεις δεδομένων και αυτό το γεγονός δείχνει την αξιοπιστία και των δυο συστημάτων αναζήτησης.

Ποιότητα απαντήσεων

Στο Σχήμα 4.19 βλέπουμε τα αποτελέσματα μέτρησης της ποιότητας των απαντήσεων των δυο συστημάτων, η οποία βασίζεται στη μέθοδο μέτρησης που είδαμε στο Κεφάλαιο 4.7.



Σχήμα 4.19 Ποιότητα απαντήσεων Discover και Discover με προτιμήσεις χρηστών.

Από τη σύγκριση ποιότητας βλέπουμε ότι δεν υπάρχουν μεγάλες διαφορές μεταξύ των δυο συστημάτων αναζήτησης. Οι ελάχιστες διαφορές οι οποίες υπάρχουν οφείλονται στο γεγονός ότι τα δυο συστήματα δεν επιστρέφουν τις ίδιες απαντήσεις .

Καταλήγουμε στο συμπέρασμα ότι και τα δυο συστήματα που είδαμε αποδίδουν αρκετά καλά αποτελέσματα σε ικανοποιητικούς χρόνους και ο σημαντικότερος

παράγοντας που συμβαίνει αυτό είναι το εξαιρετικά μειωμένο μέγεθος του γράφου δεδομένων που χρησιμοποιούν σε σχέση με τα υπόλοιπα συστήματα που εξετάσαμε. Η διαφορά στο μέγεθος του γράφου δεδομένων τους φαίνεται από το πλήθος των kw-κόμβων που είδαμε και στο Κεφάλαιο 4.8 σε σχέση με το πλήθος όλων των κόμβων της κάθε βάσης δεδομένων.

ΚΕΦΑΛΑΙΟ 5. ΣΥΜΠΕΡΑΣΜΑΤΑ

5.1 Συμπεράσματα και μελλοντικές επεκτάσεις

5.1. Συμπεράσματα και μελλοντικές επεκτάσεις

Από τις πειραματικές συγκρίσεις των συστημάτων αναζήτησης που εξετάσαμε προκύπτουν αρκετά σημαντικά συμπεράσματα. Είδαμε ότι η αναζήτηση με λέξεις-κλειδιά σε σχεσιακές βάσεις δεδομένων έχει ως αποτέλεσμα να ανακτώνται απαντήσεις με μεγάλη σχετικότητα προς τα ερωτήματα σε σχετικά ικανοποιητικούς χρόνους. Σημαντικό παράγοντα στον χρόνο εκτέλεσης των ερωτημάτων φαίνεται ότι αποτελεί εκτός από το μέγεθός τους και το μέγεθος του γράφου δεδομένων που χρησιμοποιούν, καθώς ο χώρος αναζήτησης που επεξεργάζονται είναι αρκετά μικρότερος, κάτι που φαίνεται στους χρόνους εκτέλεσης των αναζητήσεων.

Από τους μέσους χρόνους εκτέλεσης των ερωτημάτων φαίνεται ότι η ανάκτηση των απαντήσεων εξαρτάται από το περιεχόμενο του κάθε ερωτήματος αλλά και από το είδος των εγγραφών της βάσης δεδομένων. Αυτή η παρατήρηση επιβεβαιώνεται από το γεγονός ότι δεν υπάρχει συγκεκριμένο μέγεθος ερωτημάτων με διαφορετικές λέξεις-κλειδιά και για τα οποία να υπάρχουν παρόμοιοι χρόνοι εκτέλεσης, καθώς για ερωτήματα ίδιου μεγέθους υπάρχουν περιπτώσεις που το σύνολο των kw-κόμβων τους κυμαίνεται από μερικές εκατοντάδες μέχρι και αρκετές δεκάδες χιλιάδες. Σε πολλά από τα ερωτήματα οι χρόνοι αναζήτησης αυξάνονται όταν αυξάνεται το μέγεθός τους αλλά και όταν αυξάνεται το πλήθος των kw-κόμβων τους. Το γεγονός αυτό προσδίδει στα συστήματα αναζήτησης με λέξεις-κλειδιά την

εικόνα της μη προβλέψιμης απόδοσης και επιβεβαιώνει την ανάγκη για την επιπλέον βελτίωσή τους αλλά και τη δημιουργία νέων προσεγγίσεων στο μέλλον.

Βασιζόμενοι στη μέθοδο μέτρησης ποιότητας που είδαμε στο Κεφάλαιο 4.7, παρατηρήσαμε για όλα τα συστήματα αναζήτησης ότι όταν οι χρόνοι εκτέλεσης είναι αυξημένοι τότε στην πλειοψηφία των περιπτώσεων χαμηλώνει ο δείκτης ποιότητας των απαντήσεων που ανακτώνται. Επομένως ο χρόνος εκτέλεσης μιας αναζήτησης σχετίζεται άμεσα με την ποιότητα των απαντήσεων που θα ανακτηθούν, κάτι που επιβεβαιώνει τα συμπεράσματα αντίστοιχων ερευνών [3, 8, 20, 36]. Σημαντικό επομένως είναι από την πλευρά των χρηστών να προσπαθούν να δημιουργούν ερωτήματα με λίγες και όσο το δυνατόν αντιπροσωπευτικότερες λέξεις-κλειδιά με τις απαντήσεις που αναζητούν.

Για την παρούσα εργασία υπάρχουν διάφορες μελλοντικές επεκτάσεις που μπορούν να υλοποιηθούν και ιδιαίτερα στο κομμάτι της βαθμολόγησης και ταξινόμησης των απαντήσεων που ανακτώνται. Μερικά παραδείγματα είναι να συνδυαστούν συστήματα τεχνητής νοημοσύνης ώστε να «εκπαιδεύονται» οι συναρτήσεις απόδοσης βαθμολογίας για να ταξινομούν δικαιότερα τις απαντήσεις ανάλογα με το μέγεθός τους και με το πλήθος των kw-κόμβων του στον γράφο δεδομένων. Να εφαρμοστούν μέθοδοι ελέγχου και διόρθωσης της σειράς ταξινόμησης των τελικών απαντήσεων οι οποίες θα βασίζονται στο μέγεθος και τη σειρά κατάταξής τους. Ενδιαφέρον αποτελεί η ανάπτυξη μεθόδων για τη μέτρηση της δημοτικότητας των λέξεων-κλειδιών ανάλογα με τον ρυθμό ζήτησής τους από τους χρήστες αλλά και ανάλογα με το πλήθος τους στη βάση δεδομένων, ώστε να δίνεται η δυνατότητα στο σύστημα αναζήτησης να προτείνει τις δημοφιλέστερες ή παρόμοιες λέξεις-κλειδιά οι οποίες μοιάζουν με αυτές που πληκτρολογεί ο χρήστης αλλά λιγότερους kw-κόμβους. Τέλος, ενδιαφέρον αποτελεί να εξεταστεί η απόδοση του συνδυασμού του ευρετηρίου δυο επιπέδων του συστήματος Blinks με κάποιο σύστημα αναζήτησης βασισμένο στις προτιμήσεις χρηστών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Sanjay Agrawal, Surajit Chaudhuri, and Gautam Das. “DBXplorer: A System for Keyword-based Search over Relational Databases”. In Proceedings of the 18th International Conference on Data Engineering, 26 February - 1 March 2002, San Jose, CA, pp 5–16. IEEE Computer Society, 2002.

[2] Ricardo Baeza-Yates, Berthier Ribeiro-Neto. “Modern Information Retrieval”. Addison-Wesley, 1999.

[3] G. Bhalotia, C. Nakhe, A. Steiner i, S. Chakrabarti, and S. Sudarshan. “Keyword searching and browsing in databases using BANKS”. In ICDE, 2002.

[4] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. “Keyword searching and browsing in databases using BANKS”. Technical report, Indian Institute of Technology, Bombay, November 2001.

[5] S. Brin, L. Page. “The anatomy of a large-scale hypertextual Web search engine”. Computer Networks and ISDN Systems, 30. Pp 107-117. Computer Science, Stanford University, 1998.

[6] Deepayan Chakrabarti, Christos Faloutsos. “graph Mining: Laws, Tools, and Case Studies”. ISBN 9781608451159, 9781608451166, www.morganclaypool.com.

[7] Wai-Kai Chen. "Graph Theory and Its Engineering Applications". pp 320-321. Univ. Illinois, Chicago. Feb. 1997.

[8] Joel Coffman, Alfred C. Weaver. “An Empirical Performance Evaluation of Relational Keyword Search Systems”. Computer Science Department, University of Virginia, Charlottesville, VA, USA, 2014.

