Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Διδακτική της Τεχνολογίας & Ψηφιακά Συστήματα»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάκτηση Κειμένων από Μεγάλες Συλλογές Εγγράφων με Χρήση Προηγμένων Τελεστών Επερώτησης**
Ονοματεπώνυμο Φοιτητή	**ΕΛΕΝΗ ΝΙΚΗΦΟΡΑΚΗ**
Αριθμός Μητρώου	**ΜΕ/0582**
Κατεύθυνση	**Δικτυοκεντρικά Συστήματα**
Επιβλέπων	**ΧΡΗΣΤΟΣ ΔΟΥΛΚΕΡΙΔΗΣ**

Ημερομηνία Παράδοσης **02 2012**

Τριμελής Εξεταστική Επιτροπή

ΓΕΩΡΓΙΟΣ ΒΑΣΙΛΑΚΟΠΟΥΛΟΣ ΚΑΘΗΓΗΤΗΣ	ΑΝΔΡΙΑΝΑ ΠΡΕΝΤΖΑ ΕΠΙΚΟΥΡΗ ΚΑΘΗΓΗΤΡΙΑ	ΧΡΗΣΤΟΣ ΔΟΥΛΚΕΡΙΔΗΣ ΛΕΚΤΟΡΑΣ

ΠΕΡΙΛΗΨΗ

Η βελτίωση των μεθόδων ή ανακάλυψη νέων για την ανάκτηση κειμένων από μεγάλες συλλογές εγγράφων αποτελεί πρόκληση. Η ραγδαία ανάπτυξη των τεχνολογιών και των μεθόδων αποθήκευσης μεγάλου όγκου αρχείων που περιέχουν πληροφορία με τη μορφή κειμένου, οδηγεί ταυτόχρονα στην αύξηση των αναγκών για την εύρεση τεχνικών που θα βελτιώσουν τους χρόνους απόκρισης και θα είναι περισσότερο αποδοτικές. Σε αυτήν την μεταπτυχιακή διατριβή χρησιμοποιούνται οι υπάρχουσες τεχνικές που έχουν αναπτυχθεί στο επιστημονικό πεδίο της ανάκτησης πληροφορίας και επιχειρείται η ενσωμάτωση μίας νέας, της τεχνικής εύρεσης κορυφογραμμής (Skyline) για τον εντοπισμό των εγγράφων που το περιεχόμενό τους παρουσιάζει τη μεγαλύτερη ομοιότητα με τις λέξεις κλειδιά που έχουν διατυπωθεί στην επερώτηση. Για την πειραματική διαδικασία χρησιμοποιούνται συλλογές επιστημονικών άρθρων από συνέδρια, τα οποία είναι σε μορφή PDF. Για την επεξεργασία τους και την εξόρυξη του κειμένου από την περίληψη του καθενός ώστε να εξαχθούν οι όροι και να δημιουργηθεί το ευρετήριο πάνω στο οποίο θα ενεργούνται οι αναζητήσεις, χρησιμοποιούνται οι τεχνικές της βιβλιοθήκης Lucene. Αυτή η βιβλιοθήκη αποτελεί ένα πολύ διαδεδομένο, open source εργαλείο αναζήτησης. Λόγω της ευρείας αποδοχής που παρουσιάζει, τα αποτελέσματά της χρησιμοποιούνται ως μέτρο σύγκρισης για τη νέα τεχνική ανάκτησης που ενσωματώνεται σε αυτήν την πτυχιακή. Η εφαρμογή της μεθόδου εύρεσης κορυφογραμμής έγκειται στη χρησιμοποίηση πολυδιάστατων δεδομένων, τέτοιου τύπου διαχειρίζεται και η Lucene, αφού αναπαριστά τα έγγραφα ως διανύσματα. Επίσης, η τεχνική βαθμολόγησης της Lucene συνδέεται με τη σχέση υπεροχής μεταξύ δύο σημείων. Η τεχνική Skyline βασίζεται στις σχέσεις κυριαρχίας μεταξύ σημείων για τον καθορισμό της κορυφογραμμής – ουρανογραμμής.

ΠΕΡΙΕΧΟΜΕΝΑ

1. Εισαγωγή

Η πρόοδος της επιστήμης των υπολογιστών και της πληροφορικής και η αλματώδης ανάπτυξη του διαδικτύου έδωσαν τη δυνατότητα αποθήκευσης πολύ μεγάλου όγκου δεδομένων και πληροφοριών. Ταυτόχρονα όμως η αναζήτησή τους γίνεται δυσκολότερη και τα αποτελέσματα που επιστρέφονται αμφισβητήσιμα. Η διόγκωση των αναγκών για γρήγορη και αποδοτική ανάκτηση πληροφοριών καθιστά αναγκαία την εύρεση τεχνικών, ικανών να εντοπίζουν και να σταχυολογούν τις πληροφορίες ολοένα και πιο αποδοτικά. Το επιστημονικό πεδίο που ασχολείται και αναπτύσσει τεχνικές για την αντιμετώπιση αυτών των προβλημάτων είναι η επιστήμη της ανάκτησης πληροφορίας. Αυτό το πεδίο έχει προσελκύσει πολλούς ερευνητές και κορυφαίες εταιρείες πληροφορικής στην προσπάθεια να βελτιωθούν οι υπάρχουσες τεχνικές κα να ανακαλυφθούν νέες περισσότερο αποδοτικές.

Η σημαντικότητα του πεδίου προβάλλεται από τη χρησιμοποίηση των τεχνικών αυτών σε πολλές λειτουργίες της καθημερινής μας ζωής αλλά και σε γειτονικές επιστημονικές περιοχές, όπως οι βάσεις δεδομένων, η εξόρυξη δεδομένων, η τεχνητή νοημοσύνη κ.ά.. Οι μέθοδοι αυτοί εφαρμόζονται για την πρόσβαση σε δεδομένα διαθέσιμα μέσω του παγκόσμιου ιστού, σε πολυμεσικά δεδομένα (εικόνα, ήχος, βίντεο), σε βιβλιοθήκες κτλ. Η πλέον διαδεδομένη εφαρμογή, η οποία αποτελεί ένα σύστημα ανάκτησης πληροφορίας είναι η μηχανή αναζήτησης Google.

Η παρούσα μεταπτυχιακή διατριβή είναι προσανατολισμένη στην ενσωμάτωση νέων μεθόδων για την ανάκτηση κειμένων μέσα από μεγάλες συλλογές εγγράφων. Το υπάρχον σύστημα ανάκτησης πληροφορίας θα επεκταθεί με στόχο να δημιουργηθεί ένα εργαλείο που θα έχει τη δυνατότητα να εντοπίσει τα έγγραφα που περιέχουν τις λέξεις κλειδιά της επερώτησης που έχει τεθεί χρησιμοποιώντας και νέες μεθόδους ανάκτησης. Η νέα μέθοδος που θα υλοποιηθεί είναι η ενσωμάτωση ενός αλγορίθμου Skyline (κορυφογραμμής), τα αποτελέσματα τα οποία θα εξαχθούν θα είναι ιδιαιτέρως χρήσιμα διότι θα βοηθήσουν στη μέτρηση της απόδοσής της. Για την αποτίμηση της νέας μεθόδου θα χρησιμοποιηθούν τα αποτελέσματα που παράγονται από την ευρείας αποδοχής βιβλιοθήκη Lucene.

Η Lucene αποτελεί ίσως την περισσότερο διαδεδομένη βιβλιοθήκη αναζήτησης ανοιχτού κώδικα, είναι πολύ απλή στη χρήση της και είναι επεκτάσιμη. Η αναγνωσιμότητά της οφείλεται στο ότι είναι μία πλήρης, υψηλής απόδοσης βιβλιοθήκη ανάκτησης πληροφοριών υλοποιημένη στη γλώσσα Java. Στην παρούσα εργασία, εκτός από τη χρήση των αποτελεσμάτων της ως μέτρο σύγκρισης, χρησιμοποιείται για την διαχείριση και επεξεργασία των εγγράφων προς ευρετηριοποίηση, την ευρετηριοποίηση του κειμένου τους και την εκτέλεση των επερωτήσεων με στόχο των εντοπισμό αυτών που ταιριάζουν περισσότερο με τις επερωτήσεις.

Όσον αφορά την ιδέα εφαρμογής της τεχνικής skyline σε κείμενα, προκύπτει από το γεγονός ότι αυτή η τεχνική εφαρμόζεται σε πολυδιάστατα δεδομένα, με τη χρήση της Lucene η πληροφορία σχετικά με τα έγγραφα αναπαριστάται με τη μορφή διανυσμάτων. Συνεπώς οι επερωτήσεις και η αναζήτηση και με τις δύο μεθόδους εφαρμόζονται σε πολυδιάστατα δεδομένα.

1.1. Αντικείμενο της διπλωματικής

Αντικείμενο της παρούσας μεταπτυχιακής διατριβής αποτελεί η επέκταση και ανάπτυξη υπάρχουσας εφαρμογής με την οποία θα γίνεται ανάκτηση εγγράφων από μεγάλες συλλογές

εγγράφων ενσωματώνοντας την τεχνική Skyline. Θα επεκταθεί η εφαρμογή η οποία υλοποιήθηκε στα πλαίσια της εργασίας των Βανού, Κακοσίμου και Μαθιουδάκη με τίτλο «Ευρετηριοποίηση και αναζήτηση με λέξεις - κλειδιά σε συλλογές κειμένων». Πραγματοποιούνται τροποποιήσεις και προσθήκες στο ήδη υλοποιημένο σύστημα ανάκτησης πληροφορίας και ενσωματώνεται μια νέα μέθοδος ανάκτησης πληροφορίας από τις ευρετηριασμένες συλλογές εγγράφων με τη χρήση Skyline επερωτήσεων. Η νέα μεθοδολογία στηρίζεται σε τεχνικές εξόρυξης δεδομένων από το πεδίο της ανάκτησης πληροφορίας, για την παραγωγή και την ταξινόμηση των αποτελεσμάτων που λαμβάνονται από την εκτέλεση αναζητήσεων στις συλλογές εγγράφων.
Στο πλαίσιο της ανάπτυξης της εφαρμογής μελετάται το πεδίο της ανάκτησης πληροφοριών τα μοντέλα και οι μεθόδοι που χρησιμοποιούνται. Επιπλέον διερευνάται ο τρόπος λειτουργίας της βιβλιοθήκης ευρετηριασμού και αναζήτησης Apache Lucene και μελετώνται τα ερωτήματα κορυφογραμμής (Skyline) τα οποία είναι μία τεχνική διαχείρισης και επεξεργασίας πολυδιάστατων δεδομένων. Σημαντικό ρόλο στην εργασία διαδραματίζει η πειραματική αποτίμηση της νέας μεθοδολογίας. Προς αυτήν την κατεύθυνση εκτελούνται μια σειρά από πειράματα, ώστε από τα αποτελέσματα που θα προκύψουν να είναι εφικτή η σύγκριση της νέας μεθόδου με τις υπάρχουσες τεχνικές.

1.2. Γενικά για την εφαρμογή

Η desktop εφαρμογή που αναπτύχθηκε στα πλαίσια της παρούσας διατριβής αποτελεί ένα σύστημα ανάκτησης πληροφορίας. Δέχεται λέξεις-κλειδιά από το χρήστη και προχωρά στην αναζήτηση αυτών σε συλλογές επιστημονικών άρθρων που έχουν παρουσιαστεί σε συνέδρια, τα έγγραφα είναι σε μορφή PDF. Το σύστημα διαχειρίζεται τα έγγραφα .pdf, εξάγει για το καθένα από αυτά σημασιολογικές και συντακτικές πληροφορίες και τις χρησιμοποιεί κατάλληλα ώστε να αποφασίσει αν το έγγραφο είναι σχετικό ή όχι με το ερώτημα του χρήστη. Επιπλέον, μπορεί να δημιουργεί αυτοματοποιημένα ερωτήματα με λέξεις-κλειδία που έχει επιλέξει από την προεπεξεργασία των εγγράφων της συλλογής και να υποβάλει πειραματικά ερωτήματα.

Όπως αναφέρθηκε παραπάνω, η παρούσα εργασία είναι προσανατολισμένη, πέρα των υπολοίπων, και στην παρουσίαση μιας νέας μεθόδου ανάκτησης πληροφορίας, η νέα μέθοδος προκύπτει από την ενσωμάτωση ενός αλγορίθμου κορυφογραμμής (Skyline) στο σύστημα με στόχο την παραγωγή αποτελεσμάτων ώστε να είναι δυνατή η σύγκριση της απόδοσής της με τις υπάρχουσες τεχνικές που εφαρμόζονται στο πεδίο της ανάκτησης πληροφοριών.

Η ανάπτυξη του λογισμικού βασίζεται στην χρήση της αντικειμενοστραφούς γλώσσας προγραμματισμού Java και αναπτύχθηκε στην πλατφόρμα NetBeans σε περιβάλλον Windows. Στο λογισμικό χρησιμοποιείται η βιβλιοθήκη ελεύθερου κώδικα Apache Lucene, πρόκειται για μία βιβλιοθήκη ευρετηρίασης και αναζήτησης, που έχει αναπτυχθεί σε Java. Για την εξυπηρέτηση των σκοπών της Lucene έχουν αναπτυχθεί ή προσαρμοστεί πολλές βοηθητικές βιβλιοθήκες που εξυπηρετούν συγκεκριμένους σκοπούς. Συνεπώς στην εργασία χρησιμοποιούνται και άλλες βιβλιοθήκες, όπως για παράδειγμα η Pdfbox, η οποία χρησιμεύει στη διαχείριση των εγγράφων PDF. Αναλυτικότερη περιγραφή των εργαλείων και των βιβλιοθηκών που χρησιμοποιούνται ακολουθεί στα επόμενα κεφάλαια.

2. Θεωρητικό Υπόβαθρο

Το θεωρητικό υπόβαθρο πάνω στο οποίο βασίζεται η ανάπτυξη του λογισμικού είναι η επιστήμη της ανάκτησης πληροφορίας, τα μοντέλα της και οι τεχνικές που εφαρμόζονται σε καθένα από αυτά. Επιπλέον, για τη δημιουργία της νέα μεθόδου ανάκτησης δανειζόμαστε τη μέθοδο εύρεσης κορυφογραμμής (Skyline). Αυτή η μέθοδος χρησιμοποιείται και σε άλλες επιστημονικές περιοχές για την επίλυση των προβλημάτων που προκύπτουν από τη διαχείριση πολυδιάστατων δεδομένων.

Το πρόβλημα της κορυφογραμμής συνδέεται με την έννοια των πολυδιάστατων δεδομένων, στην παρούσα εργασία αυτά προκύπτουν από τη διαχείριση των εγγράφων μέσω των τεχνικών της ανάκτησης πληροφοριών. Η αποτελεσματική διαχείριση και παραγωγή αποτελεσμάτων απαιτεί τη χρήση ευρετηρίων, η αναπαράσταση των εγγράφων και των όρων που αποθηκεύονται σε αυτά δημιουργούν πολυδιάστατα δεδομένα. Τα ερωτήματα κορυφογραμμής (skyline queries) είναι ένας τύπος ερωτημάτων που μπορεί να εφαρμοστεί σε πολυδιάστατα δεδομένα, αυτή η ιδιότητά τους, σε συνδυασμό με τον στόχο της ανάκτησης πληροφορίας, που είναι η εύρεση δεδομένων τα οποία είναι παρόμοια ως προς κάποια άλλα δοθέντα, τα καθιστούν ως ένα ιδιαίτερα χρήσιμο εργαλείο.

Ένα ερώτημα κορυφογραμμής (skyline) λαμβάνει ως είσοδο ένα σύνολο πολυδιάστατων δεδομένων (π.χ. στις 2 διαστάσεις) και επιστρέφει το σύνολο των αντικειμένων (σημείων) που είναι τα καλύτερα δυνατά. Για παράδειγμα, έστω ένα σύνολο από φορητούς υπολογιστές για τους οποίους καταγράφουμε την τιμή και την ταχύτητα του επεξεργαστή. Οι καλύτεροι υπολογιστές είναι αυτοί που έχουν χαμηλή τιμή (μικρές τιμές της διάστασης «τιμή») και μεγάλη ταχύτητα επεξεργαστή (μεγάλες τιμές της διάστασης «ταχύτητα»). Στην περίπτωση αυτή, οι καλύτεροι υπολογιστές προσδιορίζονται από μία διαδικασία βελτιστοποίησης με πολλαπλά κριτήρια. Ομοίως τα μοντέλα ανάκτησης πληροφορίας επιστρέφουν για παράδειγμα ένα σύνολο από σχετικά έγγραφα, δοθέντος ενός ερωτήματος με λέξεις κλειδιά.

Τα μοντέλα ανάκτησης πληροφορίας και οι τεχνικές επεξεργασίας των δεδομένων που χρησιμοποιούνται στη συγκεκριμένη εργασία, περιέχονται ή αναπτύσσονται σε γλώσσα Java, στον πυρήνα της βιβλιοθήκης Lucene. Η Lucene αποτελεί μια υψηλής απόδοσης, επεκτάσιμη βιβλιοθήκη ανάκτησης πληροφοριών, υλοποιημένη στη γλώσσα Java, περιεκτικά, περιγράφεται ως βιβλιοθήκη ευρετηρίασης και αναζήτησης. Δηλαδή, αυτή η βιβλιοθήκη ενσωματώνει όλες τις διαδικασίες οι οποίες είναι απαραίτητες και επιτελούνται για την ανάκτηση πληροφορίας.

2.1. Γενικά για την Ανάκτηση Πληροφορίας (Information Retrieval)

Η Ανάκτηση Πληροφορίας (Information Retrieval – IR) είναι η επιστημονική περιοχή που μελετά τα προβλήματα που σχετίζονται με την αναπαράσταση, την οργάνωση και την επεξεργασία στοιχείων πληροφορίας, με στόχο την αποτελεσματική και αποδοτική πρόσβαση των χρηστών σε αυτά. Αρχικά η μελέτη επικεντρώθηκε στα έγγραφα κειμένων, όμως στη συνέχεια οι ανάγκες των σύγχρονων εφαρμογών οδήγησαν στην επέκτασή της και σε άλλους τύπους δεδομένων. Η διάδοση του διαδικτύου έδωσε μεγάλη ώθηση στον τομέα της αναζήτησης σε μη δομημένες μορφές πληροφορίας με συνέπεια να καταστεί επιτακτική η ανάγκη για αποδοτική αναζήτηση. Επίσης η διάδοση των πολυμέσων και η ραγδαία αύξηση της χρήση τους στο διαδίκτυο ανέδειξε την επιπρόσθετη ανάγκη για αναζήτηση πληροφορίας πέρα από κείμενα και σε άλλες μορφές αρχείων. Έτσι σήμερα, οι μέθοδοι ανάκτησης πληροφοριών

χρησιμοποιούνται για την πρόσβαση σε πολυμεσικά δεδομένα (εικόνα, ήχος, βίντεο), σε δεδομένα διαθέσιμα μέσω του παγκόσμιου ιστού καθώς και σε βάσεις δεδομένων. Στα περισσότερα πανεπιστήμια και δημόσιες βιβλιοθήκες χρησιμοποιούνται συστήματα IR ώστε να παρέχεται η πρόσβαση σε βιβλία, σε περιοδικά και σε άλλα έγγραφα. Οι πλέον διαδεδομένες εφαρμογές IR είναι οι μηχανές αναζήτησης του παγκόσμιου ιστού.

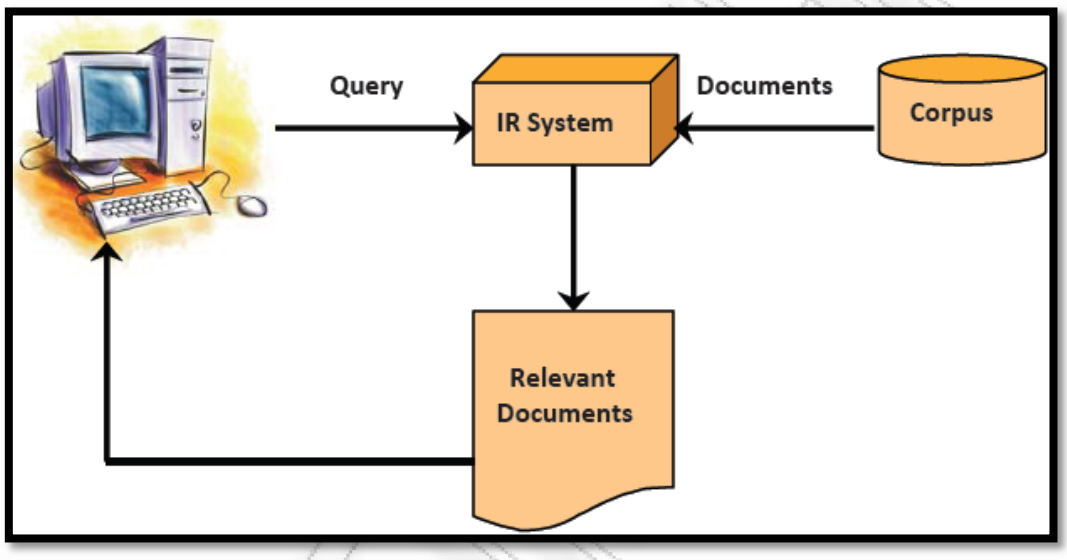
Η ιδέα της χρησιμοποίησης των υπολογιστών για την αναζήτηση σχετικών πληροφοριών δημοσιεύτηκε στο άρθρο "As We May Think" του Vannevar Bush το 1945. Οι πρώτες εφαρμογές των συστημάτων ανάκτησης πληροφορίας εισήχθησαν στις δεκαετίες του '50 και του '60 για τη διερεύνηση κειμένων σε μικρές συλλογές περιλήψεων επιστημονικών άρθρων καθώς και νομικών και επιχειρηματικών κειμένων. Κατά τη δεκαετία του '80 αναπτύχθηκαν συστήματα μεγάλων συλλογών κειμένων που χρησιμοποιήθηκαν από εταιρείες (MEDLINE, Dialog κ.ά.), στην ακόλουθη δεκαετία αναπτύχθηκε η αναζήτηση αρχείων μέσω FTP και η αναζήτηση ιστοσελίδων στο διαδίκτυο (Yahoo, Altavista κ.ά.). Επιπλέον την δεκαετία του '90 το αμερικανικό υπουργείο άμυνας μαζί με το εθνικό ίδρυμα προτύπων και τεχνολογίας (NIST), χρηματοδότησαν τη διάσκεψη ανάκτησης κειμένων Text Retrieval Conference (TREC) ως τμήμα του προγράμματος κειμένων TIPSTER. Ο στόχος ήταν η παροχή της υποδομής που ήταν απαραίτητη ώστε να αξιολογηθούν οι μεθοδολογίες ανάκτησης κειμένων σε πολύ μεγάλες συλλογές κειμένων. Αυτό διαδραμάτισε καταλυτικό ρόλο στην έρευνα σε μεθόδους μεγάλης κλίμακας και από το 2000 και έπειτα αναπτύχθηκαν μέθοδοι για την ανάλυση συνδέσμων για την αναζήτηση στο διαδίκτυο (Google), για την αυτόματη εξαγωγή πληροφορίας, την πολυμεσική IR και τη διαγλωσσική (cross-language) IR. Οι σύγχρονες τάσεις εφαρμογής επικεντρώνονται στα πεδία των διαδικτυακών μηχανών αναζήτησης, στις ψηφιακές βιβλιοθήκες (Digital Libraries), σε peer to peer περιβάλλοντα, στην βιοπληροφορική, σε συστήματα πολυμέσων και στη γεωγραφική ανάκτηση πληροφορίας. Οι γειτονικές επιστημονικές περιοχές που συνδυάζουν ή δανείζονται τεχνικές από το πεδίο της ανάκτησης πληροφορίας είναι οι:

1. Βάσεις Δεδομένων
2. Τεχνητή Νοημοσύνη
3. Εξόρυξη Δεδομένων (Data Mining)
4. Γλωσσική Τεχνολογία / Επεξεργασία Φυσικής Γλώσσας
5. Τεχνικές Μοντελοποίησης
6. Δομές Δεδομένων
7. Αλγόριθμοι (συμπίεση κειμένων, συμπίεση δομών δεδομένων)

2.1.1. Η διαδικασία της Ανάκτησης Πληροφορίας

Λόγω της ποικιλομορφίας των τύπων των δεδομένων στα οποία μπορεί να έχει ταυτόχρονα πρόσβαση ο χρήστης και λόγω του τύπου των δεδομένων στα οποία βασίζεται η παρούσα διπλωματική εργασία (αρχεία κειμένου τύπου .pdf), θεωρούμε ότι κάθε είδους πληροφορία είναι αποθηκευμένη σε έγγραφα (documents). Σε ένα σύστημα ανάκτησης, ο χρήστης πρέπει να μετατρέψει την πληροφοριακή του ανάγκη, σε μορφή ερωτήματος σύμφωνα με τη γλώσσα που του παρέχεται από το σύστημα. Αυτή η διαδικασία, στην παρούσα διπλωματική ανάγεται στην επιλογή ενός κατάλληλου συνόλου λέξεων ώστε να αντιπροσωπεύεται επαρκώς η ανάγκη του. Ακολούθως, το σύστημα θα εντοπίσει όλα τα αντικείμενα πληροφορίας που σχετίζονται με το ερώτημα που έχει εισαχθεί και τα

αποτελέσματα που θα εξαχθούν θα ταξινομηθούν με βάση τον βαθμό συσχέτισης με το ερώτημα. Η αναζήτηση της χρήσιμης πληροφορίας από τον χρήστη είναι μια διαδικασία ανάκτησης πληροφορίας. Στην εικόνα που ακολουθεί παρουσιάζεται η λειτουργία ενός συστήματος ανάκτησης πληροφοριών. Ως είσοδο δέχεται ένα ερώτημα και με βάση τη συλλογή από έγγραφα (corpus) που διαθέτει, αποφασίζει ποια από αυτά είναι σχετικά με το ερώτημα που έθεσε ο χρήστης.

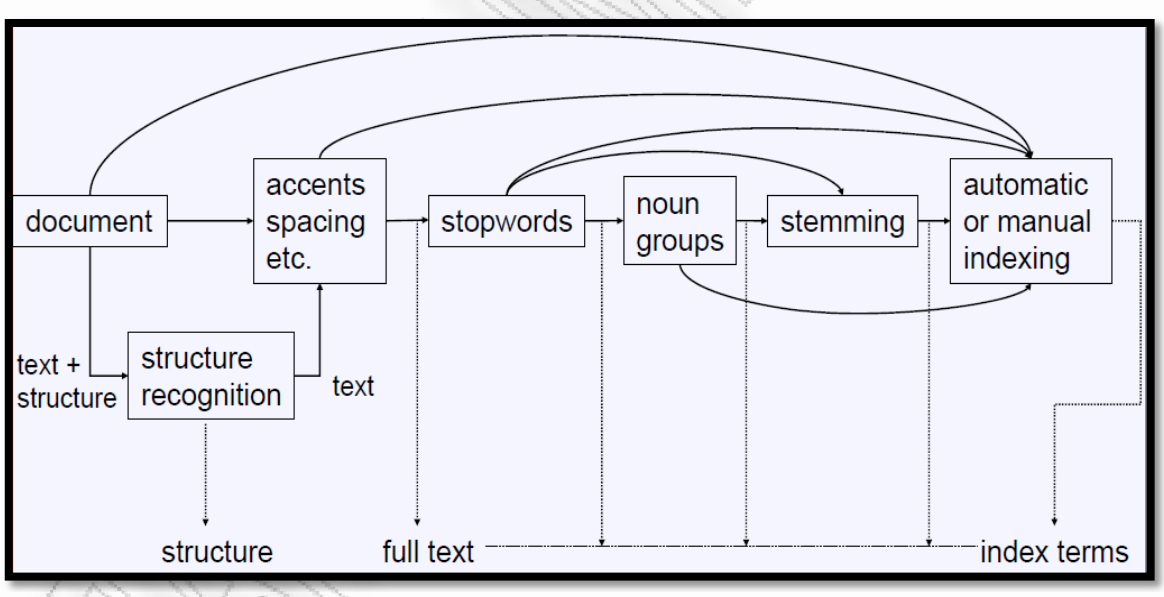


Εικόνα 1 : Διαδικασία ανάκτησης πληροφορίας, πηγή http://nlp.postech.ac.kr/research/previous_research/sir/

Αξιοσημείωτο είναι το γεγονός, ότι τα συστήματα ανάκτησης πληροφορίας σπάνια ψάχνουν απευθείας στα αντικείμενα πληροφορίας, αλλά τις περισσότερες φορές εφαρμόζουν πρωτύτερα μία διαδικασία ευρετηρίασης (indexing) η οποία μειώνει σημαντικά τον χρόνο απόκρισης του συστήματος. Κατά την δεκαετία του '60, η IR είχε γίνει πλέον ένα πολύ δημοφιλές θέμα καθώς πολλοί ερευνητές πίστευαν ότι μπορούν να αυτοματοποιήσουν μέχρι τότε χειροκίνητες διαδικασίες όπως η ευρετηρίαση και η αναζήτηση. Η ευρετηρίαση ή αλλιώς δεικτοδότηση, αναφέρεται στον τρόπο με τον οποίο αναπαρίσταται η πληροφορία για τους σκοπούς της ανάκτησης. Η αναζήτηση αναφέρεται στον τρόπο με τον οποίο δομείται η πληροφορία όταν πραγματοποιείται ένα ερώτημα. Παρόλο που οι δύο αυτές διαδικασίες αποτελούν τον πυρήνα ενός συστήματος IR, οι διαδικασίες που κερδίζουν έδαφος αφορούν τεχνικές αναπαράστασης της πληροφορίας, με σκοπό να βελτιωθεί η αποτελεσματικότητα της ανάκτησης.

2.2. Τεχνικές Αναπαράστασης IR

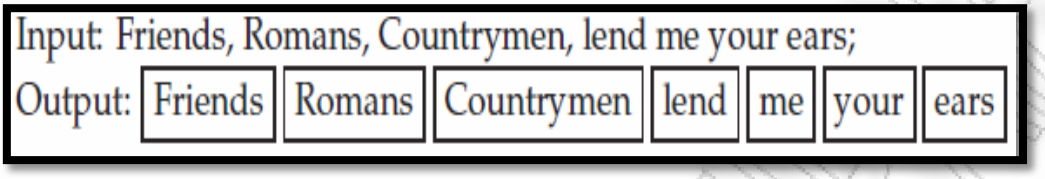
Όλα τα έγγραφα που διατίθενται στη συλλογή του συστήματος υποβάλλονται σε μια διαδικασία προεπεξεργασίας ώστε να μετατραπούν σε μία μορφή κατάλληλη για την εσωτερική τους αναπαράσταση στο σύστημα όπως παρουσιάζεται στην εικόνα 2. Το πλήρες κείμενο είναι σίγουρα η πιο ολοκληρωμένη αναπαράσταση ενός κειμένου αλλά η χρήση της συνήθως δεν είναι υπολογιστικά αποδοτική. Το σημαντικό στοιχείο που χαρακτηρίζει την επιστήμη της ανάκτησης πληροφορίας, είναι πως δεν περιορίζεται στην εύρεση εγγράφων, για παράδειγμα, τα οποία περιέχουν απλά μερικούς από τους όρους του ερωτήματος που έχει εισαχθεί, αλλά αναγνωρίζει το γεγονός ότι ο χρήστης του συστήματος επιθυμεί να ανακτήσει πληροφορίες γύρω από ένα συγκεκριμένο θέμα. Για το λόγο αυτό, προσπαθεί να εντοπίσει συσχετίσεις μεταξύ των όρων του ερωτήματος και των όρων των εγγράφων, έτσι ώστε να μεταβεί από τους όρους σε θεματικές περιοχές. Για να το επιτύχει αυτό, εφαρμόζει διάφορες μεθόδους ανάλυσης τόσο των όρων των εγγράφων όσο και των όρων του ερωτήματος, όπως απαλοιφή συχνών λέξεων (stopwords), απομάκρυνση επιθημάτων και προθεμάτων από τις λέξεις (stemming) κτλ. Επίσης, η διαδικασία της προεπεξεργασίας είναι σημαντική διότι ένα σύστημα ανάκτησης πληροφορίας, διαχειρίζεται στοιχεία όχι επαρκώς δομημένα, χαλαρά ορισμένα και αμφίσημα, αυτό πηγάζει από τη δυσκολία του χρήστη να περιγράψει με ακρίβεια την πληροφοριακή του ανάγκη.



Εικόνα 2: Φάσεις Προεπεξεργασίας Κειμένου, πηγή http://people.ischool.berkeley.edu/~hearst/irbook/1/node3.html

2.2.1. Δειγματοποίηση (Tokenization)

Η δειγματοποίηση (Tokenization) είναι η διαδικασία χωρισμού του κειμένου σε κομμάτια που ονομάζονται δείγματα - tokens, αφαιρώντας την ίδια στιγμή συγκεκριμένους χαρακτήρες όπως τα σημεία στίξης. Ένα παράδειγμα tokenization ακολουθεί στην εικόνα 3.



Εικόνα 3 : Tokenization

Τα tokens συχνά αναφέρονται ως όροι ή λέξεις, αλλά είναι σημαντικό να γίνεται ο διαχωρισμός, ένα token είναι ένα στιγμιότυπο από μια ακολουθία χαρακτήρων σε ορισμένα έγγραφα που ομαδοποιούνται μαζί ως σημασιολογικές μονάδες επεξεργασίας. Το μείζον πρόβλημα κατά τη φάση του tokenization είναι ποια είναι τα σωστά tokens που θα πρέπει να χρησιμοποιηθούν. Αρχικά, αφαιρούνται τα κενά διαστήματα και τα σημεία στίξης, μετατρέπονται όλοι οι χαρακτήρες σε μικρά γράμματα, εξαίρεση μπορεί να γίνει αν η χρήση κεφαλαίων γίνεται στη μέση μίας πρότασης (π.χ. General Motors). Το σύνολο από tokens είναι η βασική έξοδος της διαδικασίας ανάλυσης. Κατά τη διάρκεια της ευρετηρίασης τα πεδία που προσδιορίζονται για ανάλυση επεξεργάζονται και οι σημαντικές ιδιότητες από κάθε token εγγράφονται στο ευρετήριο. Κάθε token αντιπροσωπεύει μια ξεχωριστή λέξη του κειμένου και μεταφέρει την τιμή του κειμένου καθώς επίσης και κάποια μετά-δεδομένα.

2.2.2. Λέξεις Αποκλεισμού (Stopwords)

Σημαντικό τμήμα στην προεπεξεργασία είναι ο αποκλεισμός των πολύ συχνά εμφανιζόμενων λέξεων (stopwords) διότι δεν έχουν κάποιο πληροφοριακό χαρακτήρα και είναι άσχετες με το θέμα του κειμένου. Τέτοιες λέξεις είναι συνήθως άρθρα (ο, το κτλ.), αντωνυμίες, προθέσεις, σύνδεσμοι κ.ά. Οι λέξεις αυτές αφαιρούνται ώστε να παραμείνει το κείμενο αποτελούμενο μόνο με από όρους που καταδεικνύουν το περιεχόμενό του. Οι λέξεις αποκλεισμού μπορούν να θεωρηθούν ως ‘θόρυβος’ του κειμένου αφού επικαλύπτουν την πληροφορία του κειμένου και εμφανίζονται συχνά. Το όφελος είναι πολύ μεγάλο αφού το μέγεθος του ευρετηρίου παρουσιάζει μειώσεις έως και 40%. Το σύνολο με τις λέξεις αποκλεισμού εξαρτάται από τη γλώσσα και τη συλλογή εγγράφων, συνεπώς μπορεί εύκολα να εμπλουτιστεί ανάλογα με την γλώσσα και τις ιδιαιτερότητες της κάθε συλλογής ώστε να επιτευχθεί καλύτερο αποτέλεσμα.

2.2.3. Κανονικοποίηση (Normalization)

Η τεχνική «κανονικοποίησης» των όρων είναι πολύ σημαντική διότι θα πρέπει να δίνεται η δυνατότητα εύρεσης της λέξης USA στην περίπτωση που ο χρήστης για παράδειγμα πραγματοποιήσει αναζήτηση με τη λέξη U.S.A. Θα πρέπει να μπορεί να γίνει το ταίριασμα μεταξύ αυτών των όρων, όπως και στην περίπτωση των colour και color. Οι μέθοδοι που χρησιμοποιούνται για την επίτευξη της, είναι ο έμμεσος καθορισμός ισοδυναμίας των όρων με χρήση κανόνων χαρτογράφησης ή η διατήρηση σχέσεων μεταξύ μη κανονικοποιημένων δειγμάτων (tokens), για παράδειγμα το ερώτημα προς εκτέλεση με τον όρο window θα οδηγήσει στην αναζήτηση των όρων window και windows.