[9] Joel Coffman, Alfred C. Weaver. "Learning to Rank Results in Relational Keyword Search". Glasgow, Scotland, UK, CIKM-2011.

[10] B.B. Dalvi, M. Kshirsagar, S. Sudarshan. “Keyword search on external memory graphs”. In Proceedings of the VLDB Endowment, Vol 1, Issue 1, 2008.

[11] B. Ding, J.X. Yu, S. Wang, L. Qing, X. Zhang, and X. LIN. “Finding top-k min-cost connected trees in databases”. In ICDE, 2007.

[12] Smita S. Dusane. “Relational Keyword Search Systems and Empirical Performance Evaluation”. Siddhant College of Engineering, Sudumbare, Pune, India. 2014 IJIRT, Volume 1 Issue 6, ISSN: 2349-6002.

[13] E.W. Dijkstra. “A note on two problems in connection with graphs”. Numerische Mathematik, Volume 1, pp. 269-271, 1959.

[14] D. Fallows, “Search Engine Use,” Pew Internet and American Life Project, Tech. Rep., August 2008,
http://www.pewinternet.org/Reports/ 2008/Search-Engine-Use.aspx.

[15] M. Garey, D. Johnson, and L. Stockmeyer. “Some simplified NP-complete graph problems”, in “Theoretical Computer Science”, pp 237-267, 1976.

[16] Gou Gang with Prof. Rada Chirkova. Efficient Algorithms for Querying Large-scale Data in Relational, XML, and graph-structured Data Repositories. North Carolina State University. ProQuest, 2008. ISBN: 0549820914, 9780549820918.

[17] Page, L., Brin, S., Motwani, R., & Winograd, T. “The PageRank citation ranking: bringing order to the web”. 1999.

[18] Arvind Hulgeri, Gaurav Bhalotia, Charuta Nakhe, Soumen Chakrabarti. “Keyword Search in Databases”. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering . 2001.

[19] H. He, H. Wang, J. Yang, and P. S. Yu. “BLINKS: Ranked keyword searches on graphs”. In SIGMOD, 2007.

[20] Vagelis Hristidis & Yannis Papakonstantinou. “Discover: Keyword Search in Relational Databases”. Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002.

[21] Vagelis Hristidis, Nick Koudas, Yannis Papakonstantinou, Divesh Srivastava. Keyword Proximity Search in XML Trees. IEEECS. 7 Feb. 2006.

[22] Jeff Huang. “Search Engine Query Logs available for download”. Brown University, Rhode Island, USA.
http://jeffhuang.com/search_query_logs.html.

[23] Varun Kacholia, Shashank Pandit, Soumen Chakrabarti S. Sudarshan, Rushi Desai & Hrishikesh Karambelkar. “Bidirectional Expansion For Keyword Search on graph Databases”. Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005.

[24] G. Karypis and V. Kumar. “METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering”. In Supercomputing, 1995.

[25] R.M. Karp. “Reducibility among combinatorial problems”. Part of the series “The IBM Research Symposia Series” pp 85-103. IBM Thomas J. Watson Research Center, Yorktown Heights, New York. March 20–22, 1972.

[26] B. Kimelfeld, Y. Sagiv. “Finding and approximating top-k answers in keyword proximity search”. In PODS, pages 173-182, 2006.

[27] Jure Leskovec. SNAP library. University of Stanford. https://snap.stanford.edu/data/com-DBLP.html

[28] Michael Ley. "DBLP - Some Lessons Learned". Proceedings of the VLDB Endowment, 18 June 2009.

[29] Guoliang Li, Beng Chin Ooi, Jianhua Feng, Jianyong Wang, Lizhu Zhou. "EASE: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data". In Wang [Wan08], pages 903–914.

[30] Lyes Limam, David Coquil, Harald Kosch, Lionel Brunie. Extracting user interests from search query logs: A clustering approach. Proceeding. DEXA '10 Proceedings of the 2010 Workshops on Database and Expert Systems Applications. Pages 5-9.

[31] Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze. "An Introduction to Information Retrieval".Online edition (c), 2009 Cambridge UP.

[32] C. D. Manning, P. Raghavan, H. Schutze, Introduction to Information Retrieval. New York, NY: Cambridge University Press, 2008.

[33] Oracle. "Database SQL Language Reference - Contents". https://docs.oracle.com/cd/B28359_01/server.111/b28286/toc.htm

[34] Ken Q. Pu, Xiaohui Yu. Keyword Query Cleaning. PVLDB '08, August 23-28, 2008, Auckland, New Zealand. Copyright 2008 VLDB Endowment, ACM 978-1-60558-305-1/08/08

[35] Kristen Purcell, Joanna Brenner, Lee Rainie. "Search Engine Use 2012", Pew Research Center, March 9, 2012, http://www.pewinternet.org/files/old-media//Files/Reports/2012/PIP_Search_Engine_Use_2012.pdf

[36] Kostas Stefanidis, Marina Drosou & Evaggelia Pitoura. "PerK: Personalized Keyword Search in Relational Databases through Preferences". EDBT 2010, March 22–26, 2010, Lausanne, Switzerland.

[37] W. Webber, A. Moffat, J. Zobel, “Statistical Power in Retrieval Experimentation,” in CIKM ’08: Proceeding of the 17th ACM International Conference on Information and Knowledge Management, pp. 571–580, 2008.

[38] Jeffrey Xu Yu, Lu Qin, Lijun Chang. "Keyword Search in Databases. Synthesis Lectures on Data Management". Number Lecture 1. Morgan and Claypool Publishers, 2010. ISBN: 160845195X

[39] E. M. Voorhees, “The Philosophy of Information Retrieval Evaluation,” in CLEF 2001: Revised Papers from the Second Workshop of the Cross-Language Evaluation Forum on Evaluation of Cross-Language Information Retrieval Systems. Springer-Verlag, pp. 355–370, 2002.

Διαδικτυακές πηγές

Πηγές ερωτημάτων

[40] The Lemur Project, “The ClueWeb09 Dataset”.
http://lemurproject.org/clueweb09/.

[41] Jeff Huang, Computer Science at Brown University
http://jeffhuang.com/search_query_logs.html
http://www.cim.mcgill.ca/~dudek/206/Logs/AOL-user-ct-collection/

[42] Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. The web09-bst Dataset.
(http://lemurproject.org/clueweb09/)
(http://boston.lti.cs.cmu.edu/Data/web08-bst/planning.html/)

Πηγές βάσεων δεδομένων

[43] http://dblp.uni-trier.de/

[44] http://www.imdb.com/stats

[45] http://www.imdb.com

Άλλες διαδικτυακές πηγές

[46] https://en.wikipedia.org/wiki/Iterator#Java

[47] https://en.wikipedia.org/wiki/Jaccard_index

[48] https://en.wikipedia.org/wiki/Tree_(graph_theory)

[49] https://en.wikipedia.org/wiki/Breadth-first_search

ΠΑΡΑΡΤΗΜΑ Α

Πίνακες ερωτημάτων

5.2. Α.1 Πίνακες ερωτημάτων

Στο παράρτημα αυτό παραθέτουμε τους πίνακες με τα ερωτήματα που εκτελέστηκαν από το κάθε σύστημα αναζήτησης κατηγοριοποιημένα ανάλογα με την βάση δεδομένων. Δίπλα από κάθε ερώτημα υπάρχει το πλήθος των kw-κόμβων για την κάθε λέξη-κλειδί αντίστοιχα. Εκτελούνται είκοσι διαφορετικά ερωτήματα για κάθε μέγεθος ερωτημάτων με τέσσερα διαφορετικά μεγέθη για κάθε βάση δεδομένων.

Στους πίνακες Π.1, Π.2, Π.3 και Π.4 βλέπουμε τα ερωτήματα της βάση δεδομένων DBLP και το πλήθος των kw-κόμβων για κάθε λέξη κλειδί.

Πίνακας Π.1 Ερωτήματα μεγέθους 2.

α/α	ερωτήματα	Πλήθος kw-κόμβων ανά λέξη κλειδί	
1	database michael	13.836	7.424
2	distributed david	30.427	8.003
3	algorithm science	42.934	4.681
4	relational john	4.659	6.425
5	performance hector	31.546	115
6	software jennifer	33.486	566
7	jeff dynamic	617	23.629

8	jeffrey optimal	1.116	14.226
9	kevin statistical	1.269	6.250
10	abiteboul adaptive	4	20.510
11	jagadish optimization	15	17.470
12	micheli algorithm	11	42.934
13	kazutsugu performance	2	31.546
14	johanna software	84	33.486
15	jogh joep	1	13
16	cached cachera	82	1
17	carrier carrie	849	53
18	connell connective	81	39
19	deleting delegate	40	17
20	Giora Fernandez	11	496

Πίνακας Π.2 Ερωτήματα μεγέθους 3.