2.2.4. Στελέχωση Κειμένου (Stemming)

Η μέθοδος stemming είναι πολύ διαδεδομένη τεχνική ανάκτησης πληροφορίας για αυτό και έχει εφαρμοστεί σε πολλά συστήματα IR. Οι λέξεις υποβιβάζονται στη ρίζα τους με στόχο την ανεξαρτησία τους από τις μορφολογικές παραλλαγές τους (π.χ. αυτοκίνητο, αυτοκίνητα, αυτοκινήτων). Απαλείφονται οι καταλήξεις και τα προθέματα για την ανάκτηση των κειμένων που περιέχουν μορφολογικές παραλλαγές των λέξεων της επερώτησης. Παράδειγμα αποτελεί η περίπτωση που ένας χρήστης ενδιαφέρεται να πάρει πληροφορίες σχετικά με τη λέξη "computers" και εισάγει στη μηχανή αναζήτησης αυτή τη λέξη – κλειδί, τότε είναι πολύ πιθανόν να ενδιαφέρεται και για έγγραφα τα οποία περιέχουν τη λέξη - κλειδί "computer" χωρίς το γράμμα 's'. Βασικός στόχος είναι, ουσιαστικά, η βελτίωση της ανάκτησης πληροφορίας με αυτόματο χειρισμό των καταλήξεων των λέξεων και τη μείωση τους στις ρίζες τους. Η μέθοδος stemming, θα πάρει τη λέξη – κλειδί που έχει εισαγάγει ο χρήστης στη μηχανή αναζήτησης και θα τη μειώσει στη ρίζα της αφαιρώντας οποιοδήποτε επίθημα ή πρόθεμα από αυτή. Ακολούθως, η ρίζα της λέξης - κλειδί θα συγκριθεί με τις λέξεις που υπάρχουν σε κάθε έγγραφο του συνόλου στο οποίο εκτελείται η αναζήτηση και θα επιστραφούν έγγραφα που περιέχουν λέξεις με τη ρίζα αυτή.

Παρουσιάζεται ένα επιπλέον παράδειγμα υποθέτοντας ότι ο χρήστης έχει εισαγάγει τη λέξη – κλειδί "viewer" στη μηχανή αναζήτησης. Τότε, η λέξη αυτή θα μειωθεί στη ρίζα της, δηλαδή στο "view", με αφαίρεση του επιθέματος – er και θα επιστραφούν έγγραφα που περιέχουν λέξεις όπως view, viewer, preview, review κτλ. Θα μπορούσαμε να πούμε ότι η μέθοδος stemming καθορίζει στην ουσία μία ομάδα συνωνύμων για μία συγκεκριμένη λέξη.

Μέχρι σήμερα έχουν αναπτυχθεί πολλοί και διάφοροι stemming αλγόριθμοι ή αλλιώς stemmers. Ο πρώτος αλγόριθμος που εντοπίζουμε είναι ο αλγόριθμος Lovins ο οποίος αναπτύχθηκε το 1968. Υπάρχουν, όμως, κι άλλοι αλγόριθμοι όπως ο αλγόριθμος Nice, ο οποίος έχει υλοποιηθεί στα πλαίσια του συστήματος IRIS για ανάκτηση πληροφορίας στο πανεπιστήμιο του North Carolina και αποτελεί ένα συνδυασμό τεσσάρων διαφορετικών stemmers. Ωστόσο, ο γνωστότερος αλγόριθμος ο οποίος εφαρμόζεται πιο συχνά, είναι ο αλγόριθμος του Porter. Πολλές εφαρμογές του αλγορίθμου αυτού εκδόθηκαν και διανεμήθηκαν ελεύθερα αλλά περιείχαν αρκετά λάθη. Για το λόγο αυτό, ο Martin Porter εξέδωσε επίσημα μία εφαρμογή ελεύθερου λογισμικού του αλγορίθμου το 2000, ενώ αργότερα επέκτεινε την εργασία αυτή με τη δημιουργία μίας γλώσσας για stemming αλγορίθμους, το Snowball, και εφάρμοσε ένα βελτιωμένο stemmer για την αγγλική γλώσσα.

2.2.5. Ευρετηρίαση - Δεικτοδότηση (Indexing)

Σε αυτό το τμήμα της διαδικασίας επιλέγονται οι λέξεις για την ευρετηρίαση. Το ευρετήριο είναι μια δομή πολύ σημαντική διότι επιταχύνει την διαδικασία της αναζήτησης και είναι ιδιαίτερα κατάλληλο για μεγάλες συλλογές όπως στην περίπτωση της συλλογής εγγράφων που θα διαχειριστούμε στα πλαίσια της παρούσας εργασίας.

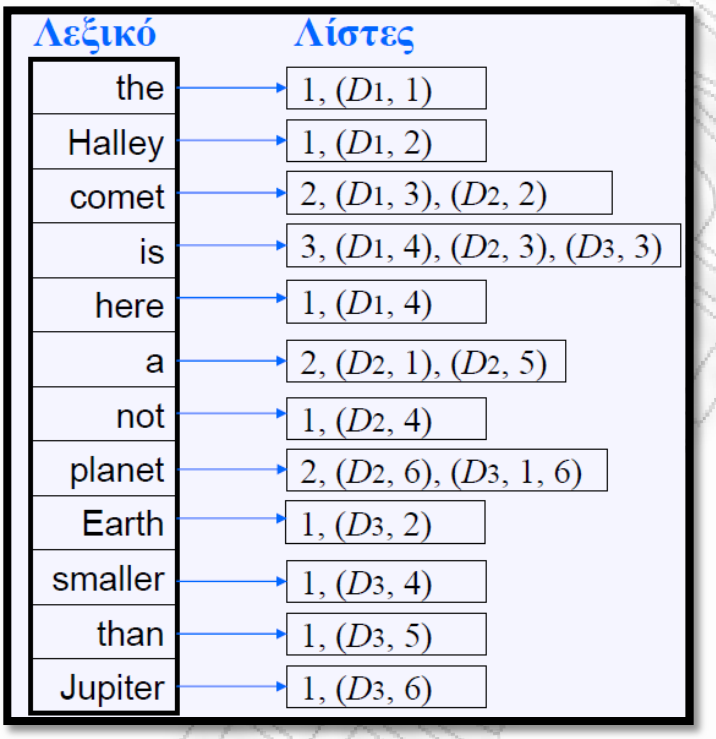
Γενικά οι όροι του ευρετηρίου είναι συνήθως ουσιαστικά, διότι αναπαριστούν μία έννοια χωρίς την ανάγκη εμφανίζονται δίπλα σε άλλο μέρος του λόγου και η σημασιολογία τους είναι εύκολα αντιληπτή. Οι σύνδεσμοι και τα επιρρήματα θεωρούνται ότι έχουν κυρίως συμπληρωματικό χαρακτήρα.

Στο ευρετήριο (index) καταγράφεται για κάθε έγγραφο κατά πόσον περιέχει κάθε όρο που περιέχεται σε κάθε έγγραφο της συλλογής. Σε μια συλλογή εγγράφων υποθέτουμε ότι κάθε έγγραφο έχει ένα μοναδικό σειριακό αριθμό γνωστός ως αναγνωριστικό του εγγράφου (docID). Κατά τη διάρκεια δημιουργίας ευρετηρίου ανατίθενται διαδοχικοί ακέραιοι σε κάθε νέο έγγραφο που εμφανίζεται. Μία από τις απλή μορφή ευρετηρίου μπορεί να είναι ένας δυαδικός πίνακας (term-document incidence matrix – μητρώο παρουσίας), όπως παρουσιάζεται στον παρακάτω πίνακα.

	Document 1	**Document 2**	**Document 3**	**Document 4**	**Document 5**	**…**
Term 1	0	1	1	0	1	…
Term 2	1	0	1	0	0	…
Term 3	0	1	1	1	1	…
Term 4	0	1	0	0	0	…
Term 5	1	0	0	1	0	…
…	…	…	…	…	…	…

Πίνακας 1: Παράδειγμα Δυαδικού Ευρετηρίου

Στην περίπτωση που η συλλογή εγγράφων είναι πολύ μεγάλη ο πίνακας θα είναι πολύ αραιός, μία καλύτερη προσέγγιση, την οποία χρησιμοποιούν τα περισσότερα συστήματα IR, αποτελεί η καταγραφή μόνο των θέσεων που έχουν 1, δηλαδή μόνο τους όρους που εμφανίζονται. Ως αποτέλεσμα λαμβάνεται το λεγόμενο ανεστραμμένο ευρετήριο (inverted index) όπου για κάθε όρο υπάρχει μία λίστα με όλα τα έγγραφα που τον περιέχουν. Αποτελείται από δύο μέρη, το λεξικό (lexicon), το οποίο αποτελείται από όλες τις λέξεις που εμφανίζονται στα έγγραφα και τις λίστες εμφανίσεων (occurrence lists), οι οποίες παρέχουν την πληροφορία εμφάνισης των λέξεων στα έγγραφα.



Εικόνα 4 : Ανεστραμμένο Ευρετήριο

Η διαδικασία για την κατασκευή του περιλαμβάνει, τα παρακάτω βήματα:

1. Το σύστημα βρίσκει τα έγγραφα που θα γίνουν indexed.
2. Γίνεται tokenization στο κείμενο.
3. Γίνεται γλωσσολογική επεξεργασία στα tokens.
4. Δημιουργείται το index.

2.3. Μοντέλα Ανάκτησης Πληροφορίας

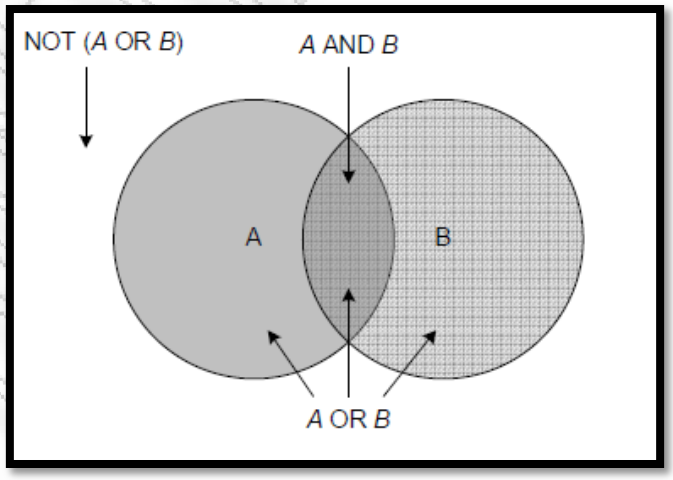
Τα κλασσικά μοντέλα στην ανάκτηση πληροφορίας θεωρούν ότι κάθε κείμενο περιγράφεται από ένα σύνολο από αντιπροσωπευτικές λέξεις κλειδιά, οι οποίοι ονομάζονται όροι δεικτοδότησης. Ένας όρος ευρετηρίασης (index term) είναι μία λέξη, της οποίας το σημασιολογικό περιεχόμενο περικλείει ένα μέρος της θεματολογίας του κειμένου. Λαμβάνοντας υπόψη ότι πολλοί από τους όρους ενός εγγράφου δεν προσφέρουν σημαντική πληροφορία (όπως τα άρθρα και τα επίθετα) στις περισσότερες περιπτώσεις όπως αναφέρθηκε παραπάνω γίνεται προεπεξεργασία των εγγράφων ώστε να διατηρηθούν μόνο οι λέξεις που περιέχουν σημαντική πληροφορία. Τα έγγραφα δηλαδή αναπαρίστανται ως σύνολα όρων, οι οποίοι συνοψίζουν το περιεχόμενο τους. Η αναπαράσταση των εγγράφων ως συλλογές όρων δεν σημαίνει ότι όλοι οι όροι έχουν την ίδια ισχύ ως προς την περιγραφή του εγγράφου. Η ερμηνεία ενός όρου συχνά μπορεί να δίνει μια γενικευμένη ή και ασαφή περιγραφή ενός εγγράφου. Αυτοί οι όροι είναι που εμφανίζονται με μεγάλη συχνότητα στην πλειονότητα των

εγγράφων μίας συλλογής. Αν για παράδειγμα έχουμε μια συλλογή εγγράφων σχετικά με τους υπολογιστές. Η λέξη «computer» σε μία τέτοια συλλογή εμφανίζεται με μεγάλη βεβαιότητα σχεδόν σε κάθε κείμενο και αν και περιγράφει κάτι συγκεκριμένο, δεν είναι αντιπροσωπευτική του εγγράφου στο οποίο εμφανίζεται. Αντίθετα αν μία λέξη εμφανίζεται σε μικρό εύρος εγγράφων, τότε είναι σχεδόν σίγουρο ότι έχει μεγαλύτερη βαρύτητα στην περιγραφή ενός κειμένου. Για παράδειγμα η λέξη «inheritance», θα εμφανίζεται σίγουρα σε πολύ λιγότερα έγγραφα απ' ότι η λέξη «computer». Η εμφάνισή της ως όρος στο ευρετήριο, μας οδηγεί στο συμπέρασμα ότι το συγκεκριμένο έγγραφο έχει θεματολογία σχετικά με την κληρονομικότητα στον αντικειμενοστραφή προγραμματισμό. Για να προσομοιώσουμε το παραπάνω γεγονός, δηλαδή διαφορετικοί όροι να έχουν διαφορετική βαρύτητα ως προς την ευρετηρίαση των εγγράφων, σε κάθε όρο του ευρετηρίου ανατίθεται ένα αριθμητικό βάρος. Αυτός ο αριθμός δείχνει πόσο αντιπροσωπευτικός είναι ο όρος για το κάθε έγγραφο, τα βάρη μεταξύ τους είναι ανεξάρτητα.

Υπάρχουν τρία βασικά μοντέλα τα οποία χρησιμοποιούνται για την αναπαράσταση των εγγράφων και των ερωτημάτων. Αυτά είναι το Δυαδικό (Boolean) μοντέλο, το Διανυσματικό (Vector Space) μοντέλο και το Πιθανοτικό (Probabilistic) μοντέλο.

2.3.1. Δυαδικό (Boolean) μοντέλο

Το Δυαδικό (Boolean) μοντέλο είναι ένα μοντέλο ανάκτησης πληροφοριών που βασίζεται στην θεωρία συνόλων. Τόσο τα κείμενα όσο και τα ερωτήματα αντιμετωπίζονται ως ένα σύνολο από όρους δεικτοδότησης. Σύμφωνα με αυτό τα ερωτήματα σχηματίζονται με τη βοήθεια των λογικών τελεστών AND, OR και NOT, δηλαδή οι όροι συνδυάζονται μεταξύ τους χρησιμοποιώντας αυτούς τους τελεστές. Κάθε όρος του ερωτήματος ή είναι παρών (το βάρος λαμβάνει τιμή 1 αν ο όρος υπάρχει) ή δεν είναι σε κάποιο έγγραφο (το βάρος λαμβάνει τιμή 0 αν ο όρος δεν υπάρχει). Επίσης, κάθε έγγραφο της συλλογής ή είναι σχετικό ή δεν είναι, δεν υπάρχει η έννοια της μερικής ικανοποίησης. Ένα έγγραφο είναι μέρος της απάντησης αν ικανοποιεί τους περιορισμούς του ερωτήματος.



Διάγραμμα 1: Αναπαράσταση λογικής έκφρασης

Τα πλεονεκτήματά του είναι :

- Η απλότητά του.
- Εύκολα κατανοητό αφού στηρίζεται στη θεωρία συνόλων.

Τα μειονεκτήματά του είναι :

- Δεν υπάρχει υποστήριξη για μερική ταύτιση (partial matching) .
- Δεν υπάρχει βαθμολόγηση των αποτελεσμάτων.
- Η ερώτηση πρέπει να διατυπωθεί με λογική έκφραση, το οποίο δεν είναι πάντα εύκολο για όλες τις κατηγορίες χρηστών.
- Τα ερωτήματα που διατυπώνονται είναι τις περισσότερες φορές πολύ απλοϊκά.
- Πολλές φορές επιστρέφει πάρα πολλά έγγραφα (απλές λογικές εκφράσεις) και άλλοτε πάρα πολύ λίγα (πολύπλοκες λογικές εκφράσεις). Αυτό οφείλεται στη χρήση των λογικών εκφράσεων για τη διατύπωση των ερωτημάτων και στο γεγονός ότι δεν χρησιμοποιούνται βάρη στους όρους τα οποία να δηλώνουν πόσο σημαντικός είναι ένας όρος για ένα έγγραφο.

2.3.2. Διανυσματικό (Vector Space) μοντέλο

Το διανυσματικό (Vector Space) μοντέλο, αντιμετωπίζει την ανεπάρκεια της ανάθεσης δυαδικών βαρών και εισάγει ένα υπόβαθρο που επιτρέπει προσεγγιστικό ταίριασμα. Αναπτύχθηκε στα πλαίσια του συστήματος SMART (Salton, c. 1970) και χρησιμοποιείται ευρέως από μηχανές αναζήτησης στο διαδίκτυο.

Αποτελείται από m διαστάσεις, όπου m είναι ο αριθμός των μοναδικών όρων που χρησιμοποιούνται στα έγγραφα. Το κάθε έγγραφο (document) και ερώτημα αναπαριστάται ως ένα διάνυσμα με συντεταγμένες w_{ij} (όρος i, έγγραφο j). Έτσι λέμε ότι το μοντέλο είναι αλγεβρικό. Τα βάρη που ανατίθενται στους όρους του ευρετηρίου, τόσο για τα έγγραφα όσο και για τα ερωτήματα είναι μη δυαδικά και χρησιμοποιούνται για τον υπολογισμό του βαθμού ομοιότητας μεταξύ του ερωτήματος και του αποθηκευμένου εγγράφου. Έπειτα τα έγγραφα διατάσσονται με φθίνουσα σειρά, με κριτήριο το βαθμό ομοιότητάς τους με το ερώτημα του χρήστη. Με αυτόν τον τρόπο λαμβάνονται υπόψη και έγγραφα που δεν ικανοποιούν πλήρως τις συνθήκες του ερωτήματος και το τελικό αποτέλεσμα είναι περισσότερο ακριβές από την ανάκτηση χρησιμοποιώντας το Boolean μοντέλο.

Τα πλεονεκτήματά του είναι :

- Κατάταξη – αξιολόγηση κειμένων με βάση τους όρους τους .
- Partial matching .
- Καλύτερη απόδοση.
- Το γρηγορότερο.

Τα μειονεκτήματά του είναι :

- Θεώρηση ανεξαρτησίας των όρων (π.χ. αγνοεί συνώνυμους όρους).
- Χάνεται συντακτική πληροφορία (π.χ. δομή φράσεων, σειρά λέξεων, εγγύτητα λέξεων).

- Ψάχνει μόνο τη λεξιλογική ομοιότητα μεταξύ των όρων των κειμένων και όχι τη σημασιολογική. Συνεπώς, μπορεί να χάνεται σημασιολογική πληροφορία (π.χ. έννοιες λέξεων).
- Έλλειψη ελέγχου που παρέχει το μοντέλο Boolean (π.χ. που απαιτεί να εμφανίζεται ένας όρος σε ένα κείμενο).