α/α	ερωτήματα	Πλήθος kw-κόμβων ανά λέξη κλειδί		
1	distributed david relational	30.427	8.003	4.659
2	algorithm implement software	42.934	398	33.486
3	algorithm implement jagadish	42.934	398	15
4	algorithm implement micheli	42.934	398	11
5	relational john kevin	4.659	6.425	1.269
6	performance hector jagadish	31.546	115	15
7	software jennifer optimal	33.486	566	14.226
8	jeff dynamic optimal	617	23.629	14.226
9	jeffrey optimal kazutsugu	116	14.226	2
10	kevin statistical johanna	1.269	6.250	84
11	abiteboul adaptive algorithm	4	20.510	42.934
12	jagadish optimization deleting	15	17.470	40
13	micheli algorithm optimization	11	42.934	17.470
14	kazutsugu performance johanna	84	33.486	5.921

15	johanna software hardware	84	33.486	5.921
16	jogh joep cached	1	13	82
17	cached cachera hardware	82	1	5.921
18	carrier carrie hardware	849	53	5.921
19	connell connective kevin	81	39	1.269
20	deleting delegate validating	40	17	473

Πίνακας Π.3 Ερωτήματα μεγέθους 4.

α/α	ερωτήματα	Πλήθος kw-κόμβων ανά λέξη κλειδί			
1	database michael david relational	13.836	7.424	8.003	4.569
2	distributed david relational statistical	30.427	8.003	4.659	6.250
3	algorithm implementing performance david	42.934	2.898	31.546	8.003
4	algorithm implementing jagadish jennifer	42.934	2.898	15	566
5	algorithm implementing micheli xml	42.934	2.898	11	6.117
6	relational john kevin networks	4.659	6.425	1.269	59.300
7	performance hector jagadish improving	31.546	115	15	6.873
8	software jennifer optimal learning	33.486	566	14.226	33.744
9	jeff dynamic optimal distributed	617	23.629	14.226	30.427
10	jeffrey optimal kazutsugu index	1.116	14.226	2	2.895
11	kevin statistical johanna learning	1.269	6.250	84	33.744
12	abiteboul adaptive algorithm optimal	4	20.510	42.934	14.226
13	jagadish optimization deleting knowledge	15	17.470	40	19.874
14	micheli algorithm optimization kevin	11	42.934	17.470	1.269
15	xml performance johanna software	6.117	31.546	84	33.486
16	johanna software hardware cached	84	33.486	5.921	82
17	jogh joep cached cachera	1	13	82	1

18	cached cachera hardware sensor	82	1	5.921	13.092
19	carrier carrie xml heuristic	849	53	6.117	2.916
20	deleting delegate validating algorithm	40	17	473	42.934

Πίνακας Π.4 Ερωτήματα μεγέθους 5.

α/α	ερωτήματα	Πλήθος kw-κόμβων ανά λέξη κλειδί				
1	Divesh Jignesh Jagadish Timber Querying	2	11	15	21	1.603
2	Wang Poor Chang software algorithm	6.715	142	3.405	33.486	42.934
3	Naughton Dewitt Chang optimization query	16	17	3.405	17.470	7.870
4	dynamic algorithm optimization query CoRR	23.269	42.934	17.470	7.870	19.655
5	performance hardware relational software querying	31.546	5.921	4.659	33.486	1.603
6	Michael Dewitt software query processing	7.424	17	33.486	7.870	13.605
7	distributed networks software validating query	30.427	59.300	33.486	473	13.605
8	distributed networks software distributed query	101	2.238	7.870	17.470	13.605
9	Microsoft recovery Chawathe David Philip	296	3.354	3	8.003	976
10	database michael david Fernandez parametric	13.836	7.424	8.003	436	2.238
11	data mining architecture software engineering	56.199	10.975	19.538	33.486	11.729
12	security simulation scheduling neural	11.461	19.495	15.236	20.372	59.300

	networks					
13	Internet optimization unsupervised learning model	9.969	17.470	1.756	3.744	51.029
14	open source software maintenance services	5.360	5.432	33.486	2.768	13.256
15	gossiping greedy algorithm unsupervised learning	177	866	42.934	1.756	33.744
16	formal languages moving objects vision	7.307	7.475	2.507	7.685	4.744
17	visual impairment wireless sensor networks	9.699	74	19.243	13.092	59.300
18	fault tolerance performance evaluation services	9.265	1.988	31.546	19.427	13.256
19	social interaction link analysis productivity	5.177	7.660	3.013	56.406	671
20	communication video streaming web services	13.585	14.629	2.756	30.636	13.256

Στους πίνακες Π.5, Π.6, Π.7 και Π.8 βλέπουμε τα ερωτήματα της βάση δεδομένων IMDB και το πλήθος των kw-κόμβων για κάθε λέξη κλειδί.

Πίνακας Π.5 Ερωτήματα μεγέθους 2.

α/α	ερωτήματα	Πλήθος kw-κόμβων ανά λέξη κλειδί	
1	matrix reloaded	4.670	33
2	TOM HANKS	1.144	12

3	williams carrey	650	76
4	John Travolta	10.694	21
5	MEN BLACK	360	1.681
6	Marathon day	8	716
7	sheila office	693	480
8	hotel california	961	90
9	boat trip	185	111
10	courtney driver	270	2.730
11	daryl mitchell	155	929
12	galaxy quest	39	87
13	gary cole	1.720	623
14	three company	239	124
15	kevin dobson	1.962	88
16	river phoenix	143	81
17	graham russell	894	1.405
18	russell hitchcock	1.045	52
19	full house	88	871
20	pride prejudice	40	5

Πίνακας Π.6 Ερωτήματα μεγέθους 3.

α/α	ερωτήματα	Πλήθος kw-κόμβων ανά λέξη κλειδί		
1	little women scary	1.752	354	44
2	lou diamond bonus	897	308	22
3	underworld evolution world	18	15	611
4	stephen king laws	1.439	1.904	22
5	Biter het audio	3	102	6
6	Firehouse juedi fink	2	1	106
7	platoon war man	42	351	10.102
8	hotel california dreaming	961	90	23
9	boat trip antony	185	111	89

10	courtney driver beach	270	2.730	460
11	alex hooser photos	1.989	2	8
12	daryl mitchell galaxy	155	929	39
13	save last dance	26	455	431
14	david murphy meteorologist	7.503	773	23
15	tiffany amber thiessen	410	307	5
16	ironed jawed angels	1	1	195
17	leonard fagot photos	702	8	8
18	carol fagot holland	1.211	8	259
19	helen blair girlfriend	1.627	335	632
20	patron saint liars	1.611	246	8

Πίνακας Π.7 Ερωτήματα μεγέθους 4.

α/α	ερωτήματα	Πλήθος kw-κόμβων ανά λέξη κλειδί			
1	Al Pacino Diane Keaton	1.462	12	976	39
2	gain video clips dream	8	636	8	423
3	pierce brosnan james bond	302	8	5.623	224
4	current picture danielle spencer	12	78	495	573
5	songs between sixteen candles	38	71	6	6
6	grim adventures billy mandy	30	225	2.120	271
7	robert "de niro" personality type	6.185	3	7	41
8	movies many people being	48	37	263	53
9	motel same birthday party	272	15	178	2.557
10	larry cable food inspector	1.709	66	120	812
11	does freddy kruger kill	49	309	66	122
12	elm street the first murders	14	974	887	28
13	star wars empire strikes	528	89	59	13
14	fushigi yugi voice cast	2	3	1.411	166
15	kingdom hearts voice actors	43	78	1.411	9

16	american idol ryan seacrest	617	34	1.325	2
17	three company wedge black	239	124	15	1.681
18	ellie may beverly hillbillies	181	787	448	2
19	kay knapp china paint	635	54	115	25
20	sean hannity date birth	1.268	2	286	27

Πίνακας Π.8 Ερωτήματα μεγέθους 5.

α/α	ερωτήματα	Πλήθος kw-κόμβων ανά λέξη κλειδί				
1	Eddy boat trip Antony Royle	235	185	111	89	14
2	Mission Impossible Detective Cruise John	103	15	2.921	61	10.694
3	mucha lucha weight gain clip	5	14	9	8	24
4	Mission Impossible Cruise John Mary	103	15	61	10.694	3.666
5	adventures billy mandy coloring pages	225	2.120	271	3	18
6	movies between people being drawn	48	71	263	53	5
7	drawn motel Pool same birthday	5	272	333	15	178
8	music valley dolls dionne warwick	278	69	36	60	97
9	entertaining angels dorothy story movies	3	195	814	490	48
10	bissinger friday night lights woman	8	71	885	27	6.156
11	altered version studio logo singing	2	328	250	3	215
12	pirate captain black beard battle	174	2.121	1.681	76	120
13	missionary former lover duke judge	11	148	338	447	1.802
14	attempted mutiny british soldier	4	6	277	1.574	129

	hero					
15	escape prison returning during killed	68	751	15	27	36
16	prime minister london england drinking	118	666	237	67	15
17	british navy adventure hero action	277	86	94	129	93
18	cannon ball explosion opening credits	142	370	7	113	14
19	man wearing poison explosion cannon	10.102	27	30	7	142
20	altered version femme fatale studio	2	328	301	19	250

ΠΑΡΑΡΤΗΜΑ Β

Τομή απαντήσεων
Συνολικές αποτυχίες συστημάτων αναζήτησης
Στατιστικά στοιχεία βάσεων δεδομένων

5.3. Β.1 Αναλυτικά στοιχεία για την τομή απαντήσεων.

Στους πίνακες Π.9, Π.10, Π.11, Π.12 και Π.13 βλέπουμε αναλυτικά την τομή απαντήσεων στη βάση δεδομένων DBLP σύμφωνα με τον αύξων αριθμό του κάθε ερωτήματος από το Παράρτημα Α:

Πίνακας Π.9 Τομή απαντήσεων - Μήκος ερωτημάτων 2.

Α/Α	Κοινές απαντήσεις	Μέσος όρος top-10	Ποσοστό μέσου όρου
1	1	10%	14,50%
2	2	20%	
3	2	20%	
4	0	0%	
5	2	20%	
6	2	20%	
7	1	10%	
8	2	20%	
9	3	30%	
10	1	10%	

11	1	10%	
12	2	20%	
13	1	10%	
14	2	20%	
15	0	0%	
16	0	0%	
17	1	10%	
18	2	20%	
19	2	20%	
20	2	20%	

Πίνακας Π.10 Τομή απαντήσεων - Μήκος ερωτημάτων 3.

Α/Α	Κοινές απαντήσεις	Μέσος όρος top-10	Ποσοστό μέσου όρου
1	1	10%	10,50%
2	1	10%	
3	3	30%	
4	0	0%	
5	1	10%	
6	3	30%	
7	2	20%	
8	2	20%	
9	0	0%	
10	1	10%	
11	0	0%	
12	0	0%	
13	1	10%	
14	2	20%	
15	1	10%	
16	0	0%	

17	0	0%	
18	0	0%	
19	2	20%	
20	1	10%	

Πίνακας Π.11 Τομή απαντήσεων - Μήκος ερωτημάτων 4.

Α/Α	Κοινές απαντήσεις	Μέσος όρος top-10	Ποσοστό μέσου όρου
1	1	10%	8,00%
2	1	10%	
3	2	20%	
4	0	0%	
5	0	0%	
6	2	20%	
7	0	0%	
8	2	20%	
9	2	20%	
10	0	0%	
11	3	30%	
12	0	0%	
13	0	0%	
14	3	30%	
15	0	0%	
16	0	0%	
17	0	0%	
18	0	0%	
19	0	0%	
20	0	0%	

Πίνακας Π.12 Τομή απαντήσεων - Μήκος ερωτημάτων 5.

Α/Α	Κοινές απαντήσεις	Μέσος όρος top-10	Ποσοστό μέσου όρου
1	2	20%	10%
2	0	0%	
3	0	0%	
4	0	0%	
5	1	10%	
6	2	20%	
7	1	10%	
8	2	20%	
9	0	0%	
10	0	0%	
11	1	10%	
12	1	10%	
13	2	20%	
14	2	20%	
15	0	0%	
16	2	20%	
17	0	0%	
18	0	0%	
19	2	20%	
20	1	10%	

Πίνακας Π.13 Η τομή των απαντήσεων.

DBLP		
Μήκος ερωτημάτων	Μέσος όρος τομής	Συνολική τομή

2	14,50%	10,75%
3	10,50%	
4	8,00%	
5	10,00%	

Στους πίνακες Π.14, Π.15, Π.16, Π.17 και Π.18 βλέπουμε αναλυτικά την τομή απαντήσεων στη βάση δεδομένων IMDB σύμφωνα με τον αύξων αριθμό του κάθε ερωτήματος από το Παράρτημα Α:

Πίνακας Π.14 Τομή απαντήσεων - Μήκος ερωτημάτων 2.

Α/Α	Κοινές απαντήσεις	Μέσος όρος top-10	Ποσοστό μέσου όρου
1	2	20%	16,00%
2	0	0%	
3	3	30%	
4	2	20%	
5	1	10%	
6	0	0%	
7	3	30%	
8	3	30%	
9	2	20%	
10	2	20%	
11	1	10%	
12	0	0%	
13	2	20%	
14	2	20%	
15	1	10%	
16	3	30%	

17	2	20%	
18	1	10%	
19	2	20%	
20	0	0%	

Πίνακας Π.15 Τομή απαντήσεων - Μήκος ερωτημάτων 3.

Α/Α	Κοινές απαντήσεις	Μέσος όρος top-10	Ποσοστό μέσου όρου
1	2	20%	12,50%
2	2	20%	
3	2	20%	
4	1	10%	
5	2	20%	
6	1	10%	
7	0	0%	
8	1	10%	
9	2	20%	
10	2	20%	
11	0	0%	
12	2	20%	
13	1	10%	
14	0	0%	
15	2	20%	
16	0	0%	
17	1	10%	
18	1	10%	
19	2	20%	
20	1	10%	

Πίνακας Π.16 Τομή απαντήσεων - Μήκος ερωτημάτων 4.

Α/Α	Κοινές απαντήσεις	Μέσος όρος top-10	Ποσοστό μέσου όρου
1	1	10%	8,50%
2	0	0%	
3	0	0%	
4	1	10%	
5	0	0%	
6	2	20%	
7	0	0%	
8	2	20%	
9	1	10%	
10	3	30%	
11	2	20%	
12	1	10%	
13	0	0%	
14	0	0%	
15	1	10%	
16	0	0%	
17	1	10%	
18	0	0%	
19	2	20%	
20	0	0%	

Πίνακας Π.17 Τομή απαντήσεων - Μήκος ερωτημάτων 5.

Α/Α	Κοινές απαντήσεις	Μέσος όρος top-10	Ποσοστό μέσου όρου
1	2	20%	11.00%
2	3	30%	
3	0	0%	
4	3	30%	
5	0	0%	
6	0	0%	
7	1	10%	
8	1	10%	
9	1	10%	
10	0	0%	
11	1	10%	
12	0	0%	
13	2	20%	
14	2	20%	
15	1	10%	
16	2	20%	
17	1	10%	
18	1	10%	
19	1	10%	
20	0	0%	

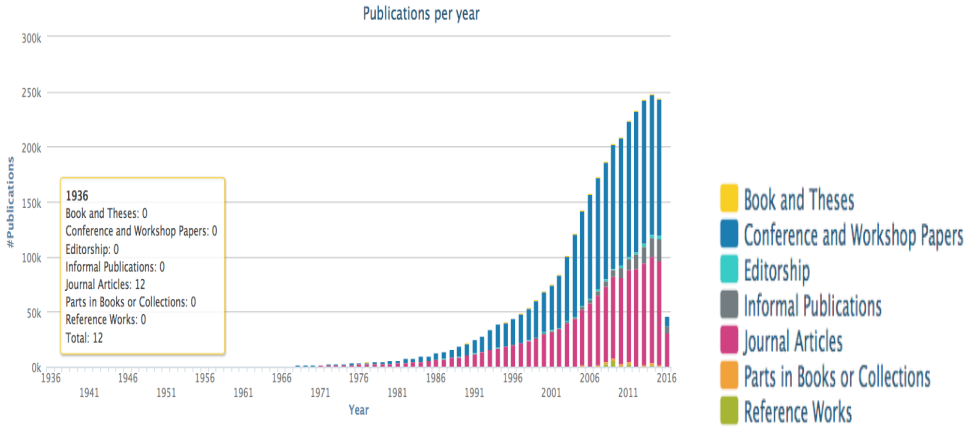
Πίνακας Π.18 Η τομή των απαντήσεων.