2.3.2.1. Υπολογισμός Σημαντικότητας Όρων

Υπάρχουν διάφοροι τρόποι για να υπολογιστεί το βάρος για τους όρους, ένας τρόπος που δηλώνει τη σημαντικότητα (βάρος) του όρου είναι η συχνότητα εμφάνισής του στο έγγραφο. Η συχνότητα εμφάνισης (term frequency - ft), από το term (όρος) και frequency (συχνότητα), αντιστοιχεί στο πλήθος των εμφανίσεων ενός όρου σε ένα δεδομένο σύνολο κείμενων και η τιμή του κανονικοποιείται ως εξής :

$$ft_i = \frac{n_i}{\sum_j n_j}$$

Όπου n_i είναι ο αριθμός των εμφανίσεων της λέξης – κλειδί (όρου) i στο συγκεκριμένο έγγραφο και ο παρονομαστής το άθροισμα του αριθμού των εμφανίσεων κάθε όρου j (n_j) που περιέχεται στο έγγραφο. Ουσιαστικά, ο παρονομαστής αντιστοιχεί στο συνολικό αριθμό των όρων του εγγράφου που έχουν ευρετηριαστεί. Η κανονικοποίηση διαιρώντας με το άθροισμα $\sum_j n_j$ κρίνεται απαραίτητη διότι εάν ορίζαμε το βάρος ίσο με τη συχνότητα του όρου στο έγγραφο θα υπήρχε πρόβλημα στην περίπτωση που τα έγγραφα είχαν μεγάλη διαφορά ως προς το μέγεθός τους. Τότε οι όροι που θα εμφανίζονταν σε μεγάλα έγγραφα θα είχαν μεγαλύτερο βάρος διότι θα αυξανόταν η πιθανότητα ύπαρξής τους στο έγγραφο. Με τον τρόπο αυτό αποφεύγεται η ανάθεση μεγαλύτερου βάρους σε όρους που περιέχονται σε έγγραφα τα οποία είναι απλά μεγάλα σε μέγεθος και πιθανόν να περιέχουν περισσότερες φορές τη λέξη - κλειδί, ανεξάρτητα από τη σημασία που έχει πραγματικά η λέξη - κλειδί στο συγκεκριμένο έγγραφο.

Το πλήθος των εμφανίσεων ενός όρου σε ένα έγγραφο δηλώνει τη σημαντικότητα του όρου για το έγγραφο αυτό. Ωστόσο θα πρέπει να παρατηρηθεί ότι όροι που εμφανίζονται σε πολλά έγγραφα έχουν μικρή διακριτική ικανότητα. Δηλαδή, το γεγονός ότι εμφανίζονται πολλές φορές στα έγγραφα μειώνει τη σημαντικότητά τους. Για παράδειγμα, σε μία συλλογή εγγράφων που περιλαμβάνει άρθρα σχετικά με την επιστήμη της ανάκτησης πληροφορίας, είναι πολύ πιθανό σε κάποια έγγραφα να περιέχεται πολλές φορές ο όρος ανάκτηση. Συνεπώς το βάρος αυτού του όρου θα πρέπει να είναι μικρό, καθώς δεν αποτελεί αντιπροσωπευτική λέξη για κανένα έγγραφο της συλλογής.

Η παραπάνω παρατήρηση οδήγησε στη χρήση ενός νέου παράγοντα την επίλυση του προβλήματος αυτού κατά τον υπολογισμό των βαρών. Αυτός ο παράγοντας καλείται αντίστροφη συχνότητα εγγράφου (inverse document frequency) ενός όρου και συμβολίζεται με idf. Ο παράγοντας αυτός λαμβάνει μικρές τιμές για τις λέξεις - κλειδιά που εμφανίζονται συχνά και μεγαλύτερες τιμές για τις λέξεις – κλειδιά που εμφανίζονται σπάνια ώστε να γίνει σωστά η ταξινόμηση των αποτελεσμάτων. Αν συμβολίσουμε με D το πλήθος των εγγράφων της συλλογής και με $f_{w,d}$ το πλήθος των εγγράφων που περιέχουν τον όρο t, τότε ορίζεται ως :

$$idft = \log\left(\frac{D}{f_{w,d}}\right)$$

Για να αποφευχθεί η διαίρεση με το 0, σε περίπτωση που η λέξη – κλειδί δεν εμφανίζεται σε κανένα έγγραφο πολλές φορές προστίθεται στον παρανομαστή μία μονάδα.

Ως βάρος στη διαδικασία στάθμισης χρησιμοποιείται ο συντελεστής tf-idf. Πρόκειται για τον συνδυασμό της συχνότητας των όρων και της αντίστροφης συχνότητας εγγράφων για την παραγωγή ενός σύνθετου βάρους ώστε να αποδοθεί σε κάθε όρο σε κάθε έγγραφο. Το βάρος που θα λάβει ο κάθε όρος ορίζεται από το παρακάτω γινόμενο :

$$tf - idf = tf * idf$$

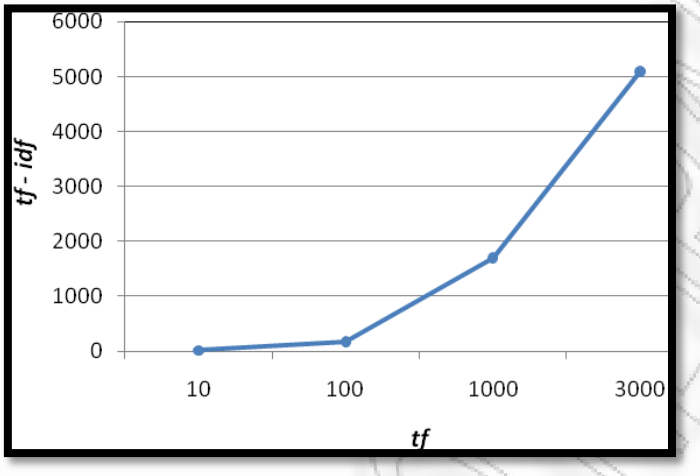
Ο παραπάνω δείκτης αποτελεί ένα στατιστικό μέτρο που χρησιμοποιείται για να αξιολογήσει πόσο σημαντικός είναι ένας όρος σε ένα έγγραφο μιας συλλογής. Η σημασία αυξάνει αναλογικά με τον αριθμό των εμφανίσεων του όρου στο έγγραφο αλλά αντισταθμίζεται από την συχνότητα του όρου στο σώμα των εγγράφων. Δηλαδή, λέξεις – κλειδιά του ερωτήματος που εμφανίζονται σε ένα μικρό σύνολο εγγράφων έχουν μεγαλύτερο βάρος (tf–idf) από λέξεις – κλειδιά που είναι πολύ συνηθισμένες και εμφανίζονται σε πολλά έγγραφα.

Ο δείκτης tf-idf αναθέτει στον όρο t ένα βάρος στο έγγραφο d το οποίο είναι:

- Υψηλότερο όταν ο όρος t εμφανίζεται πολλές φορές μέσα σε μια μικρή συλλογή από έγγραφα.
- Χαμηλότερο όταν ο όρος t εμφανίζεται λιγότερες φορές μέσα σε κάποιο έγγραφο ή εμφανίζεται σε πολλά έγγραφα.
- Ελάχιστο όταν ο όρος t εμφανίζεται σχεδόν σε όλα τα έγγραφα.

Στη γραφική παράσταση του σχήματος 1, παρατηρούμε πως μεταβάλλεται η τιμή του tf-idf, δηλαδή το βάρος που ανατίθεται σε κάθε όρο, καθώς αυξάνεται η συχνότητα εμφάνισής του (tf) στο ίδιο σύνολο εγγράφων. Βλέπουμε πως όσο αυξάνεται η συχνότητα εμφάνισης της λέξης στα έγγραφα τόσο αυξάνεται και το βάρος που της αναλογεί.

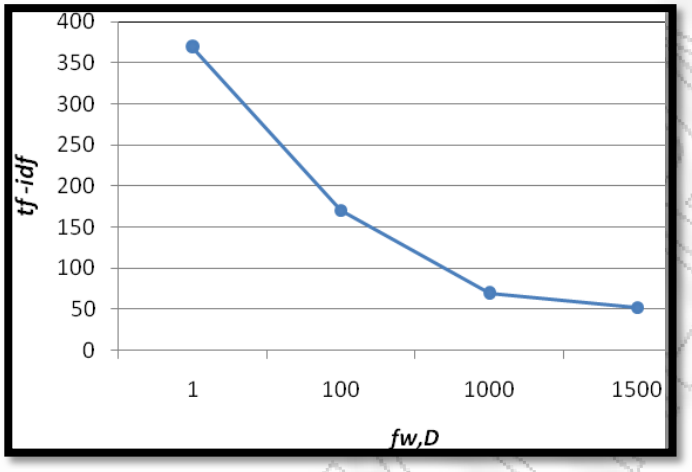
tf	***D***	***fw,D***	***idf***	***tf-idf***
10	5000	100	1,69	16,9
100	5000	100	1,69	169,9
1000	5000	100	1,69	1698,97
3000	5000	100	1,69	5096,91



Σχήμα 1: Γραφική παράσταση του tf-idf συναρτήσει του παράγοντα tf

Στη γραφική παράσταση του σχήματος 2, παρατηρούμε πως μεταβάλλεται η τιμή του tf-idf καθώς αυξάνεται ο αριθμός των εγγράφων στα οποία εμφανίζεται η λέξη - κλειδί. Βλέπουμε πως όσο μεγαλώνει το σύνολο των εγγράφων στα οποία εμφανίζεται τόσο μειώνεται το βάρος που της αναλογεί.

tf	*D*	*fw,D*	*idf*	*tf-idf*
100	5000	1	3,69	369,9
100	5000	100	1,69	169,9
100	5000	1000	0,69	69,9
100	5000	1500	0,522	52,28



Σχήμα 2: Γραφική παράσταση του tf-idf συναρτήσει του παράγοντα $f_{w,d}$

Επιπροσθέτως, ο δείκτης tf-idf χρησιμοποιείται συχνά στον τομέα της ανάκτησης πληροφορίας, της εξόρυξης κειμένων (text mining) και συχνά από τις μηχανές αναζήτησης ως κεντρικό εργαλείο για τη βαθμολόγηση και ιεράρχηση της σχετικότητας των εγγράφων βάσει του ερωτήματος που υποβάλει ο χρήστης.

2.3.2.2. Υπολογισμός ομοιότητας εγγράφων και ερωτημάτων

Στο διανυσματικό μοντέλο ανατίθενται βάρη και στους όρους του ευρετηρίου του ερωτήματος, με αυτόν τον τρόπο τα έγγραφα και το ερώτημα του χρήστη αναπαρίστανται ως διανύσματα διάστασης t (όσοι οι όροι που χρησιμοποιούνται). Ο τρόπος υπολογισμού της ομοιότητας στο διανυσματικό μοντέλο είναι ανεξάρτητος από τον τρόπο προσδιορισμού των βαρών. Μία μέθοδος για την ποσοτικοποίηση της ομοιότητας ενός ερωτήματος και ενός εγγράφου της συλλογής είναι η ευκλείδια απόσταση μεταξύ των διανυσματικών αναπαραστάσεων του εγγράφου και του ερωτήματος. Όσο θα αυξάνεται η απόσταση τόσο θα μειώνεται η ομοιότητα μεταξύ του ερωτήματος και του εγγράφου. Σε αυτήν την περίπτωση υπάρχει ένα σοβαρό πρόβλημα που πρέπει να αναφερθεί. Συνήθως, το ερώτημα είναι αρκετά μικρότερο σε σχέση με τα έγγραφα της συλλογής. Αυτό σημαίνει ότι οι περισσότερες συνιστώσες του διανύσματος του ερωτήματος θα είναι μηδενικές. Επίσης, όσο μεγαλύτερο είναι ένα έγγραφο, τόσο αυξάνει ο αριθμός των μη-μηδενικών συνιστωσών. Αυτό σημαίνει ότι ακόμα και αν τα μεγάλα έγγραφα σχετίζονται με το ερώτημα, λόγω της ευκλείδιας απόστασης, η απόστασή τους από το ερώτημα θα είναι μεγάλη.

Μία άλλη προσέγγιση για τον υπολογισμό της ομοιότητας μεταξύ ερωτήματος και εγγράφων είναι η χρησιμοποίηση του εσωτερικού γινομένου των διανυσμάτων :

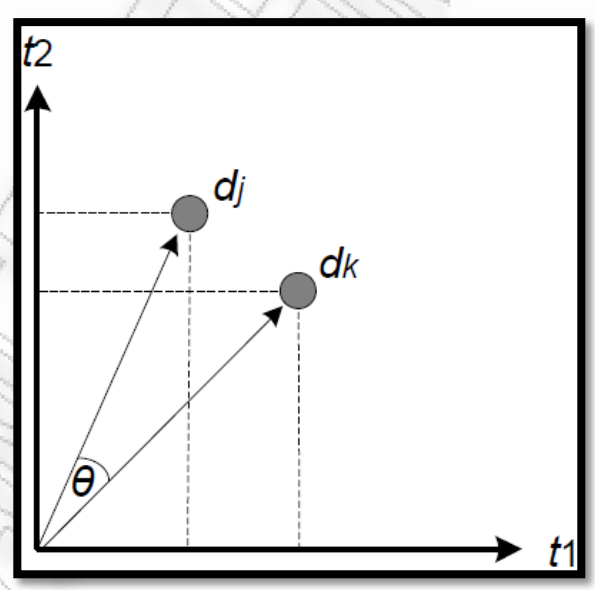
$$Sim\,(q,d) = \vec{V}(q) * \vec{V}(d)$$

Όσο πιο όμοια είναι τα διανύσματα $\vec{q}$ και $\vec{d}$ τόσο μεγαλύτερες τιμές λαμβάνει η παραπάνω συνάρτηση. Το αντίστροφο συμβαίνει όταν τα διανύσματα είναι ανόμοια. Η εφαρμογή του εσωτερικού γινομένου για τον υπολογισμό της ομοιότητας έχει το μειονέκτημα ότι λειτουργεί ανασταλτικά για τα μικρότερα έγγραφα, σε αντίθεση με την ευκλείδια απόσταση που αδικεί τα μεγαλύτερα. Για να ξεπεραστεί αυτό το πρόβλημα, προτάθηκε ως μέτρο του βαθμού ομοιότητας το συνημίτονο της εμπεριεχομένης γωνίας (cosine similarity) των δύο διανυσμάτων. Το συνημίτονο της γωνίας των δύο διανυσμάτων, ενός αρχείου d και μιας ερώτησης q υπολογίζεται ως :

$$\cos(\theta) = \frac{\vec{V}(q) * \vec{V}(d)}{|\vec{V}(q)| * |\vec{V}(d)|}$$

Όπου αριθμητής είναι το εσωτερικό γινόμενο των διανυσμάτων $\vec{V}(q)$ και $\vec{V}(d)$ και παρανομαστής είναι το γινόμενο των ευκλείδιων αποστάσεών τους. Όταν μικραίνει η γωνία θ, μεγαλώνει το συνημίτονο της γωνίας και αντιστρόφως. Όταν τα διανύσματα ταυτίζονται, τότε η γωνία είναι μηδέν μοιρών, συνεπώς το συνημίτονό της λαμβάνει τιμή 1. Όταν τα διανύσματα είναι κάθετα μεταξύ τους η γωνία είναι ορθή (90°) και το συνημίτονό της λαμβάνει τιμή 0.

Στο παρακάτω σχήμα παρουσιάζεται ένα παράδειγμα προσδιορισμού της γωνίας μεταξύ δύο διανυσμάτων για το χώρο των δύο διαστάσεων (δύο όρων).



Σχήμα 3: Γωνία μεταξύ δύο διανυσμάτων (ερώτημα - έγγραφο) σε δύο διαστάσεις (δύο όροι)

Η εφαρμογή του τύπου του συνημιτόνου είναι ανεξάρτητη από τον τρόπο υπολογισμού των βαρών. Η μέθοδος εφαρμόζεται τόσο στην περίπτωση δυαδικών βαρών όσο και στην περίπτωση που τα βάρη είναι πραγματικοί αριθμοί.

2.3.3. Το Πιθανοτικό μοντέλο (Probabilistic model)

Το Πιθανοτικό μοντέλο πρωτοπαρουσιάστηκε από τους Robertson και Sparck Jones το 1976, αργότερα έγινε γνωστό και ως μοντέλο ανάκτησης δυαδικής ανεξαρτησίας (BIR). Η χρήση της θεωρίας πιθανοτήτων στην ανάκτηση πληροφορίας προτάθηκε για πρώτη φορά από τους Maron και Kuhns το 1960. Το κίνητρο για την εφαρμογή πιθανοτήτων ήταν το γεγονός ότι η διαδικασία της ανάκτησης χαρακτηρίζεται από ασάφεια. Σε αυτό το μοντέλο εισάγεται ένας τρόπος αναπαράστασης ο οποίος βασίζεται στην θεωρία πιθανοτήτων. Κατά συνέπεια το μοντέλο είναι πιθανοτικού χαρακτήρα και μεταχειρίζεται τη διαδικασία της ανάκτησης εγγράφων ως πιθανολογικό συμπέρασμα. Η κεντρική ιδέα είναι ότι το πρόβλημα ανάθεσης κάποιας μετρικής σχετικότητας μεταξύ ενός εγγράφου και ενός ερωτήματος μπορεί να γίνει ένα πρόβλημα εύρεσης της πιθανότητας το έγγραφο να είναι σχετικό με το ερώτημα. Δηλαδή προσπαθεί να ποσοτικοποιήσει την ασάφεια, υπολογίζοντας πόσο πιθανό είναι το γεγονός ένα έγγραφο να είναι σχετικό ως προς το ερώτημα που έχει τεθεί. Για την εκτίμηση της πιθανότητας χρησιμοποιούνται οι λέξεις που εμφανίζονται στο έγγραφο και στο ερώτημα αντίστοιχα.

Παραδοχές για τη λειτουργία του πιθανοτικού μοντέλου:

- Η πιθανότητα ένα έγγραφο να είναι σχετικό ως προς το ερώτημα θεωρείται ότι εξαρτάται μόνο από τους όρους που περιέχονται στο έγγραφο και από τους όρους που περιέχονται στο ερώτημα
- Η σχετικότητα ενός εγγράφου d ως προς το ερώτημα q δεν εξαρτάται από τη σχετικότητα άλλων εγγράφων της συλλογής.
- Για κάποιο ερώτημα q το σύνολο των σχετικών εγγράφων R είναι το ιδανικό σύνολο που μπορούμε να λάβουμε ως απάντηση.

Ο βαθμός ομοιότητας μεταξύ ενός εγγράφου d και ενός ερωτήματος q ορίζεται ως ο λόγος των πιθανοτήτων το έγγραφο να είναι σχετικό προς την πιθανότητα το έγγραφο να μην είναι σχετικό. Χρησιμοποιείται ο λόγος ώστε να αμβλύνει τα αποτελέσματα κάποιας λανθασμένης πρόβλεψης. Συμβολίζουμε με $Sim_{prob}(q,d)$ τη συνάρτηση που επιστρέφει την ομοιότητα μεταξύ d και q :

$$Sim_{prob}(d,q) = \frac{\pi\theta\alpha\nu\acute{o}\tau\eta\tau\alpha\ to\ d\ \nu\alpha\ \epsilon\acute{\iota}\nu\alpha\iota\ \sigma\chi\epsilon\tau\iota\kappa\acute{o}}{\pi\theta\alpha\nu\acute{o}\tau\eta\tau\alpha\ to\ d\ \nu\alpha\ \mu\eta\nu\ \epsilon\acute{\iota}\nu\alpha\iota\ \sigma\chi\epsilon\tau\iota\kappa\acute{o}}$$

Στο πιθανοτικό μοντέλο όλα τα βάρη των όρων του ευρετηρίου έχουν δυαδική μορφή, δηλαδή $w_{i,j} \in \{0,1\}$. Ορίζουμε ως R το σύνολο των εγγράφων τα οποία είναι σχετικά με το ερώτημα και ως συμπλήρωμά του, το$\bar{R}$, δηλαδή το σύνολο των μη σχετικών εγγράφων. Επιπλέον ορίζουμε ως $P(R|\vec{d}_j)$ την πιθανότητα το έγγραφο να είναι σχετικό προς το ερώτημα και ως $P(\bar{R}|\vec{d}_j)$ την πιθανότητα το έγγραφο να μην είναι σχετικό προς το ερώτημα. Η συνάρτηση ομοιότητας είναι η ακόλουθη :

$$Sim_{prob}(d_j, q) = \frac{P(R|\vec{d}_j)}{P(\overline{R}|\vec{d}_j)}$$

Από τον κανόνα το Bayes :

$$Sim_{prob}(d_j, q) = \frac{P(R|\vec{d}_j) * P(R)}{P(\overline{R}|\vec{d}_j) * P(\overline{R})}$$

Όπου $P(R|\vec{d}_j)$ ειναι η πιθανότητα το d_j να επιλέχθηκε τυχαία από το σύνολο R, δηλαδή να είναι σχετικό. Επιπλέον $P(R)$ είναι η πιθανότητα το έγγραφο που επιλέξαμε με τυχαίο τρόπο από ολόκληρη τη συλλογή να είναι τυχαίο. Οι ερμηνείες των ποσοτήτων $P(\overline{R}|\vec{d}_j)$ και $P(\overline{R})$ ειναι οι δυικές των παραπάνω.

Αφού οι πιθανότητες $P(\overline{R})$ και $P(R)$ ειναι ίσες, τότε :

$$Sim_{prob}(d_j, q) \approx \frac{P(R|\vec{d}_j)}{P(\overline{R}|\vec{d}_j)}$$

Λόγω της ανεξαρτησίας τον όρων, η παραπάνω σχέση γράφεται ως :

$$Sim_{prob}(d_j, q) \approx \frac{\left(\prod_{\vec{d}_{j=1}} P(R|k_i)\right) * \left(\prod_{\vec{d}_{j=0}} P(R|\overline{k}_i)\right)}{\left(\prod_{\vec{d}_{j=1}} P(\overline{R}|k_i)\right) * \left(\prod_{\vec{d}_{j=0}} P(\overline{R}|\overline{k}_i)\right)}$$

Όπου $P(R|k_i)$ ειναι η πιθανότητα ο όρος δεικτοδότησης k_i να εμφανίζεται σε ένα έγγραφο το οποίο επιλε΄χθηκε τυχαία από το σύνολο R. Ο όρος $P(R|\overline{k}_i)$ δίνει την πιθανότητα ο όρος k_i να μην εμφανίζεται σε ένα έγγραφο το οποίο επιλέχθηκε τυχαία απο το σύνολο R.

Λογαριθμίζοντας και λαμβάνοντας υπόψη ότι $P(R|k_i) + P(R|\overline{k}_i) = 1$ και αγνοώντας τους παράγοντες που ειναι σταθεροί για όλα τα έγγραφα για συγκεκριμένο ερώτημα, μπορούμε να γράψουμε :

$$Sim_{prob}(d_j, q) \approx \sum_{i=1}^{t} w_{i,q} * w_{i,j} * \left(\log\frac{P(R|k_i)}{1 - P(R|k_i)} + \log\frac{1 - P(\overline{R}|k_i)}{P(\overline{R}|k_i)}\right)$$

Ο οποίος είναι ουσιαστικά ο τύπος με τον οποίο υπολογίζουμε την κατάταξη των εγγράφων στο πιθανοτικό μοντέλο.

Επειδή δεν είναι γνωστό το σύνολο R εξ' αρχής, είναι απαραίτητο να οριστεί μία μέθοδος υπολογισμού για τις πιθανότητες $P(R|k_i)$ και $P(R|\overline{k}_i)$. Έπειτα από τη διατύπωση του ερωτήματος, δεν υπάρχουν άμεσα διαθέσιμα ανακτημένα έγγραφα. Γι' αυτό γίνονται απλοποιητικές υποθέσεις σε ότι αφορά τις πιθανότητες, η $P(R|k_i)$ ειναι σταθερή για όλους τους όρους k_i και ότι η κατανομή των όρων δεικτοδότησης στα μη σχετικά έγγραφα μπορεί να προσεγγιστεί από την κατανομή των όρων δεικτοδότησης στο σύνολο των εγγράφων (δηλαδή

το μέγεθος του συνόλου των μη σχετικών κειμένων $\bar{R}$ ειναι πολύ μεγαλύτερο από το μέγεθος R. Οι δύο υποθέσεις δίνουν :

$$P(R|k_i) = 0.5 \text{ (τυπικά)}$$

και

$$P(\bar{R}|k_i) = \frac{n_i}{N}$$

Ορίζεται ως n_i ο αριθμός των εγγράφων που περιέχουν τον όρο k_i και ως N ο συνολικός αριθμός των εγγράφων της συλλογής. Έχοντας την αρχική εκτίμηση, γίνεται η ανάκτηση ενός αρχικού συνόλου εγγράφων που περιέχουν όρους που εμφανίζονται στο ερώτημα και παρέχεται μια πιθανοτική κατάταξη γι' αυτά.

Η βελτίωση των $P(R|k_i)$ και $P(\bar{R}|k_i)$ οδηγεί στη βελτίωση της πιθανοτικής κατάταξης, για να επιτευχθεί αυτό, προσεγγίζονται η $P(R|k_i)$ με την κατανομή του όρου k_i στα έγγραφα που ανακτώνται αρχικά και η $P(\bar{R}|k_i)$ θεωρώντας όλα τα μην ανεκτημένα έγγραφα ως μη σχετικά. Έστω V ένα υποσύνολο των εγγράφων που ανακτήθηκαν αρχικά και στα οποία δόθηκε μία κατάταξη από το πιθανοτικό μοντέλο. Για παράδειγμα το παραπάνω σύνολο θα μπορούσε να είναι, τα κορυφαία r έγγραφα, όπου το r είναι ένα προκαθορισμένο κατώφλι. Έστω V_i ένα υποσύνολο του V το οποίο αποτελείται από τα έγγραφα που περιέχουν τον όρο k_i.

$$P(R|k_i) = \frac{\mathrm{V}_i}{\mathrm{V}}$$

και

$$P(\bar{R}|k_i) = \frac{n_i - \mathrm{V}_i}{N - \mathrm{V}}$$

Η διαδικασία για τη βελτίωση των εκτιμήσεων μπορεί να επαναληφθεί αναδρομικά, υπολογίζοντας κάθε φορά νέα V και V_i. Οι παραπάνω τύποι εμφανίζουν προβλήματα για μικρές τιμές των V και V_i που εμφανίζονται στην πράξη, για παράδειγμα V=1 και $\mathrm{V}_i = 0$. Αυτά τα προβλήματα αντιμετωπίζονται εισάγοντας ένα σταθερό προσθετικό παράγοντα ή θεωρώντας ως προσθετικό παράγοντα την ποσότητα n_i/N :

$$P(R|k_i) = \frac{\mathrm{V}_i + \frac{n_i}{N}}{\mathrm{V} + 1}$$

και

$$P(\bar{R}|k_i) = \frac{n_i - \mathrm{V}_i + \frac{n_i}{N}}{N - \mathrm{V} + 1}$$

Το κύριο πλεονέκτημα του πιθανοτικού μοντέλου είναι ότι τα έγγραφα κατατάσσονται σε φθίνουσα σειρά με βάση την πιθανότητα να είναι σχετικά με το αρχικό ερώτημα. Τα μειονεκτήματα είναι :

- Χρειάζεται μια αρχική εκτίμηση για το διαχωρισμό της συλλογής των εγγράφων σε σχετικά και μη.
- Το γεγονός ότι δεν λαμβάνεται υπόψη η συχνότητα εμφάνισης του όρου μέσα σ' ένα έγγραφο (όλα τα βάρη είναι 0 ή 1).
- Η υπόθεση της ανεξαρτησίας των όρων.
- Υπολογιστικό κόστος

2.3.3.1. Μέθοδος OKAPI BM25

Μία παραλλαγή του βασικού πιθανοτικού μοντέλου είναι το OKAPI BM25. Έλαβε το όνομα της από το σύστημα στο οποίο εφαρμόστηκε για πρώτη φορά, το OKAPI. Το OKAPI είναι ένα σύστημα ανάκτησης πληροφορίας που αναπτύχθηκε στον Πανεπιστήμιο City του Λονδίνου από τον Robertson και τους συνεργάτες του και έχει χρησιμοποιηθεί ως πλατφόρμα για τη μελέτη της αποτελεσματικότητας διαφόρων μεθόδων προσδιορισμού της ομοιότητας. Το BM προκύπτει από τις λέξεις Best Match. Αποτελεί μία από τις πιο αποτελεσματικές μεθόδους υπολογισμού της ομοιότητας, η οποία έδωσε πολύ καλά αποτελέσματα στις συλλογές εγγράφων TREC, πρόκειται για συνδυασμό των συναρτήσεων ομοιότητας BM11 και BM15. Αυτή η μέθοδος λαμβάνει υπόψη τις συχνότητες των όρων στα έγγραφα και το μήκος αυτών και πειραματικά πετυχαίνει καλά αποτελέσματα. Αυτή η μέθοδος στηρίζεται στη χρήση δύο κατανομών Poisson. Σύμφωνα με αυτήν, ένα έγγραφο θεωρείται ως μία τυχαία ροή εμφανίσεων όρων. Κάθε εμφάνιση ενός όρου t θεωρείται ένα γεγονός που αφορά τον όρο t. Θεωρούμε ότι ο επόμενος όρος του εγγράφου θα είναι ο όρος t με πιθανότητα p, συνεπώς 1-p είναι η πιθανότητα ο επόμενος όρος του εγγράφου να είναι διαφορετικός του t. Υποθέτουμε ότι η πιθανότητα να εμφανίζεται ο όρος t σε ένα έγγραφο και η συχνότητα εμφάνισής του ακολουθούν την κατανομή Poisson. Τα έγγραφα που περιέχουν τον όρο t διαχωρίζονται σε αυτά τα οποία η θεματολογία τους αναφέρεται στο θέμα στο οποίο αναφέρεται και ο t, και σε αυτά τα οποία δεν αναφέρονται στο θέμα του t. Η συνάρτηση που υπολογίζει το βαθμό ομοιότητας μεταξύ ενός ερωτήματος q και ενός εγγράφου της συλλογής d είναι η παρακάτω :

$$S_{BM25}(d,q) = \sum log\frac{(r_i + 0.5) * (N - R - n_i - r_i + 0.5)}{(n_i - r_i + 0.5) * (R - r_i + 0.5)} * \frac{(C_1 + 1) * f_{t_i,d}}{C_4 + f_{t_i,d}} * \frac{(C_3 + 1) * f_{t_i,d}}{C_3 + f_{t_i,d}} + C_2 * |q| * \frac{avdl - dl_d}{avdl + dl_d}$$

Το N είναι ο συνολικός αριθμός εγγράφων στη συλλογή, το R είναι το πλήθος των σχετικών ως προς το ερώτημα εγγράφων της συλλογής, το r_i είναι το πλήθος των σχετικών εγγράφων που περιέχουν τον όρο t και n_i είναι το πλήθος των συνολικών εγγράφων που περιέχουν τον όρο t.

Το $f_{t_i,d}$ είναι η συχνότητα του όρου στο έγγραφο, οι C_1, C_2, C_3, C_4 είναι σταθερές ρύθμισης, dl_d είναι το μήκος του εγγράφου d και $avdl$ είναι το μέσο μήκος των εγγράφων της συλλογής.

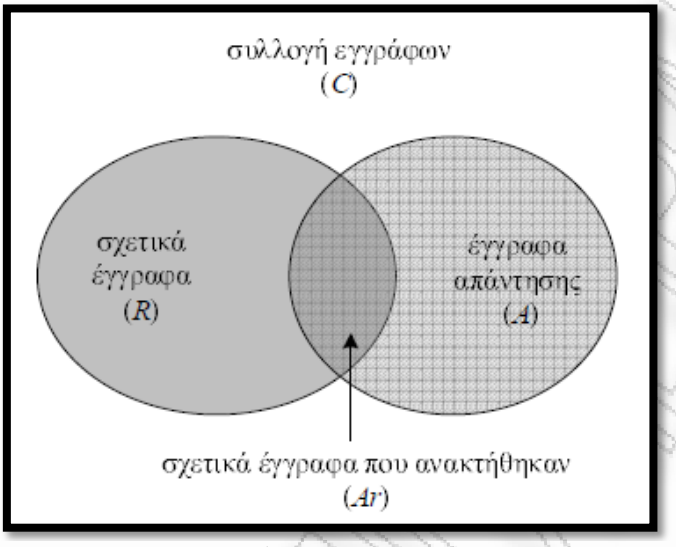
Η συνάρτηση ομοιότητας BM25 έχει πολλές ρυθμιστικές σταθερές και υπάρχουν και άλλες προσεγγίσεις, στην πραγματικότητα υπάρχει μια ολόκληρη οικογένεια συναρτήσεων ομοιότητας, με ελαφρώς διαφορετικές συνιστώσες και οι παραμέτρους.

Η παραπάνω προσέγγιση καλείται και OKAPI TF, αυτός ο τύπος σχεδιάστηκε για να χρησιμοποιηθεί με το OKAPI πιθανοτικό μοντέλο, αλλά έχει αποδειχθεί ότι όταν χρησιμοποιείται με το διανυσματικό μοντέλο δίνει καλύτερα αποτελέσματα ανάκτησης.

2.3.4. Μετρικές Απόδοσης

Η αποτελεσματικότητα κάποιου συστήματος IR, δηλαδή η ποιότητα των αποτελεσμάτων του, μπορεί να υπολογιστεί χρησιμοποιώντας τις βασικές μετρικές αποτίμησης της απόδοσής του, που είναι η ανάκληση (recall) και η ακρίβεια (precision). Η αποτελεσματικότητα του συστήματος εξαρτάται από πολλούς παράγοντες όπως: τη συλλογή εγγράφων που διαχειρίζεται το σύστημα, τα ερωτήματα των χρηστών και το μοντέλο ανάκτησης που χρησιμοποιείται.

Έστω ότι έχουμε τη δυνατότητα να διαπιστώσουμε εάν ένα έγγραφο d που ανήκει στη συλλογή εγγράφων C είναι ή όχι σχετικό ως προς ένα ερώτημα q. Θεωρούμε ότι μετά την επεξεργασία του ερωτήματος q, το σύστημα επέστρεψε κάποια έγγραφα που αποτελούν το σύνολο απάντησης A. Έστω επίσης ότι το σύνολο R περιέχει όλα τα σχετικά ως προς το ερώτημα έγγραφα της συλλογής C. Τέλος με A_r συμβολίζουμε το υποσύνολο A που περιέχει τα σχετικά ως προς το q έγγραφα. Προφανώς το σύνολο A_r είναι η τομή των συνόλων A και R. Στο παρακάτω διάγραμμα Venn φαίνονται τα σύνολα εγγράφων, ενώ στο σχήμα 4 δίνονται οι τέσσερις δυνατοί χαρακτηρισμοί των εγγράφων ως προς ένα ερώτημα. Στην ιδανική περίπτωση τα έγγραφα θα ανήκουν στο είτε στο κάτω αριστερό είτε στο επάνω δεξί τεταρτημόριο, δηλαδή όσα έγγραφα είναι σχετικά ως προς το ερώτημα έχουν ανακτηθεί, ενώ στη δεύτερη περίπτωση δεν θα έχει ανακτηθεί κάποιο μη σχετικό έγγραφο.



Διάγραμμα 2 :
Απεικόνιση των εγγράφων που έχουν ανακτηθεί και των εγγράφων που θα έπρεπε να είχαν ανακτηθεί

	ανακτηθέν	μη ανακτηθέν
μη σχετικό	Μη σχετικά έγγραφα που έχουν ανακτηθεί (*A-Ar*)	Μη σχετικά έγγραφα που δεν έχουν ανακτηθεί (*C* - (*R* U *A*))
σχετικό	Σχετικά έγγραφα που έχουν ανακτηθεί (*Ar*)	Σχετικά έγγραφα που δεν έχουν ανακτηθεί (*R-A*)

Σχήμα 4 : Κατηγοριοποίηση εγγράφων

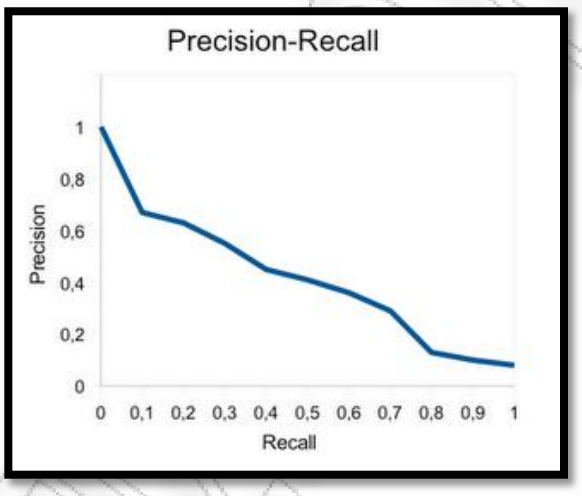
Στο πεδίο της ανάκτησης πληροφοριών η ανάκληση (recall) ορίζεται ως ο αριθμός των σχετικών εγγραφών που ανακτώνται από μια αναζήτηση προς το συνολικό αριθμό των σχετικών εγγραφών της συλλογής.

$$\text{Recall} = \frac{\text{αριθμός σχετικών εγγράφων που ανακτήθηκαν}}{\text{αριθμός σχετικών εγγράφων στη συλλογή}} = \frac{A_r}{R}$$

Η ανάκληση μετρά στην ουσία το ποσοστό των σχετικών εγγράφων που το σύστημα μπόρεσε να ανακτήσει σε σχέση με όλα τα σχετικά έγγραφα που υπάρχουν στη συλλογή. Κάτι που δεν λαμβάνει υπόψη η ανάκληση είναι το πλήθος των εγγράφων που ανακτήθηκαν αλλά δεν είναι σχετικά ως προς το ερώτημα. Πολλές φορές εμφανίζεται φαινόμενο να έχουν αποκτηθεί όλα τα σχετικά έγγραφα αλλά μαζί με αυτά να έχουν αποκτηθεί πολλά μη σχετικά ως προς το ερώτημα.