IMDB		
Μήκος ερωτημάτων	Μέσος όρος	Συνολική τομή

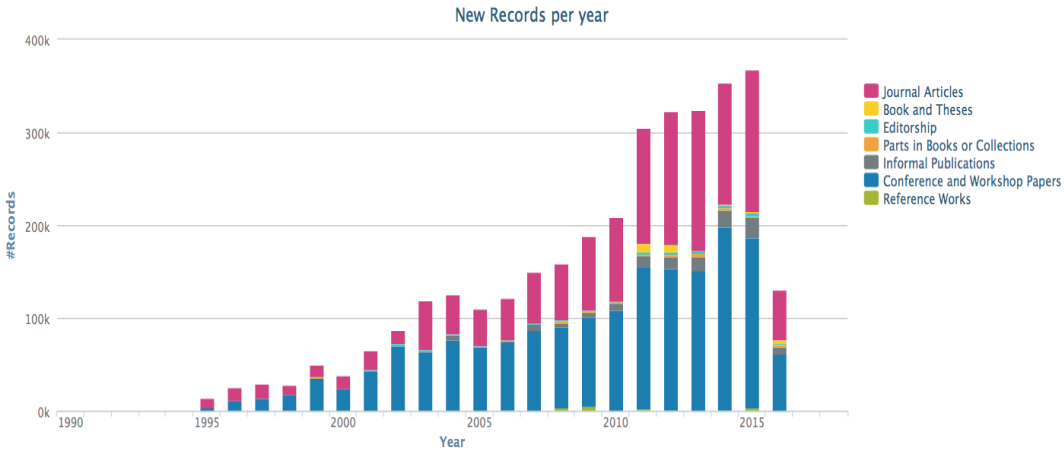
2	16,00%	12,00%
3	12.50%	
4	8.50%	
5	11,00%	

5.4. Β.2 Στατιστικά στοιχεία των βάσεων δεδομένων

Όπως βλέπουμε στο Σχήμα 6.1 τα τελευταία περίπου 30 έτη έχει συμβεί μια αλματώδης ανάπτυξη των εγγραφών της βάσης δεδομένων DBLP, η οποία αυξάνεται με μεγάλο ρυθμό σε κάθε έτος.



Σχήμα 0.1 Η εξέλιξη της βάσης δεδομένων DBLP μέχρι σήμερα.



Σχήμα 0.2 Οι εγγραφές που προστίθενται στη βάσης δεδομένων DBLP κάθε χρόνο.

Στον Πίνακα Π.19 βλέπουμε τα στατιστικά στοιχεία για τη βάση δεδομένων IMDB.

Πίνακας Π.19 Κατηγορίες ταινιών.

IMDB	
τίτλοι: 3.738.056	Χρονική περίοδος: 1874 - 2015
2.397.751	TV episode
415.377	short
347.447	feature
101.685	TV series
96.128	video
81.743	TV movie
80.135	documentary short
71.462	documentary
34.927	video short
28.347	TV documentary
16.562	video game
13.26	TV series documentary
12.428	TV mini-series
11.956	video documentary short
10.681	video documentary
5.826	TV special
4.884	TV short
2.361	TV mini-series documentary
1.859	TV documentary short
1.483	TV special documentary
1.255	TV series short
219	TV mini-series short

109	TV series documentary short
83	TV special documentary short
39	TV special short
20	video game documentary
16	video game short
13	TV mini-series documentary short

ΠΑΡΑΡΤΗΜΑ Γ

Ψευδοκώδικες των συστημάτων αναζήτησης

5.5. Γ.1 Ψευδοκώδικας των συστημάτων αναζήτησης

5.5.1. Banks

Βλέπουμε τον ψευδοκώδικα του συστήματος Banks και την ανάλυση των βασικών βημάτων λειτουργίας του:

Λειτουργία Banks σε βήματα

Περιγράφουμε τα βασικά στάδια λειτουργείας του Banks ως εξής:

- Για κάθε λεξη-κλειδί Ti βρίσκει το σύνολο των kw-κόμβων Si και θεωρεί το σύνολο όλων των kw-κόμβων το S=U(Si).

Εκτελεί αντίστροφη αναζήτηση απαντήσεων κατά την οποία:

- Δημιουργεί |S| αντίγραφα του Dijkstra's Single Source Shortest Path αλγορίθμου.
- Κάθε αντίγραφό του Dijkstra διασχίζει τον γράφο σε αντίθετη από αυτή των ακμών, ξεκινώντας από τον αντίστοιχο kw-κόμβο *Ti*.
- Προσπαθεί να βρει κάποιον κοινό κόμβο από τον οποίο υπάρχουν μονοπάτια με την κανονική κατεύθυνση των ακμών σε τουλάχιστον έναν κόμβο από το κάθε σύνολο Si, μέσω μιας λίστα κόμβων u.Li για κάθε κόμβο.
- Κάθε απάντηση έχει για ρίζα κάποιον κόμβο και ως φύλλα kw-κόμβους Οι απαντήσεις που θα βρεθούν ταξινομούνται σε αύξουσα σειρά ανάλογα με τη βαθμολογία τους.

Σε περίπτωση παρόμοιων απαντήσεων (με την ίδια δομή αλλά διαφορετικού κόμβορίζα), επιλέγεται αυτός με την υψηλότερη σχετικότητα.

Ψευδοκώδικας Banks

Σύνολα Kw-κόμβων $S = \cup S_i$ (S1,S2,…,Sn)

input: keywords $t_i, i = (1, \ldots, m)$ (ερώτημα)

output: Joining Networks of Tuples (JNTs) with size up to s for queries QU{wi}

1 Begin
2 For each kw-node $n \in S$ do
3 Create a Dijkstra SSSP iterator
4 Put the iterator to iteratorHeap
5 End for
6 While iteratorHeap not empty And more results required
7 u = Get next kw-node from iterator
8 if u is not visited by an iterator then
9 For i = 1 to n do
10 Create a node list u.Li : 0
11 End for
12 CrossProduct = $origin * \prod_{i \neq j} u.L_j$, $origin \in S_i$ (generate all connection trees containing node u)
13 For each $tuple \in CrossProduct$ do
14 Create ResultTree from tuple
15 If root of ResultTree has one child then
16 Continue
17 If outputHeap is full then
18 Output and remove top result from outputHeap
19 Insert ResultTree in to outputHeap ordered by its relevance score
20 End

5.5.2. Bidirectional

Ακολουθεί η περιγραφή και ο ψευδοκώδικας του συστήματος Bidirectional.

Για όσο οι iterators (incoming ή outgoing) δεν είναι άδειοι κάνει τα παρακάτω:

- Επιλέγει μεταξύ των iterators τον κόμβο με το καλύτερο activation για να συνεχίσει ανάλογα την αναζήτηση.
- Μεταδίδει το activation που έχει στους γειτονικούς του κόμβους και αντίστοιχα χαμηλώνει το δικό του σύμφωνα με τον παράγοντα διάδοσης μ.
- Αναβαθμίζει τα βάρη των μονοπατιών σύμφωνα με τις νέες διαδρομές.
- Όσους κόμβους εξερευνήθηκαν σε αντίστροφη κατεύθυνση τους εισάγει στον outgoing iterator, ώστε να ελέγξει στη συνέχεια αν μπορούν να γίνει αναζήτηση στην κανονική κατεύθυνση των ακμών από κάποιον υποψήφιο κόμβο-ρίζα.
- Κάθε κόμβο που βρίσκει να πληροί τις προϋποθέσεις για ρίζα απάντησης τον εισάγει στη λίστα απαντήσεων.
- Σταματά όταν εξαχθούν οι top-k απαντήσεις.