Η ακρίβεια (precision) ορίζεται ως ο αριθμός των σχετικών εγγράφων που ανακτώνται από μια αναζήτηση προς το συνολικό αριθμό των σχετικών εγγράφων που ανακτήθηκαν.

$$\text{Precision} = \frac{\text{αριθμός σχετικών εγγράφων που ανακτήθηκαν}}{\text{αριθμός εγγράφων που ανακτήθηκαν}} = \frac{A_r}{A}$$



Σχήμα 5: Ακρίβεια - Ανάκληση

Η συνηθισμένη απόκριση που έχει ένα σύστημα IR είναι αυτή που παρουσιάζεται στο παραπάνω σχήμα (5), στην οποία φαίνεται ότι τα μεγέθη της ακρίβειας και ανάκλησης είναι αντιστρόφως ανάλογα. Αυτό σημαίνει πως για αν αυξηθεί η ανάκληση θα πρέπει να μειωθεί η ακρίβεια, φυσικά ισχύει και το αντίστροφο.

Τα διαγράμματα ακρίβειας/ανάκλησης θεωρούνται ως μια από τις κλασσικές μεθόδους εκτίμησης της απόδοσης ανάκλησης ενός συστήματος IR και χρησιμοποιούνται εκτεταμένα στη βιβλιογραφία. Επίσης, είναι χρήσιμα επειδή επιτρέπουν την ποσοτική εκτίμηση τόσο της ποιότητας του ανακτώμενου συνόλου εγγράφων όσο και του εύρους του αλγορίθμου ανάκτησης. Επιπλέον, είναι απλά στην κατανόηση και μπορούν να συνοψιστούν εύκολα με τη χρήση μιας απλής αριθμητικής τιμής

2.4. Γενικά για τη βιβλιοθήκη Lucene

Η Lucene είναι μία open-source βιβλιοθήκη υλοποιημένη στην γλώσσα προγραμματισμού Java. Είναι μια πλήρης, υψηλής απόδοσης επεκτάσιμη βιβλιοθήκη ανάκτησης πληροφοριών. Πρόκειται για την αρχική βιβλιοθήκη αναζήτησης και ευρετηρίασης που δημιουργήθηκε από τον Doug Cutting. Από τότε η Lucene επιλέχθηκε ως ένα κορυφαίο λογισμικό από το Apache Software Foundation. Η Lucene έχει γίνει εξαιρετικά δημοφιλής και ίσως αποτελεί την πιο χρησιμοποιημένη βιβλιοθήκη ανάκτησης πληροφοριών, αφού η ενσωμάτωση της λειτουργικότητάς της σε μια διαδικτυακή ή desktop εφαρμογή είναι σχετικά εύκολη. Η απλότητά της είναι ένας από τους βασικούς παράγοντες που οδήγησαν στην αύξηση της δημοτικότητά της.

Η Lucene αποτελεί ένα cross-platform καλά σχεδιασμένο λογισμικό και παρέχει έναν απλό, όμως ισχυρό πυρήνα API που απαιτεί την ελάχιστη κατανόηση των τεχνικών ευρετηριοποίησης και αναζήτησης κειμένου. Η ευρεία χρήση της, οδήγησε στην εξέλιξη και ενσωμάτωση και άλλων βιβλιοθηκών ώστε να εξασφαλιστεί η διαλειτουργικότητα μεταξύ τους. Αυτές, προσφέρουν πρόσθετη λειτουργικότητα και αυξάνουν το εύρος εφαρμογής της Lucene. Για τις βοηθητικές - υποστηρικτικές βιβλιοθήκες που ενσωματώνονται στη Lucene και χρησιμοποιούνται στην παρούσα εργασία θα γίνει εκτενής αναφορά σε επόμενα κεφάλαια, για την εύρεση του πλήρη καταλόγου τους ο αναγνώστης παραπέμπεται στο http://lucene.apache.org/ .

2.5. Εξέλιξη της Lucene

Η Lucene ξεκίνησε ως ένα αυτό-εξυπηρετούμενο έργο από τον Doug Cutting το 1999, η πρώτη έκδοση της δημοσιεύθηκε νωρίτερα από το 2001, έτος στο οποίο συγκαταλέχθηκε στο λογισμικό που αναπτύσσει και διανέμει το Apache Software Foundation. Το 2005 κατέστει ως ένα από τα κορυφαία project του οργανισμού. Από τότε έχουν γίνει πολλές αλλαγές και έχουν συμπεριληφθεί σε αυτήν νέες λειτουργίες όπως η αναζήτηση σε σχεδόν πραγματικό χρόνο, η αναζήτηση για αριθμητικά πεδία, εφαρμογές για τη χρησιμοποίηση των payloads, εφαρμογές συναλλαγών για ευρετηριοποίηση και αναζήτηση κτλ.

Μέχρι πρόσφατα, περιελάμβανε μια σειρά από επιμέρους έργα, όπως τα Lucene Java, Droids, Lucene.Net, Lucy, Mahout, Solr, Nutch, Open Relevance Project, PyLucene και Tika. Το Solr συγχωνεύθηκε με το Lucene και τα Mahout, Nutch Tika έγιναν ανεξάρτητα υψηλού επιπέδου project. Η ευρεία χρήση και εφαρμογή της οδήγησε στην ανάπτυξη μεθόδων ώστε να μπορεί να χρησιμοποιηθεί και από άλλες γλώσσες προγραμματισμού όπως η C++, C#, Perl και Python. Η τελευταία διαθέσιμη έκδοσή της είναι η 3.5.0 και δημοσιεύθηκε το Νοέμβριο του 2011.

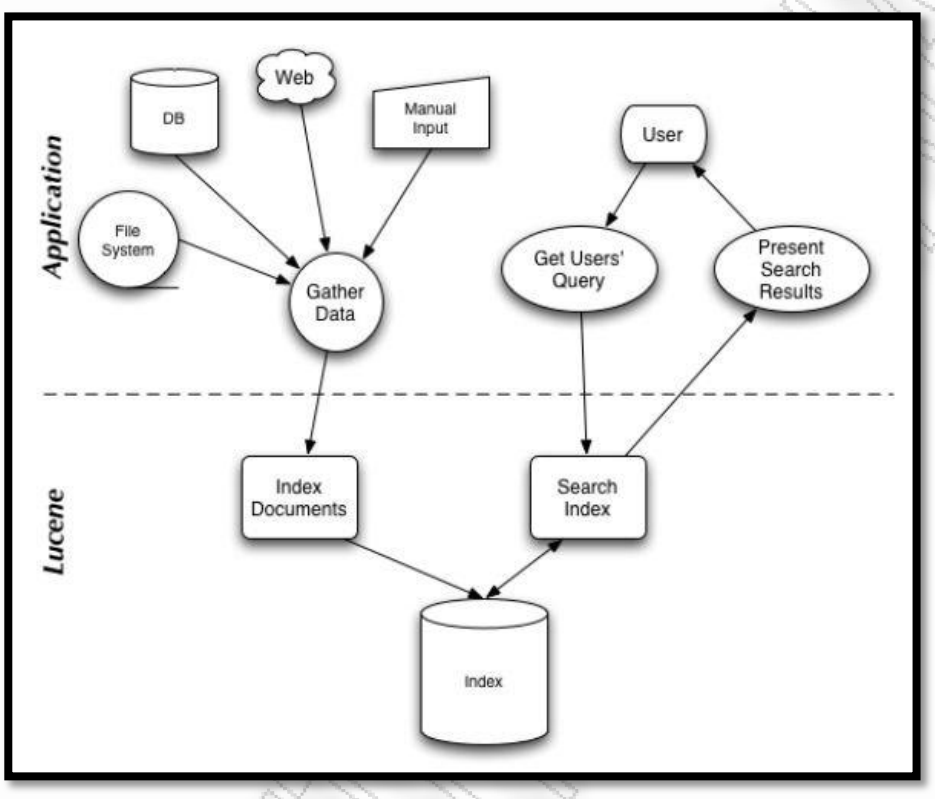
Το εύρος εφαρμογής και η σημαντικότητα του εργαλείου αντικατοπτρίζεται από τη χρησιμοποίησή της από διαπρεπής εταιρείες του κλάδου της πληροφορικής αλλά και γενικότερα των επιχειρήσεων. Ενδεικτικά αναφέρονται η Apple, η IBM τα κοινωνικά δίκτυα Twitter και LinkedIn, η Disney,η AOL κ.ά

2.6. Λειτουργίες της βιβλιοθήκης Lucene

Μια κοινή παρερμηνεία αποτελεί ότι η βιβλιοθήκη Lucene είναι μια ολοκληρωμένη εφαρμογή αναζήτησης, ενώ στην πραγματικότητα είναι ο πυρήνας στον οποίο στηρίζεται η μηχανή αναζήτησης για να επιτελέσει τις λειτουργίες της.

Η βιβλιοθήκη Lucene παρέχει συναρτήσεις για διάσχιση του σκληρού δίσκου, εξαγωγή κειμένου από διάφορους τύπους εγγράφων, ανάλυση του κειμένου και αποθήκευση της εξαγόμενης πληροφορίας σε ευρετήρια και αναζήτηση λέξεων κλειδιών σε αυτά. Στον πυρήνα της λογικής αρχιτεκτονικής της Lucene είναι η ιδέα ενός εγγράφου που περιέχει πεδία κειμένου. Αυτή η ευελιξία επιτρέπει στη Lucene να είναι ανεξάρτητη από τη μορφή αρχείου, δηλαδή, παρέχει τη δυνατότητα άντλησης κειμένου ανεξάρτητα από τη μορφή των αρχείων στο οποίο αυτό περιέχεται, της γλώσσας στην οποία είναι γραμμένο, το σχηματισμό του και την πηγή από την οποία προέρχεται.

Επεκτείνοντας τα παραπάνω, παρέχεται η δυνατότητα ευρετηρίασης και αναζήτησης σε αποθηκευμένα δεδομένα αποθηκευμένα δεδομένα στα αρχεία, ιστοσελίδες σε μακρινούς κεντρικούς υπολογιστές δικτύου, έγγραφα που αποθηκεύονται στο τοπικό file-system, απλά αρχεία κειμένων, έγγραφα Microsoft Word, αρχεία XML, HTML, PDF ή οποιαδήποτε άλλα αρχεία από τα οποία, μπορεί να εξαχθεί πληροφορία με τη μορφή κειμένου. Ομοίως, χρησιμοποιώντας τη Lucene είναι δυνατή η ευρετηριοποίηση αποθηκευμένων δεδομένων στις βάσεις δεδομένων, παρέχοντας τη δυνατότητα αναζήτησης ολοκληρωμένου κειμένου σε αυτές, αφού πολλές βάσεις δεδομένων την υποστηρίζουν περιορισμένα. Στη εικόνα που ακολουθεί παρουσιάζεται ο τρόπος με τον οποίο χρησιμοποιείται η βιβλιοθήκη από τις εφαρμογές αναζήτησης.



Εικόνα 5: Ενσωμάτωση της Lucene σε εφαρμογή αναζήτησης,πηγή http://blog.csdn.net/

Αναφέρουμε ως παράδειγμα τη λειτουργία που επιτελεί η Lucene στη σελίδα κοινωνικής δικτύωσης Twitter, κάθε καταχώρηση (tweet) από τους χρήστες αναπαριστάται από ένα έγγραφο το οποίο έχει ως πεδία (fields) τις πληροφορίες του tweet που μπορεί να ενδιαφέρουν (ημερομηνία υποβολής, κείμενο, αναγνωριστικό κ.ά.). Τα δεδομένα κάθε εγγράφου πριν αποθηκευθούν στο ευρετήριο υπόκεινται σε κάποια προεπεξεργασία για τη διαδικασία αυτή χρησιμοποιούνται οι κλάσεις της βιβλιοθήκης που περιγράφονται παρακάτω.

2.7. Δημιουργία ευρετηρίου με τη βιβλιοθήκη Lucene

Πυρήνας όλων των μηχανών αναζήτησης είναι η δημιουργία του ευρετηρίου, σε αυτό βασίζονται ώστε να επεξεργαστούν τα δεδομένα προκειμένου να είναι αποδοτική και γρήγορη η αναζήτησή τους. Η δημιουργία ευρετηρίου με τη Lucene προσφέρει υψηλή επεκτασιμότητα, cross - platform υποστήριξη, έχει μικρές απαιτήσεις σε μνήμη και παρέχει τη δυνατότητα αναζήτησης με λέξεις κλειδιά ορισμένες από το χρήστη. Οι περισσότερες μηχανές αναζήτησης χρησιμοποιούν B-δένδρα για τη δημιουργία ευρετηρίου το οποίο αυξάνει την πολυπλοκότητα, αντί αυτού, στη Lucene υιοθετείται μια ελαφρώς διαφορετική προσέγγιση: αντί να διατηρείται μόνο ένα ευρετήριο, δημιουργούνται πολλαπλά τμήματα αυτού (index segments) και συγχωνεύονται περιοδικά. Για κάθε νέο έγγραφο που ευρετηριοποιείται, δημιουργείται ένα νέο

index segment, ενώ τα μικρά τμήματα συγχωνεύονται με μεγαλύτερα κρατώντας τον αριθμό των index segments μικρό, με αυτήν την τεχνική επιτυγχάνεται γρηγορότερη αναζήτηση.

Η μετατροπή ενός κειμένου σε μορφή ευρετηρίου όπως αναφέρθηκε παραπάνω, περιλαμβάνει τη χρήση :

- Συντακτικού ή λεκτικού αναλυτή (parser).
- Λίστα κοινών λέξεων, όπως άρθρα, προθέσεις, κ.ά. (stop words).
- Διάφορα επιπλέον φίλτρα πληροφορίας.

Τα βασικά βήματα που λαμβάνουν χώρα κατά την ευρετηριοποίηση με τη Lucene είναι :

- Διαχωρισμός λεκτικών μονάδων (tokenization).
- Διύλιση πληροφορίας (filtration).
- Μετατροπή κάθε εμφανιζόμενης λέξης στη ρίζα της (stemming).

Τα παραπάνω υλοποιούνται χρησιμοποιώντας τις παρακάτω κλάσεις της Lucene :

- IndexWriter
- Directory
- Analyzer
- Document
- Field

2.7.1. Κλάση IndexWriter

Η κλάση IndexWriter αποτελεί την κεντρική συνιστώσα δημιουργίας του ευρετηρίου, χρησιμοποιείται για τη δημιουργία ενός ευρετηρίου και την πρόσθεση νέων εγγράφων σε αυτό.

2.7.2. Κλάση Directory

Η κλάση Directory αναπαριστά την τοποθεσία στην οποία είναι αποθηκευμένο το ευρετήριο της βιβλιοθήκης.

2.7.3. Κλάση Analyzer

Η κλάση Analyzer χρησιμοποιείται για να φιλτράρει τα δεδομένα που ευρετηριοποιούνται, η Lucene περιέχει αρκετές υλοποιήσεις αυτής, η πιο σημαντική από αυτές είναι εκείνη (Porter stemmer) που ενσωματώνει τον αλγόριθμο του Porter για την εύρεση των λεκτικών ριζών μιας λέξης. Θα ακολουθήσει αναλυτικότερη περιγραφή στα επόμενα κεφάλαια, για τις επεκτάσεις της κλάσης που χρησιμοποιήθηκαν στη συγκεκριμένη εφαρμογή. Συνοπτικά, αυτή η κλάση και

οι επεκτάσεις της χρησιμοποιούνται ώστε γίνει λεξικογραφική ανάλυση του κειμένου για τη δημιουργία των ελάχιστων λεκτικών μονάδων (tokens), με μετατροπή των κεφαλαίων γραμμάτων σε πεζά, αφαίρεση τόνων, διαγραφή των κοινών λέξεων (stop words) και τέλος στελέχωση, δηλαδή αφαίρεση των καταλήξεων των λέξεων ώστε να παραχθεί η ρίζα τους (stemming).

2.7.4. Κλάση Document

Ένα αντικείμενο της κλάσης Document αναπαριστά ένα σύνολο από πεδία (fields), ένα έγγραφο (Document) μπορεί να είναι ένα σύνολο από δεδομένα, μια ιστοσελίδα, ένα μήνυμα ηλεκτρονικού ταχυδρομείου ή ένα αρχείο το οποίο θα προστεθεί στο ευρετήριο ώστε να είναι δυνατή η ανακτησή του. Ένα ευρετήριο αποτελείται από ένα σύνολο εγγράφων και κάθε έγγραφο αποτελείται από ένα ή περισσότερα πεδία (fields). Συνεπώς είναι αναγκαία η υλοποίηση μιας δομής με την οποία θα αποθηκεύονται τα δεδομένα στο ευρετήριο. Η συγκεκριμένη δομή εκτός από πληροφορίες για λέξεις και κλειδιά, μπορεί να περιέχει και πληροφορίες όπως το όνομα, η ημερομηνία τροποποίησης του κάθε αρχείου αλλά και τη συχνότητα εμφάνισης κάποιου όρου που περιέχει.

2.7.5. Κλάση Field

Κάθε αντικείμενο της κλάσης Document περιέχει ένα ή περισσότερα πεδία τα οποία αναπαριστώνται από την κλάση Field. Κάθε πεδίο αντιστοιχεί στα δεδομένα που χρησιμοποιούνται κατά την αναζήτηση ως λέξεις κλειδιά ή στα δεδομένα που ανακτώνται από το ευρετήριο κατά την αναζήτηση .

2.8. Αναζήτηση με τη βιβλιοθήκη Lucene

Στην ενότητα αυτή περιγράφονται οι βασικότερες κλάσεις της Lucene που χρησιμοποιούνται για την ανάκτηση δεδομένων από το ευρετήριο, αυτές είναι οι ακόλουθες :

- IndexSearcher
- Query Parser
- Query
- Hits

2.8.1. Κλάση IndexSearcher

Η κλάση αυτή αποτελεί στην αναζήτηση ότι η κλάση IndexWriter κατά την ευρετηριοποίηση. Προσπελαύνει ένα ευρετήριο για διάβασμα μόνο (read-only mode) και προσφέρει ένα σύνολο από μεθόδους αναζήτησης.