Algorithm Bidirectional

Kw-node sets {S1,S2,…,Sn} $S = \cup S_i$; (all kw-nodes matching query keywords)

$Qin \leftarrow S$, Qout:0; (ουρές συγκράτησης κόμβων για το backward/forward expand)

Xin:0, Xout:0; (σύνολο κόμβων που έγιναν expand για incoming/outgoing paths)

1. Begin
2. For each node u from S do
3. Pu : 0; (Pu= σύνολο κόμβων u που η ακμή (u,v) έχει ελεγχθεί)
4. $depth_u : 0;$ (απόσταση κόμβου u από kw-κόμβους)
5. End for
6. For each keyword t_i
7. For each node u ∈ S do

8. $sp_{ui}: \infty$; (κόμβος που θα ακολουθήσει ο u για το καλύτερο μονοπάτι προς τη λέξη-κλειδί t_i)
9. If node u ∈ S_i then
10. $dist_{u,i}: 0$; (μήκος καλύτερου γνωστού μονοπατιού από τον κόμβο u προς τον kw-κόμβο στο Si)
11. Else $dist_{u,i}: \infty$;
12. While Qin OR Qout are not empty do
13. If Qin has node with highest activation then
14. Pop best u from Qin and insert in Xin;
15. For i=1 to N do
16. If $dist_{u,i} \neq \infty$ then
17. Construct result tree rooted at u;
18. Add it to result heap;
19. If $depth_u < d_{max}$ then
20. $\forall u \in incoming\ [v]$ EXPLORE(u,v)
21. If $u\ not \in X_{in}$ then
22. insert it to Qin with $depth_u += 1$;
23. If $u\ not \in X_{out}$ then
24. insert it to Qout;
25. End if;
26. If Qout has node with highest activation then
27. Pop best u from Qout and insert in Xout;
28. For i=1 to N do
29. If $dist_{u,i} \neq \infty$ then
30. Construct result tree rooted at u;
31. Add it to result heap;
32. If $depth_u < d_{max}$ then

33. $\forall v \in outgoing\ [u]$ EXPLORE(u,v)
34. If $u\ not \in X_{out}$ then
35. insert it to Qout with $depth_u += 1;$
36. End if
37. End if
38. End if
39. End for
40. end

EXPLORE(u,v)

1. for each keyword i do
2. if u has better path to t_i via v then
3. $sp_{ui}: v$; update $dist_{u,i}$ with this new dist;
4. Update priority of u if presented in Q_{in};
5. Propagate change in cost $dist_{u,i}$ to all its reached ancestors in best first manner;
6. If $dist_{u,i} \neq \infty$ then
7. Construct result tree rooted at u;
8. Add it to result heap;
9. If v spreads more activation to u from I then
10. Update $a_{u,i}$ with the new activation;
11. Update priority of u if presented in Q_{in};
12. Propagate change in cost $a_{u,i}$ to all its reached ancestors in best first manner;
13. end

5.5.3. Discover

Ακολουθεί ο ψευδοκώδικας του Discover και η ανάλυση των βασικών βημάτων του.

Algorithm Discover

input: keywords $kw_i, i = \{1, \ldots, m\}$ (Query)

output: Joining Networks of Tuples (JNTs) with size up to s for queries QU{wi}

1 begin

2 Queue Qr: empty queue (queue of tuple sets $R_i^{k_1}, \ldots, R_i^{k_m}$)

3 **for** each keyword kw_i **do** (create the basic tuple sets)

4 search the Index for keywords

5 add to queue Qr every relation $R_i^{k_j}$ includes kw_j

6 end for

7 call **Algorithm 2** Create_GTS (create graph of Tuple Sets according to Qr)

8 call Algorithm 3: Candidate Networks Generation (create Candidate Networks)

9 Plan Generator (order answers by joining number)

9.1 Measure the joins from each JNT and store them to a list in increase order.

9.2 Convert answers to SQL statements

10 Execute plan (send answers to DB)

11 send Answers to user

12 end

Algorithm 2: Create_GTS (graph of Tuple Sets according to Qr)

input: schema graph, Qr

output: GTS (compute graph of Tuple Sets according to Qr)

1: begin

2: **for** each tuple set $R_i^{k_j}$ in Qr **do**

3: create a node R_i^k in GTS

4: end for

5: **for** each edge $R_i \rightarrow R_j$ in schema graph **do**

6: create an edge $R_i^k \rightarrow R_i^k$ in GTS

7: end for

8: return GTS

9: end

Algorithm 3: Candidate Networks Generation

input: graph of Tuple Sets (GTS), T (max size of JNTs), query keywords k1,…,km

output: Candidate Networks of tuples with size up to T

1 begin

2 Queue Q : empty queue (queue of JNTs)

3 pick a keyword $k_t \in \{k_1, \dots, k_m\}$;

4 **for** each tuple set R_i^k where $i = 0, \dots, n$ and $k_t \in K$ **do**

5 add Tuple Sets R_i^K to Q;

6 While Q is not empty do

7 $C \leftarrow head\ from\ Q$;

8 If C satisfies the pruning condition then

9 Indonre C; (current tuple set not acceptable)

10 Else if C satisfies the acceptance condition then

11 output C as MTJNT with max size T; (current tuple set is an answer)

12 Else for each tuple set R_i^K adjaced in GTS (ignore directions) to a node of C

13 Expand the C according to expansion rule

14 If R_i^K is adjacent to R_j^M in $C = R_j^M[\dots]$ then

15 $C = R_i^K\left[R_j^M[\dots]\right]$; (expand C with adjacent nodes)

16 Put C in Q; (put new C in Q)

17 Else ignore C;

18 End

Pruning Condition (συνθήκη απόρριψης): Μία υποψήφια απάντηση (candidate network) δεν περιέχει ένα υποδένδρο της μορφής $R^K — S^l — R^M$, με R, S να είναι πεδία και στο schema-graph να υπάρχει ακμή $R \rightarrow S$.

Περιγράφουμε περιληπτικά τα κύρια στάδια λειτουργίας του ως εξής:

- Οι λέξεις κλειδιά που εισάγει ο χρήστης k1,…,km αναζητούνται στο ευρετήριο της βάσης δεδομένων.
- Παράγεται ένα σύνολο από εγγραφές (tuple sets) $R_i^{k_1}, \ldots, R_i^{k_m}$ (για i=1,…,n & j=1,…,m) για κάθε πεδίο R_i όπου κάθε εγγραφή $R_i^{k_j}$ περιέχει τη λέξη-κλειδί k_j και προστίθεται σε μια λίστα Qr.
- Δημιουργείται ο γράφος των tuple sets (GTS) σύμφωνα με τη λίστα Qr διαβάζοντας το σχήμα της ΒΔ.
- Με τον γράφο GTS και τις λέξεις-κλειδιά δημιουργεί τις υποψήφιες απαντήσεις: Επιλέγει κάθε φορά τυχαία κάποια λέξη-κλειδί του ερωτήματος και προσθέτει το αντίστοιχο $R_i^{k_j}$ σε μια ουρά Q ως πιθανή απάντηση.
- Στο πρώτο σύνολο εγγραφών της ουράς Q κάνει τους εξής ελέγχους:
  - Εάν πληροί τις προϋποθέσεις για να αποτελέσει υποψήφια απάντηση ή να απορριφθεί.
  - Εάν περιέχει όλες τις λέξεις-κλειδιά ώστε να το εξάγει ως τελική απάντηση.
  - Διαφορετικά ελέγχει στο GTS για να το ενώσει με γειτονικά tuple sets και να σχηματίσει μια τελική απάντηση JNT ή να το απορρίψει.
- Οι απαντήσεις ταξινομούνται σύμφωνα με το πλήθος των ενώσεών τους αυξητικά και μετατρέπονται σε μορφή SQL ερωτημάτων.
- Στη συνέχεια αποστέλλονται στη ΒΔ για να προκύψουν τα τελικά αποτελέσματα.

5.5.4. Blinks

Περιγράφουμε περιληπτικά τα κύρια στάδια λειτουργίας του Blinks ως εξής:

- Για να υποστηρίξει την αντίστροφη αναζήτηση μεταξύ πολλών blocks χρησιμοποιεί μια ουρά Q_i από cursors για κάθε λέξη-κλειδί w_i.
- Για να βοηθήσει την διαδικασία της αναζήτησης προ-υπολογίζει στην αρχή τη συντομότερη απόσταση από τον κάθε κόμβο προς όλους τους kw-κόμβους και αποθηκεύει το αποτέλεσμα σε μια λίστα keyword node list (L_{KN}).
- Για κάθε keyword w_i βρίσκονται τα blocks που το περιέχουν και τοποθετούνται στην ουρά Q_i.
- Κάθε φορά που η αναζήτηση συναντά κάποιον κόμβο in-portal u σε κάποιο block, συνεχίζει την αντίστροφη αναζήτηση σε όλα τα blocks που έχουν τον u ως out-portal. Για κάθε τέτοιο block b χρησιμοποιεί έναν νέο cursor με αρχική απόσταση ίση με το συντομότερο μονοπάτι από τον κόμβο u προς τη λέξη-κλειδί w_i. Αναγνωρίζει αυτά τα blocks μέσω μιας λίστας portal block list L_{PB}.
- Μέσω της μεθόδου pickKeyword επιλέγει ο κόμβος με το ελάχιστο πλήθος των κόμβων που ελέγχθηκαν με τη βοήθεια μιας ουράς Q_i.
- Εάν ένας κόμβος u επισκέπτεται για πρώτη φορά αντίστροφα από κάποιους kw-κόμβους που περιέχουν την ίδια λέξη-κλειδί αλλά ανήκουν σε διαφορετικό block, τότε θα εξεταστεί ο κόμβος u μόνο την πρώτη φορά. Οι πληροφορίες συγκρατιούνται στη δομή bitmap *crossed*, για να αποφεύγεται η ίδια αναζήτηση στις μελλοντικές επισκέψεις του κόμβου u.
- Ελέγχει εάν η απόσταση μεταξύ δυο κόμβων u, w_i είναι η μικρότερη μέσα από το ίδιο το block ή μέσω διαδρομής από άλλα block.
- Για την αναζήτηση με την κανονική κατεύθυνση των ακμών ελέγχει εάν η απόσταση μεταξύ u και w_i είναι η μικρότερη εντός του block ή εάν υπάρχει άλλη συντομότερη μέσω άλλων blocks.
- Βαθμολογεί κάθε απάντηση και την προσθέσει στη λίστα απαντήσεων.
- Ελέγχει εάν πληρούνται οι συνθήκες τερματισμού της αναζήτησης ώστε να εξάγει τις top-k ή να συνεχίσει την αναζήτηση.