2.8.2. Κλάση Query Parser

Η κλάση αύτη παράγεται από τον JavaCC, ο οποίος είναι ένας parser generator και lexical analyzer για την γλώσσα Java. Παρότι η Lucene παρέχει τη δυνατότητα στο χρήστη να διατυπώσει μέσω του API της ερωτήματα, παρέχει επιπλέον μία πλούσια γλώσσα ερωτημάτων μέσω αυτής της κλάσης. Η σημαντικότερη συνάρτηση που χρησιμοποιεί είναι η parse(String), χρησιμοποιείται για τη διάπσαση των ερωτημάτων σε όρους, προτάσεις και τελεστές. Ως απλό όρο ορίζει για παράδειγμα τη λέξη "Hello", ενώ ως πρόταση ορίζει λέξεις ανάμεσα σε "", όπως για παράδειγμα "Hello Max". Επίσης μπορεί να δεθχεί σε ένα ερώτημα πεδία, δηλαδή ο χρήστης μπορεί να αναζητήσει κάτι βάσει πεδίου χρησιμοποιώντας το όνομα του πεδίου και άνω κάτω τελεία, ένα παράδειγμα αποτελεί η αναζήτηση βάσει τίτλου, title:"The Right Way" AND text:go.

2.8.3. Κλάση Query

Η κλάση Query περιέχει τα κριτήρια αναζήτησης που δημιουργούνται από τον Query Parser, στην βιβλιοθήκη περιέχονται αρκετές υλοποιήσεις της κλάσης αυτής, αναφέρονται ενδεικτικά οι BooleanQuery, TermQuery, RangeQuery, κ.ά.

2.8.4. Κλάση Hits

Η αναζήτηση χρησιμοποιώντας τη Lucene επιστρέφει ένα αντικείμενο της κλάσης Hits, αυτό το αντικείμενο περιέχει δείκτες προς εκείνα τα έγγραφα (documents) τα οποία ικανοποιούν τα κριτήρια αναζήτησης.

2.9. Βαθμολόγηση με τη Lucene

Η μέθοδος βαθμολόγησης της Lucene είναι ένας από τους λόγους που είναι εξαιρετικά δημοφιλής. Πετυχαίνει υψηλή ταχύτητα ενώ ταυτόχρονα κρύβει σχεδόν όλη την πολυπλοκότητα από το χρήστη. Για τον καθορισμό της ομοιότητας ενός εγγράφου με το ερώτημα που τίθεται από το χρήστη χρησιμοποιείται ένας συνδυασμός δύο βασικών μοντέλων που χρησιμοποιούνται στην επιστήμη της ανάκτησης πληροφορίας, το διανυσματικό μοντέλο (Vector Space Model) και το δυαδικό (Boolean). Πρώτα χρησιμοποιείται το Boolean μοντέλο με στόχο τον περιορισμό των εγγράφων που πρέπει να βαθμολογηθούν, έχοντας ως βάση τη Boolean λογική, έπειτα χρησιμοποιείται το διανυσματικό μοντέλο για τον υπολογισμό του βαθμού ομοιότητας. Σύμφωνα με αυτό, όσες περισσότερες φορές εμφανίζεται ένας όρος ερωτήματος σε ένα έγγραφο σε σχέση με τον αριθμό των φορών που εμφανίζεται σε όλα τα έγγραφα στη συλλογή, τόσο πιο σχετικό είναι το έγγραφο αυτό με το ερώτημα.

Με το διανυσματικό μοντέλο τα έγγραφα και τα ερωτήματα αναπαρίστανται ως σταθμισμένα διανύσματα σε ένα πολυδιάστατο χώρο, όπου κάθε όρος (term) του ευρετηρίου είναι μία διάσταση και τα βάρη είναι οι τιμές που έχουν προκύψει από τη συνάρτηση tf-idf (term frequency-inverse document frequency), έγινε αναλυτική αναφορά στην ενότητα 2.1.3.2.1. Πάνω σε αυτή την συνάρτηση βασίζεται και η συνάρτηση του Lucene, το διανυσματικό μοντέλο δεν απαιτεί να έχουν προκύψει οι τιμές από την tf-idf, αλλά η tf-idf έχει αποδειχθεί ότι παράγει αποτελέσματα αναζήτησης υψηλής ποιότητας και γι' αυτό χρησιμοποιούνται. Τα διανύσματα που αναφέρονται παραπάνω λαμβάνουν μη μηδενικές τιμές για τις λέξεις κλειδιά που εμφανίζονται στο έγγραφο και οι τιμές τους εξαρτώνται από τη συχνότητα εμφάνισης των λέξεων κλειδιών - όρων και τη σημαντικότητα τους μέσα στο αρχείο.

Η βαθμολογία για ένα έγγραφο d υπολογίζεται ταιριάζοντας κάθε t στο ερώτημα q, και προκύπτει από το συνημίτονο της εμπεριεχομένης γωνίας (cosine similarity) των διανυσμάτων. Ένα έγγραφο του οποίο το διάνυσμα βρίσκεται πιο κοντά στο διάνυσμα του ερωτήματος στο μοντέλο αυτό, βαθμολογείται υψηλότερα, δηλαδή παρουσιάζει καλύτερη αντιστοιχία στην ερώτηση, ο βαθμός ομοιότητας του Lucene υπολογίζεται ως εξής:

$$\cos(\theta) = \frac{\vec{V}(q) * \vec{V}(d)}{|\vec{V}(q)| * |\vec{V}(d)|}$$

Όπου αριθμητής είναι το εσωτερικό γινόμενο των διανυσμάτων $\vec{V}(q)$ και $\vec{V}(d)$ και παρανομαστής είναι το γινόμενο των ευκλείδιων αποστάσεών τους. Αυτό το αποτέλεσμα είναι το ακατέργαστο αποτέλεσμα, το οποίο είναι ένας αριθμός κινητής υποδιαστολής μεγαλύτερος ή ίσος του μηδενός.

2.9.1. Βελτίωση αποτελεσμάτων

Η Lucene παρέχει τη δυνατότητα για βελτίωση της βαθμολογίας που προκύπτει από το διανυσματικό μοντέλο τόσο για την ποιότητα των αποτελεσμάτων όσο και για τη χρηστικότητα της αναζήτησης. Προς αυτήν την κατεύθυνση χρησιμοποιούνται κάποιες τεχνικές ώθησης (boost).

Η βαθμολόγηση εξαρτάται σε μεγάλο βαθμό από τον τρόπο που τα έγγραφα έχουν ευρετηριαστεί, συνεπώς η ευρετηρίαση διαδραματίζει σημαντικό ρόλο στον υπολογισμό των αποτελεσμάτων. Κατά την ευρετηρίαση, οι χρήστες μπορούν να ορίσουν ότι κάποια έγγραφα είναι περισσότερο σημαντικά από άλλα, αναθέτοντας μια ώθηση στο έγγραφο (document boost). Αυτό αποτυπώνεται, με τον πολλαπλασιασμό της βαθμολογίας κάθε εγγράφου με την τιμή της ώθησης doc-boost(d).

Κατά την αναζήτηση ο χρήστης μπορεί να ορίσει ωθήσει – τονώσει κάθε όρο του ερωτήματος, οι όροι ενός ερωτήματος επηρεάζουν τη βαθμολογία του εγγράφου

πολλαπλασιάζοντας την τιμή της ώθησης που έχουν λάβει με αυτή, query-boost(q). Οι τιμές για την αυτήν την ώθηση εισάγονται πριν την αναζήτηση.

Ένα έγγραφο μπορεί να ταιριάξει με ένα ερώτημα δίχως να περιλαμβάνει όλους τους όρους του ερωτήματος, η βιβλιοθήκη δίνει τη δυνατότητα στους χρήστες να προάγουν τα έγγραφα που περιέχουν και ταιριάζουν τους περισσότερους όρους τους ερωτήματος που έθεσαν, μέσω ενός παράγοντα συντονισμού coord-factor(q,d).

Για να είναι συγκρίσιμοι οι βαθμοί ομοιότητας των εγγράφων για δύο διαφορετικά ερωτήματα και για την χρησιμοποίηση των αποτελεσμάτων από άλλες εφαρμογές γίνεται κανονικοποίηση στη μονάδα, του διανύσματος του ερωτήματος. Κανονικοποιώντας τα αποτελέσματα στη μονάδα, αποφεύγεται η απώλεια δεδομένων λόγω περιορισμών στην ακρίβεια της στρογγυλοποίησης.

Η συνάρτηση ομοιότητας που προκύπτει ενσωματώνοντας τους παράγοντες ώθησης ώστε να αποτυπώνονται οι αλληλεπιδράσεις, είναι η παρακάτω :

$$\text{score}(q,d) = coord(q,d) * queryNorm(q) * \sum_{t\ in\ q}\left(tf(t\ in\ d) * idf(t)^2 * t.getBoost( ) * normt,d\right.$$

1. Ο παράγοντας $tf(t\ in\ d)$ είναι η κανονικοποιημένη συχνότητα εμφάνισης του όρου t στο έγγραφο d. Τα έγγραφα που έχουν περισσότερες εμφανίσεις ενός δεδομένου όρου λαμβάνουν υψηλότερη βαθμολογία. Ο προκαθορισμένος υπολογισμός του είναι :

$$tf(t\ in\ d) = \sqrt{frequency}$$

2. Ο παράγοντας $idf(t)$ είναι η αντίστροφη συχνότητα του όρου t. Εμφανίζεται για τον ίδιο όρο t και στο ερώτημα και στο έγγραφο, γι' αυτό έχει υψωθεί στο τετράγωνο. Η τιμή του συσχετίζεται με το αντίστροφο του *docFreq* (δηλαδή του αριθμού των εγγράφων στα οποία εμφανίζεται ο όρος *t*). Αυτό σημαίνει ότι πιο σπάνιοι όροι δίνουν μια υψηλότερη συνεισφορά στη συνολική βαθμολογία. Ο προκαθορισμένος υπολογισμός για το $idf(t)$ είναι:

$$idf(t)\ = 1 + \log\left(\frac{numDocs}{docFreq + 1}\right)$$

3. Ο παράγοντας $coord(q,d)$ εξαρτάται από το πλήθος των όρων του ερωτήματος που εμφανίζονται στο συγκεκριμένο έγγραφο. Τυπικά ένα έγγραφο που περιλαμβάνει περισσότερους από τους όρους του ερωτήματος λαμβάνει μια υψηλότερη βαθμολογία από ένα άλλο με λιγότερους όρους.

4. $O\ queryNorm(q)$ είναι ένας παράγοντας κανονικοποίησης που χρησιμοποιείται ώστε να καθιστούν συγκρίσιμες, οι βαθμολογίες μεταξύ ερωτημάτων. Αυτός ο παράγοντας δεν επηρεάζει την κατάταξη των εγγράφων (αφού όλα τα έγγραφα πολλαπλασιάζονται με το ίδιο διάνυσμα ερωτήματος), αλλά απλά προσπαθεί να κάνει τις βαθμολογίες μεταξύ διαφορετικών ερωτημάτων συγκρίσιμες.

5. Οι παράγοντες $getBoost(\ )$ και $norm(t,d)$ χρησιμοποιούνται για να προάγουν ένα συγκεκριμένο όρο t για το ερώτημα q. Οι προκαθορισμένες τιμές στη Lucene για αυτούς είναι ίση με τη μονάδα.

Συνοψίζοντας τα παραπάνω :

- Έγγραφα που περιλαμβάνουν όλους τους όρους του ερωτήματος είναι περισσότερο επιθυμητά.
- Οι αντιστοιχήσεις σπάνιων λέξεων μεταξύ εγγράφου και ερωτήματος είναι παρέχουν υψηλότερη βαθμολογία από τις αντιστοιχήσεις συνηθισμένων λέξεων.
- Τα μεγάλα σε έκταση έγγραφα δεν λαμβάνουν τόσο υψηλή βαθμολογία όσο τα μικρότερα.
- Έγγραφα που περιέχουν πολλές φορές τους όρους του ερωτήματος είναι περισσότερο επιθυμητά.

2.9.2. Εναλλακτικές μέθοδοι βαθμολόγησης

Εκτός από την καθορισμένη συνάρτηση βαθμολόγησης η Lucene είναι επεκτάσιμη, παρέχει δηλαδή τη δυνατότητα δημιουργίας και άλλων μεθόδων βαθμολόγησης. Η ανάπτυξη αυτών των τεχνικών χαρακτηρίζεται ως expert level task, συνεπώς συνίσταται ο χρήστης να γνωρίζει επαρκώς τις τεχνικές ανάκτησης πληροφορίας αλλά και τη λειτουργία της βιβλιοθήκης. Αλλάζοντας τον τρόπο βαθμολόγησης μπορεί να αλλάξουν πολλά στην λειτουργία της βιβλιοθήκης πέρα από τον καθορισμό της ομοιότητας, διότι η τεχνική βαθμολόγησης είναι ένας πολύπλοκος μηχανισμός που βασίζεται σε τρείς βασικές κλάσεις :

- Η Query, αποτελεί τον καταλύτη για τις υπόλοιπες κλάσεις βαθμολόγησης, αφού είναι αυτή που τις «δημιουργεί» και συντονίζει τη λειτουργία τους.

- Η Weight, χρησιμεύει για την εσωτερική αναπαράσταση των ερωτημάτων του χρήστη. Περιλαμβάνει τις παρακάτω έξι μεθόδους που πρέπει να υλοποιηθούν :

1. Weight#getQuery()
2. Weight#getValue()
3. Weight#sumOfSquaredWeights()
4. Weight#normalize(float)
5. Weight#scorer(IndexReader, boolean, boolean)
6. Weight#explain(Searcher, IndexReader, int)

- Η Scorer, περιέχει κοινές λειτουργίες για όλες τις υλοποιήσεις βαθμολόγησης και αποτελεί τον πυρήνα της διαδικασίας βαθμολόγησης της Lucene και απαιτεί την υλοποίηση των παρακάτω μεθόδων :

1. DocIdSetIterator#docID()
2. Scorer#score(Collector)
3. Scorer#score()
4. DocIdSetIterator#advance(int)

2.9.3. Βαθμολόγηση με τη μέθοδο Okapi-BM25 στη Lucene

Μία εναλλακτική μέθοδος βαθμολόγησης που έχει υλοποιηθεί στη Lucene, αποτυπώνει το πιθανοτικό μοντέλο και χρησιμοποιεί τη συνάρτηση ομοιότητας Okapi-BM25. Μία υλοποίηση έχει γίνει από τον Joaquin Perez-Iglesias και παρουσιάζεται στο http://nlp.uned.es/~jperezi/Lucene-BM25/. Σύμφωνα με αυτή χρησιμοποιούνται δύο προσεγγίσεις για τον υπολογισμό της ομοιότητας η BM25 και η BM25F. Η βαθμολογία ενός εγγράφου d σε σχέση με ένα ερώτημα q με τη BM25 υπολογίζεται ως εξής :

$$R(q,d) = \sum_{t\ in\ q} \frac{tf(t\ in\ d)}{k_1\left((1-b) + b * \frac{l_d}{avl_d}\right) + tf\ (t\ in\ d)} * idf(t)$$

Όπου l_d είναι το μήκος του εγγράφου d, avl_d είναι το μέσο μήκος των εγγράφων στη συλλογή, οι ελεύθερες παράμετροι k_1 και b συνήθως λαμβάνουν τιμή 2 και 0,75 αντίστοιχα. Για την λειτουργία αυτής της μεθόδου είναι αναγκαίος ο υπολογισμός του μήκους κάθε εγγράφου που ευρετηριάζεται καθώς και η αποθήκευση της τιμής του, ώστε έπειτα να υπολογιστεί το μέσο μήκος.

Στη δεύτερη προσέγγιση BM25F η οποία υλοποιήθηκε και βασίζεται στο πιθανοτικό μοντέλο, αρχικά υπολογίζεται το συγκεντρωτικό βάρος ενός όρου από όλα τα πεδία και έπειτα πολλαπλασιάζεται με τη σταθερά (k_1 + 1), ώστε να ομαλοποιηθεί το βάρος του με τη

συχνότητα που ισούται με τη μονάδα, η οποία προκύπτει στα έγγραφα με μέσο μήκος. Το βάρος του όρου υπολογίζεται ως :

$$weight(t,d) = \sum_{c\ in\ d} \frac{occurs_{t,c}^{d} * boost_c}{\left((1-b_c) + b_c * \frac{l_c}{avl_c}\right)}$$

Όπου l_c είναι το μήκος του πεδίου, avl_c είναι το μέσο μήκος του πεδίου, η σταθερά b_c σχετίζεται με το μήκος του πεδίου, όπως η b στη BM25, και ο παράγοντας $boost_c$ αποτελεί την ώθηση που εφαρμόζεται στο πεδίο c. Έπειτα η ομοιότητα υπολογίζεται ως :

$$R(q,d) = \sum_{c\ in\ d} \frac{weight(t,d)}{k_1 + weight(t,d)} * idf(t)$$

2.10. Μέθοδος Κορυφογραμμής (Skyline)

Η ραγδαία ανάπτυξη του διαδικτύου και των εφαρμογών που διαχειρίζονται πολυδιάστατα δεδομένα όπως τα γεωγραφικά συστήματα πληροφοριών (GIS), πολυμεσικού περιεχομένου, συστήματα υποστήριξης για τη λήψη αποφάσεων κ.ά., οδήγησε στην ανάγκη για την εύρεση αποδοτικών τεχνικών που θα βελτιώσουν την επεξεργασία των πολυδιάστατων δεδομένων και θα προσφέρουν ικανοποιητικά αποτελέσματα.

Προς αυτήν την κατεύθυνση αναπτύχθηκε η τεχνική του υπολογισμού της κορυφογραμμής, το πρόβλημα της κορυφογραμμής ήταν γνωστό παλαιότερα ως «το πρόβλημα των μέγιστων διανυσμάτων». Σήμερα, χρησιμοποιείται ο όρος κορυφογραμμή (skyline) λόγω της γραφικής αναπαράστασής του.