Algorithm Blinks

input: keywords w_i, $i = \{1, \ldots, m\}$

output: Joining Networks of Tuples (JNTs) with size up to T_{prune}

R:0 (potential answers)

A : 0 (completed answers)

T_{prune} limit threshold

Q_i queue of cursors

Crossed(i,u) (bitmap to indicate if backward expansion of w_i is crossed portal node u)

1 Begin
2 For each $i \in [1 \text{ to } m]$ do
3 $Q_i \leftarrow new\ Queue()$;
4 For each block b contains w_i do
5 add new cursor $L_{KN}(b, w_i)$ to Q_i
6 End for
7 End for
8 While $\exists j \in [1, \ldots, m]: Q_j \neq 0$ do
9 i : pickKeyword$(Q_i, \ldots, Q_m)$;
10 c : $Q_i.pop()$;
11 (u, d): c. next;
12 visitNode(i,u,d);
13 If $\neg crossed(i,u)\ and\ L_{PB}(u) \neq 0$ then
14 For each $b \in L_{PB}$ do (cross portal)
15 add new cursor $L_{PN}(b,u)$ to Q_i
16 End for
17 End if

18 If distance of c not ∞ then

19 Add c to Q_i;

20 If stop_condition if true then

21 Exits and output the top-k answers;

22 End while

23 Output top-k answers

24 End

5.5.5. Disvover με προτιμήσεις χρηστών

Ακολουθεί ο ψευδοκώδικας του συστήματος Discover με προτιμήσεις χρηστών.

Περιγράφουμε τα βασικά στάδια της λειτουργίας του συστήματος Discover ως εξής:

- Οι λέξεις-κλειδιά που εισάγει ο χρήστης $\mathrm{k}_1, \ldots, \mathrm{k}_m$ αναζητούνται στο ευρετήριο της βάσης δεδομένων.
- Παράγεται ένα σύνολο από εγγραφές (tuple sets) $\mathrm{R_i}^{\mathrm{k_1}}, \ldots, \mathrm{R_i}^{\mathrm{k_m}}$ για κάθε σχέση πρωτεύοντος-ξένου κλειδιού R_i (για i=1,…,n). Κάθε εγγραφή $R_i^{k_j}$ θα περιέχει τη λέξη-κλειδί k_j (για *j=1,..,m* λέξεις-κλειδιά) και προστίθεται σε μια λίστα εγγραφών Qr.
- Δημιουργείται ο γράφος προτιμήσεων με τη μέθοδο που είδαμε στο Σχήμα 3.6.
- Οι υποψήφιες απαντήσεις αναζητώνται στον γράφο εγγραφών *Gts* επιλέγοντας κάθε φορά όχι τυχαία κάποια λέξη-κλειδί, αλλά σύμφωνα με τα επίπεδα προτεραιότητας του γράφου προτιμήσεων *Gp*.
- Πριν προστεθεί μια απάντηση στην λίστα τελικών απαντήσεων, ελέγχεται για τη μοναδικότητά των εγγραφών της με όλες τις υπόλοιπες.
- Στην πρώτη υποψήφια απάντηση της ουράς Q ελέγχει:
  - Εάν περιέχει όλες τις λέξεις-κλειδιά ώστε να την εξάγει ως τελική απάντηση αυτούσια.

- Διαφορετικά ελέγχει τον γράφο εγγραφών Gts για να την ενώσει με γειτονικά δίκτυα εγγραφών ώστε να σχηματίσει μια τελική απάντηση.
- Εάν δεν μπορεί να αποτελέσει τελική απάντηση τότε την απορρίπτει.

- Οι τελικές απαντήσεις ταξινομούνται και μετατρέπονται σε μορφή SQL ερωτημάτων τα οποία αποστέλλονται στη βάση δεδομένων.
- Τα αποτελέσματα που θα ανακτηθούν από τη βάση δεδομένων μεταφέρονται στον χρήστη.

Discover_Pref Algorithm pseudo-code

Algorithm 1 Discover_Pref

(Discover with Preference ranking)

input: keywords $kw_i, i = \{1, \ldots, m\}$ (Query)

preferences.txt

(user preferences)

output: Joining Networks of Tuples (JNTs) with size up to s for queries QU{wi}

1: begin

2: Queue Qr: empty queue

(queue of tuple sets)

3: **for** each kw_i **do**

(create the basic tuple sets)

4: search the Master Index

5: add to queue Qr every relation $R_i^{k_j}$ includes kw_j

6: end for

7: **return** Qr (return of tuple sets $R_i^{k_1}, \ldots, R_i^{k_m}$)

8: call **Algorithm 2** Create_GTS (graph of Tuple Sets according to Qr)

9: call **Algorithm 3** WinnowOP (order kw_i in levels of preference hierarchy)

10: call **Algorithm 4** Compute JNT's (compute networks of JNTs from kw_i ordering)

11: call **Algorithm 5** PlanExecution (SQL statement production from JNTs network)

12: send Answers to user

13: end

Algorithm 2: Create_GTS (graph of Tuple Sets according to Qr)

input: schema graph, Qr

output: GTS (compute graph of Tuple Sets according to Qr)

1: begin

2: **for** each tuple set $R_i^{k_j}$ in Qr **do**

3: create a node R_i^k in GTS

4: end for

5: **for** each edge $R_i \rightarrow R_j$ in schema graph **do**

6: create an edge $R_i^k \rightarrow R_i^k$ in GTS

7: end for

8: return GTS

9: end

Algorithm 3: WinnowOP (order kw_i in levels from user preferences)

input: preferences.txt (user preferences)

output: winnow result list (order kw_i in levels of preference hierarchy)

1: begin

2: L=1 (integer), winnow_result: list (empty)

3: G_{PC} (empty directed graph of user choises)

Create the graph of preferences $G_{PC}(V_G, E_G)$

4: **for** each kw_i in preferences.txt **do** (kw_i exists in usre preferences)

5: create a node V_{GWi} to G_{PC}

4: **for** each Dominance $w_i \succ w_j$ **do** (kw_i dominates to kw_j)

5: create a directed edge $E_{GWi} \rightarrow E_{GWj}$ to G_{PC}

6: **return** G_{PC}

7: **while** G_{PC} is not empty **do**

8: **for** all $kw_i \in V_G$ with no incoming edges **do**

9: winpc(L)= $winpc \cup \{w_i\}$ (take the kw's ho dominates to others)

10: end for

11: winnow_result= winnow_result+winpc(L) (put those kw's in level L)

12: $V_G = V_G - winpc(L)$

13: **for** all adges $e \in (w_i, w_j)$, $w_i \in winpc(L)$ **do**

(delete outgoing edges from kw's added in winnow_result level (L))

$E_G = E_G - e$

14: end for

15: L++

16: end while

17: return winnow_result

18: end

Algorithm 4: Compute JNTs

(compute networks of JNTs by GPC ordering)

input: GTS, winnow_result list, T (T=max size of JNT's)

output: JNTs list

(ordered Joining Network of Tuples with max size T)

1: begin

2: Q=queue of JNTs

3: L=1, top=0, JNT_LIST=empty list

4: **while** winnow_result (L) is not empty AND top not equals toT **do**

5: pick next kw of level (L)

(take kw from level L and put in Q the JNT this exist)

6: **for** each Tuple Set R_i^k, where i=1,…n and $k_t \in K$ **do**

7: add JNT R_i^k to Q

8: **end for** L++, top++

9: end while

10: **while** Q is not empty **do**

11: get head C from Q

12: **if** C satisfies the pruning condition

13: ignore C

14: **else if** C satisfies the acceptable conditions

15: **for** all JNTs in Q **do** (check for Diversity in each JNT of Q with C)

16: compare the size of each JNT with size of C (chek distance of kws of every 2 JNTs)

17: **if** size of C equals to size of at least one JNT in Q **do**

18: stop the for loop

19: **else** put C to JNT_LIST

20: end for

21: **else for** each tuple set R_i^k adjacent in GTS (expand JNTs ignoring edge direction)

22: **if** R_i^k satisfies the expansion rule **do**

23: put expanded C in Q

24: end for

25: end while

26: **return** JNT_LIST

27: **end**

Algorithm 5 PlanExecution (SQL statement production from JNTs network)

input: JNTs list (ordered Joining Network of Tuples with max size T)

output: Answers list (list of answers retrieved from database)

1: begin

2: Answers=empty list

3: **while** Q is not empty **do**

4: convert JNT in to SQL statement

5: send SQL statement to Database

6: put the answer of DB to Answers list

6: end while

7: **return** Answers

8: **end**

Βλέπουμε τον πίνακα με τις εισόδους και τις εξόδους για τις μεθόδους που καλούνται σε όλα τα στάδια εκτέλεσης αλγορίθμου.

Algo No	Εισοδος αλγορίθμου	Εξοδος αλγορίθμου
1	Query Preferences	**Qr**: queue of tuple sets
2	**Qr**: queue of tuple sets (-1-)	**GTS**: graph of Tuple Sets
3	Preferences	**winnow_result**: List of Kw's with ordered by preference hierarchy
4	**GTS**: -2- Winnow_result -3-	**Q**=queue of JNTs ordered by preference hierarchy
5	**Q** -4-	**Answers**= Answers retrieved from DB after JNT converted to SQL statements

5.6. Γ.2 Προτιμήσεις χρηστών για το σύστημα αναζήτησης Discover

Στον Πίνακα Π.22 και Πίνακα Π.23 βλέπουμε τις προτιμήσεις χρηστών που δηλώθηκαν για την εκτέλεση των ερωτημάτων του συστήματος Discover με προτιμήσεις χρηστών.

Πίνακας Π.20 Προτιμήσεις χρηστών για τη βάση δεδομένων DBLP.

Μέγεθος ερωτημάτων 2
AAAI > ICIS
ICIS > STOC

OSDI > ICDE
ICDE > VLDB
MIT > Oxford
Singapore > ITT
Harvard > Toronto
Conference > Journal
Journal > Theses
Journal > Collections
Conference > Papers
Science > database
Algorithm > software
Software > dynamic
Dynamic > relational
Optimal > statistical
Informal > editorship
Mit > Stanford
Engineering > Computer science
Keyword > relational
implement > debug
runn > execute
Μέγεθος ερωτημάτων 3
ISSAC > VLDB
ICIS > STOC
OSDI > ICDE
ICIS > OSDI
Ioannina > Cambridge
Singapore > Toronto
Harvard > Oxford
Conference > Journal
Journal > Thesis

Collection > Journal
Paper > Journal
Distributed > optimal
Optimal > dynamic
Dynamic > implement
Engineering > Computer science
Department > area
Computer science > physics
Relational > retrieval
Rdms > sql
Rdms > retrieval
retrieval > optimization
Algorithm > software
dynamic > Software
machine > physical
Μέγεθος ερωτημάτων 4
VLDB > ACMMM
AAAI > VLDB
ICPF > ICIS
AAAI > ICIS
Singapore > Toronto
Harvard > Toronto
Toronto > Edinburgh
Ioannina > Washington
Article > publication
Conference > Journal
Journal > Thesis
Thesis > Collection
Conference > Paper
software > machine

hardware > software
learning > machine
optimal > minimal
implementing > debuging
runing > executing
index > inverted list
xml > html
xml > sql
sql > html
Optimal > dynamic
Dynamic > implement
Engineering > Computer science
learning > machine
hardware > software
optimal > minimal
runing > implementing
Μέγεθος ερωτημάτων 5
VLDB > ACMMM
AAAI > VLDB
ICPF > ICIS
AAAI > ICIS
ISSAC > ICPF
Ioannina > MIT
Singapore > Toronto
Toronto > Harvard
Harvard > Toronto
learning > machine
hardware > software
optimal > minimal
runing > implementing

software > optimization
optimization > hardware
performance > processing
network > query
processing > optimization
optimization > unsupervised
unsupervised > hardware
fault > debugging
hardware > software
learning > machine
optimal > minimal
Dynamic > relational
Optimal > statistical
Informal > editorship

Πίνακας Π.21 Προτιμήσεις χρηστών για τη βάση δεδομένων IMDB.

Μεγεθος ερωτημάτων 2
Angelina Jolie > Nicole Kidman
Comedy > Drama
SciFii > Drama
Hanks > Carrey
Hitchcock > Spielberg
Aronofski > Spielberg
Morgan Freeman > Brad Pitt
Anthony Hopkins > deNiro
deNiro > Brad Pitt
house > quest
Harrison Ford > Richard Gere
Tom Cruise > Brad Pitt
Brosnan > Daniel Craig

Sean Connery > Daniel Craig
Tom Cruise > Richard Gere
Stalone > Jean Claude VanDam
Μεγεθος ερωτημάτων 3
Jack Nicholson > Kevin Spacey
Jack Nicholson > Kevin Bacon
Hotel > dance
Leonard > helen
Kevin Spacey > Kevin Bacon
Adventure > Dramaa
Dramaa > Comedy
SciFii > Dramaa
Dramaa > Romance
Julia Roberts > Nicole Kidman
Harrison Ford > Tom Cruise
Tom Hanks > Caprio
Antonio Banderas > Caprio
Daniel Craig > Richard Gere
Adventure Mel Gibson > Keanu Reeves
Adventure Brad Pitt > Mel Gibson
Adventure Mel gibson > Will Smith
Spielberg Harrison Ford > Tom Cruise
Spielberg Tom Hanks > Caprio
Spielberg Tom Cruise > Caprio
Μεγεθος ερωτημάτων 4
Horror > SciFi
Comedy > Action
Comedy> SciFi
Angelina Jolie > Sandra Bullock
Sandra Bullock > Nicole Kidman

Sean Connery > Michael Douglas
Mel Gibson > Sean Connery
Action Arnold Schwarzenegger > Stalone
Action Steven Seagal > Mel Gibson
Action Sean Connery > Brosnan
Action Sean Connery > Arnold Schwarzenegger
Angelina Jolie > Julia Roberts
Alfred Hitchcock > Spielberg
Alfred Hitchcock > Tarantino
Julia Roberts > Nicole Kidman
Harrison Ford > Tom Cruise
Tom Hanks > Caprio
Μεγεθος ερωτημάτων 5
Comedy> Drama
Comedy> Romance
Drama > Horror
Drama > SciFi
American > China
Dream > kill
Jason Statham > Tom Cruise
Antony Hopkins > Lee Jones
Sean Connery > Donald Sutherland
Jason Statham > Bruce Willis
Spielberg Harrison Ford > Tom Cruise
Spielberg Tom Hanks > Donald Sutherland
Spielberg Antonio Banderas > Caprio
Spielberg Donald Sutherland > Richard Gere
Meryl Streep > Diane Keaton
Drama Johnny Depp > Jack Nicholson
Drama Nicole Kidman > Jodie Foster

Comedy Hoffman > Robert deNiro
Comedy Brad Pitt > Robert deNiro
Comedy Johnny Depp > Hoffman
Comedy Meryl Streep > Nicole Kidman
Niro > Matt Damon
Jack Nicholson > Anthony Hopkins
Jack Nicholson > Niro
Meryl Streep > Foster
Foster > Matt Damon

ΣΥΝΤΟΜΟ ΒΙΟΓΡΑΦΙΚΟ

Ο Λάμπρος Βατσιλίδης γεννήθηκε το 1976 στην Άρτα. Από το 1999 ασχολείται επαγγελματικά με την Πληροφορική στον τομέα της τεχνικής υποστήριξης υπολογιστικών συστημάτων και λογισμικού. Το 2015 απέκτησε την παιδαγωγική επάρκεια για τη διδασκαλία μαθημάτων Πληροφορικής από την Α.Σ.ΠΑΙ.Τ.Ε. και είναι πιστοποιημένος εκπαιδευτής ενηλίκων. Απέκτησε βασικό Πτυχίο από το Τμήμα Πληροφορικής του Ελληνικού Ανοικτού Πανεπιστημίου το 2013 και από το ίδιο έτος παρακολουθεί το μεταπτυχιακό πρόγραμμα του τμήματος Μηχανικών Η/Υ και Πληροφορικής του Πανεπιστημίου Ιωαννίνων με κατεύθυνση το Λογισμικό.