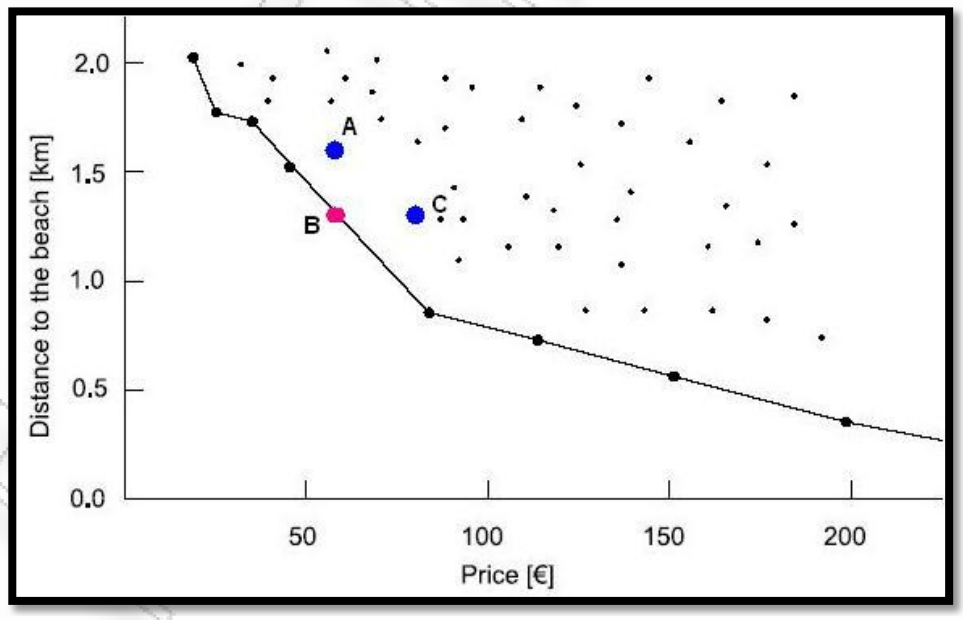
Ο τελεστής κορυφογραμμής (skyline operator) έχει αποδειχθεί σημαντικός για πολλές εφαρμογές που αφορούν την λήψη απόφασης με χρήση πολλαπλών κριτηρίων. Το πρόβλημα υπολογισμού της κορυφογραμμής εντοπίζεται σε πολλούς τομείς της πληροφορικής, στη μηχανική κ.ά. Έχει ιδιαίτερη χρησιμότητα στην εξόρυξη δεδομένων, στις βάσεις δεδομένων, όταν θα πρέπει να παρθεί κάποια απόφαση με βάση πολλά κριτήρια, στους προοδευτικούς αλγορίθμους οι οποίοι μπορούν να επιστρέψουν γρήγορα τα πρώτα σημεία της κορυφογραμμής χωρίς να χρειάζεται να διαβάσουν ολόκληρο το αρχείο δεδομένων, στην εύρεση των top-k ερωτημάτων (top-k queries) και της αναζήτησης του πλησιέστερου γείτονα (nearest neighbor search). Η εύρεση της κορυφογραμμής (skyline) ενός συνόλου πολυδιάστατων δεδομένων αποτελεί πολύ βασική λειτουργία για τα σύγχρονα συστήματα, συνήθως πρέπει να εξυπηρετηθούν ανάγκες που να ικανοποιούν πολλά χαρακτηριστικά των διαθέσιμων δεδομένων. Τα ερωτήματα κορυφογραμμής (skyline queries) είναι ένας από τους πολλούς τύπους ερωτημάτων που μπορούν να εφαρμοστούν πάνω σε πολυδιάστατα δεδομένα (multidimensional data). Αυτά τα ερωτήματα έχουν προσελκύσει το ενδιαφέρον μεγάλου μέρους της επιστημονικής κοινότητας, από ποικίλες περιοχές λόγω της ιδιαίτερης χρησιμότητας που παρουσιάζουν.

Συγκεκριμένα το Skyline ενός συνόλου σημείων ορίζεται ως τα σημεία που δεν κυριαρχούνται από κανένα άλλο σημείο. Ένα σημείο p κυριαρχεί ενός άλλου σημείου q αν το p δεν είναι χειρότερο σε καμία διάσταση (από αυτές που εξετάζονται) από το q και είναι καλύτερο σε τουλάχιστον μια. Δηλαδή, όταν είναι το ίδιο καλό ως προς όλα τα χαρακτηριστικά του και υπερτερεί τουλάχιστον σε ένα από αυτά.

Η σύνταξη ενός Skyline ερωτήματος, όπως ορίστηκε από τους Borzsonyi, Kossmann και Stocker, είναι η εξής :

```
SELECT … FROM … WHERE …
GROUP BY … HAVING …
SKYLINE OF [DISTINCT] d1 [MIN |MAX |DIFF], …, dm [MIN |MAX |DIFF]
ORDER BY …
```

Όπου όταν έχουμε MIN επιχειρείται η ελαχιστοποίηση, MAX η μεγιστοποίηση και DIFF η διαφοροποίηση. Σύμφωνα με τα παραπάνω λοιπόν, ένα ερώτημα κορυφογραμμής επιστρέφει εκείνα τα σημεία του συνόλου τα οποία δεν κυριαρχούνται από κανένα άλλο. Ένα παράδειγμα εφαρμογής του Skyline ερωτήματος αποτελεί η εύρεση των ξενοδοχείων τα οποία βρίσκονται κοντά στην παραλία αλλά ταυτόχρονα είναι φθηνά. Υποθέτουμε την ύπαρξη ενός συνόλου δεδομένων που περιέχει πληροφορίες για ξενοδοχεία, τα δεδομένα που θα εξεταστούν είναι η απόσταση από την παραλία και η τιμή, συνεπώς μπορεί να αναπαρασταθεί σε δύο διαστάσεις όπως φαίνεται στο παρακάτω σχήμα:



Σχήμα 6: Παράδειγμα Skyline, πηγή http://www.dbis.ethz.ch/research/research_topics/exploring_large_db

2.10.1. Skyline Text

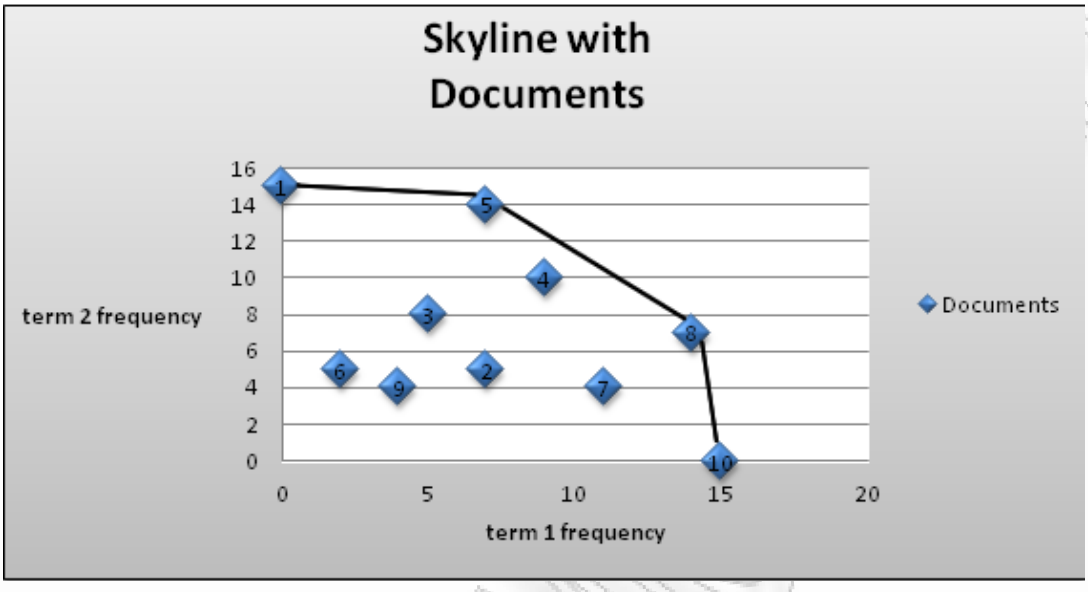
Στην παρούσα πτυχιακή όπως αναφέρθηκε, στοχεύουμε στη δημιουργία μιας εναλλακτικής μεθόδου ανάκτησης πληροφορίας. Με τη χρήση της εφαρμογής παρέχεται η δυνατότητα διαχείρισης πολυδιάστατων δεδομένων και αφού οι skyline επερωτήσεις εφαρμόζονται πάνω σε αυτού του τύπου τα δεδομένα, επεκτείνουμε αυτή τη μεθοδολογία σε έγγραφα ώστε να ανακτήσουμε πληροφορία. Επιπλέον, στην προηγούμενη ενότητα αναλύθηκε η σχέση υπεροχής μεταξύ δύο σημείων, αυτή η έννοια σχετίζεται με το βαθμό ομοιότητας που προκύπτει από τον υπολογισμό του συνημιτόνου της γωνίας των διανυσμάτων του ερωτήματος και του εγγράφου προς βαθμολόγηση. Δηλαδή όταν ένα έγγραφο d_i κυριαρχεί επί ενός άλλου d_j, τότε για οποιοδήποτε δοθέν ερώτημα το d_i παρουσιάζει μεγαλύτερο βαθμός ομοιότητας σε αυτό από ότι το d_j. Συνεπώς, το d_i κατατάσσεται υψηλότερα από το d_j για οποιοδήποτε δοθέν ερώτημα, όταν χρησιμοποιείται για τη βαθμολόγηση η ομοιότητα συνημιτόνου. Αυτό σημαίνει ότι η σχέση κυριαρχίας μεταξύ των εγγράφων υποσκελίζει την έννοια της ομοιότητας συνημιτόνου.

Από τη στιγμή που ως καλύτερα έγγραφα θεωρούνται αυτά που έχουν την υψηλότερη βαθμολογία, η εύρεση της κορυφογραμμής θα είναι πρόβλημα μεγιστοποίησης. Παρατίθεται ένα παράδειγμα για την κατανόηση της μεθόδους σε κείμενα.

Έστω ότι εκτελείται μία επερώτηση με δύο όρους, τον Term 1 και τον Term 2, ως συλλογή εγγράφων διαθέτουμε δέκα αρχεία. Από την ευρετηρίαση γνωρίζουμε τη συχνότητα εμφάνισης του κάθε όρου σε κάθε έγγραφο και την παραθέτουμε στον παρακάτω πίνακα :

	Doc.1	**Doc.2**	**Doc.3**	**Doc.4**	**Doc.5**	**Doc.6**	**Doc. 7**	**Doc. 8**	**Doc.9**	**Doc.10**
Term 1	0	7	5	9	7	2	11	14	4	15
Term 2	15	5	8	10	14	5	4	7	4	0

Από την εκτέλεση του επερωτήματος επιστρέφονται τα παραπάνω έγγραφα, και επειδή υπάρχουν μόνο δύο όροι είναι δυνατή η απεικόνιση της κορυφογραμμής σε δυσδιάστατο σχήμα. Ο άξονας των X λαμβάνει τις τιμές της συχνότητας του όρου Term 1 και ο άξονας Y λαμβάνει τις τιμές της συχνότητας του όρου Term 2. Τα έγγραφα απεικονίζονται ως σημεία στο σχήμα που ακολουθεί, τα 1,5,8 και 10 που κυριαρχούν συνθέτουν την κορυφογραμμή και αυτά που ακολουθούν σε βαθμολογία, τα 4 και 7, βρίσκονται κοντά στο σύνορο.


Skyline with Documents
term 2 frequency
term 1 frequency
Documents
0
2
4
6
8
10
12
14
16
0
5
10
15
20
1
5
4
3
8
6
9
2
7
10

3. Σχεδιασμός - Ανάλυση Απαιτήσεων

Η ανάλυση απαιτήσεων αποτελεί ιδιαίτερα σημαντικό κομμάτι για την επιτυχή ανάπτυξη κάθε εφαρμογής. Αυτή η διαδικασία περιλαμβάνει τον καθορισμό των προδιαγραφών που πρέπει να πληροί το λογισμικό, τη βάση των απαιτήσεων λειτουργίας του, καθώς και τη λειτουργικότητα που θα προσφέρει στο χρήστη.

Το σύστημα ανάκτησης πληροφορίας που υλοποιείται στη παρούσα εργασία, θα προσπελαύνει αρχεία τύπου PDF, ειδικότερα θα διαχειρίζεται επιστημονικά άρθρα από συλλογές συνεδρίων. Θα χρησιμοποιεί τη βιβλιοθήκη Lucene ώστε να πραγματοποιεί τις απαραίτητες μεθόδους ευρετηρίασης και αναζήτησης, και θα δημιουργεί αυτόματα επερωτήσεις ώστε να χρησιμοποιηθούν πειραματικά από την τεχνική της εύρεσης κορυφογραμμής (Skyline). Επιπλέον, θα παράγει αποτελέσματα ώστε να είναι δυνατή η σύγκριση της απόδοσης της αποδεκτής μεθόδου Lucene με της νέας, αυτής του Skyline. Τέλος, για την εκτέλεση όλων των παραπάνω λειτουργιών θα επεκταθεί ανάλογα η υπάρχουσα διεπαφή χρήστη.

3.1. Απαιτήσεις

Οι βασικές λειτουργίες που περιλαμβάνει η εφαρμογή είναι :

- Η δημιουργία ευρετηρίου για μεγάλες συλλογές εγγράφων.
- Η εξαγωγή των όρων του ευρετηρίου σε ένα αρχείο κειμένου.
- Η εκτέλεση επερώτησης μέσω της βιβλιοθήκης Lucene.
- Η εκτέλεση επερώτησης μέσω του αλγορίθμου Skyline.
- Η αυτόματη δημιουργία επερωτήσεων και εκτέλεση τους μέσω της Lucene και του Skyline αλγορίθμου.
- Η εξαγωγή και σύγκριση αποτελεσμάτων.

3.2. Αρχιτεκτονική

Οι απαιτήσεις της εφαρμογής αποτυπώνονται με τη δημιουργία των παρακάτω modules :

- Query Generator, αυτό το module αναπτύχθηκε για την παραγωγή keyword based queries (n το πλήθος μήκους 1 έως k λέξεων από συχνά εμφανιζόμενες λέξεις ή έγγραφα).
- Export Terms, χρησιμοποιείται για την εξαγωγή των όρων του ευρετηρίου έτσι ώστε να είναι δυνατή η άντληση τους από το Query Generator module για την κατασκευή των επερωτήσεων.

- Αξιολόγηση αποτελεσμάτων, δημιουργήθηκε για είναι δυνατή η σύγκριση των αποτελεσμάτων που παράγονται από την εκτέλεση επερωτήσεων με τη μέθοδο Lucene και τον αλγόριθμο Skyline.

- File Manager, χρησιμοποιείται για την διαχείριση όλων των αρχείων που περιέχονται σε μία συλλογή στον ίδιο υποφάκελο με συγκεκριμένη δομή εσωτερικών υποφακέλων.

- Skyline Module, υλοποιήθηκε με στόχο την ενσωμάτωση του αλγορίθμου Skyline στην εφαρμογή.

3.3. Η δυνατότητα επέκτασης και αναβάθμισης

Η γλώσσα Java και οι μεθόδοι υλοποίησης του λογισμικό που παρελήφθη και βασίστηκε η παρούσα εργασία υποστηρίζουν την επεκτασιμότητα του. Προς αυτήν την κατεύθυνση οι βελτιώσεις και οι προσθήκες που πραγματοποιήθηκαν, έγιναν με γνώμονα τη μελλοντική επέκταση και αναβάθμιση της εφαρμογής. Είναι δυνατή η χρησιμοποίηση άλλων μεθόδων για τη δημιουργία του ευρετηρίου, για παράδειγμα η χρήση άλλου analyzer ώστε να γίνεται ειδικά αναζήτηση σε προτάσεις. Επιπλέον, υποστηρίζεται η ευρετηριοποίηση και άλλων γλωσσών πέρα από τα κείμενα που είναι στην αγγλική γλώσσα. Άλλη μια προσθήκη μπορεί να αποτελέσει η ανάπτυξη άλλων μεθόδων βαθμολόγησης μέσω της Lucene, διανέμεται από το Apache Foundation μία υλοποίηση της συνάρτησης βαθμολόγησης Okapi BM25.

4. Υλοποίηση - Ανάπτυξη

Η ανάπτυξη της εφαρμογής ακολουθεί τις προδιαγραφές που τέθηκαν κατά το σχεδιασμό και την ανάλυση των απαιτήσεων, κατά την υλοποίηση λήφθηκε υπόψη η εύχρηστης η διεπαφή χρήστη θα πρέπει να διευκολύνει την εκτέλεση των βασικών λειτουργιών της εφαρμογής. Μέσω της διεπαφής ο χρήστης θα έχει τη δυνατότητα να πραγματοποιεί αναζητήσεις εισάγοντας επερωτήσεις αλλά και να λαμβάνει από την εκτέλεση αυτοματοποιημένων επερωτήσεων αποτελέσματα μέσω δύο διαφορετικών διαδικασιών, της Lucene και του αλγορίθμου Skyline. Η εφαρμογή αποσκοπεί στο να αποτελέσει ένα εργαλείο για τη μέτρηση και σύγκριση της απόδοσης των παραπάνω διαδικασιών ανάκτησης πληροφορίας.

4.1. Εργαλεία Υλοποίησης

Η επέκταση της εφαρμογής πραγματοποιήθηκε σε περιβάλλον Windows, χρησιμοποιώντας την γλώσσα προγραμματισμού Java έκδοσης 1.6.0 και την πλατφόρμα ανάπτυξης Netbeans έκδοσης 7.0. Η γλώσσα και το περιβάλλον ανάπτυξης είναι ανοιχτού κώδικα και διατίθενται από την εταιρεία Sun Microsystems. Επιλέχθηκε το εργαλείο NetBeans διότι προσφέρει ένα εύχρηστο και ολοκληρωμένο περιβάλλον ανάπτυξης και βιβλιοθήκες για την ανάπτυξη της γραφικής διεπαφής χρήστη (GUI). Η γλώσσα προγραμματισμού Java επιλέχθηκε λαμβάνοντας υπόψη ότι ήδη ο πυρήνας του προγράμματος, δηλαδή η βιβλιοθήκη Lucene είναι υλοποιημένη στη γλώσσα Java και επιπλέον τα διάφορα πλεονεκτήματα που μπορούν να προσφέρουν οι κλάσεις που τη συνοδεύουν, και τέλος ότι με αυτήν εξασφαλίζεται η τυχόν μελλοντική επέκταση του λογισμικού.

Επιπροσθέτως, ένα από τα βασικότερα πλεονεκτήματα της Java είναι το χαρακτηριστικό γνώρισμα ότι τα προγράμματα γράφονται μία φορά και μπορούν να τρέξουν σε οποιοδήποτε λειτουργικό σύστημα χωρίς να χρειαστεί να γίνει ξανά μεταγλώττιση ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα.

4.2. Βιβλιοθήκες - Libraries

Όπως έχει αναφερθεί τον πυρήνα της εφαρμογής αποτελεί η βιβλιοθήκη Lucene, χρησιμοποιούμε την έκδοση 3.1.0, πέρα από αυτή χρησιμοποιούνται οι βιβλιοθήκες που παρέχει το NetBeans και αφορούν την ανάπτυξη του γραφικού περιβάλλοντος και κάποιες που διανέμονται από το Apache Foundation και λειτουργούν υποστηρικτικά –συνεργατικά με τη Lucene.

Η εφαρμογή διαχειρίζεται συλλογές εγγράφων .pdf συνεπώς ήταν αναγκαία η χρήση κάποιων βιβλιοθηκών που θα περιείχαν τις μεθόδους επεξεργασίας αυτών. Η Lucene ενσωματώνει στη λειτουργία της, τις βιβλιοθήκες PDFBox και FontBox. Στην εργασία αυτή, χρησιμοποιούνται οι εκδόσεις 1.6.0 και των δύο. Αυτές χρησιμοποιούνται για την επεξεργασία και ανάλυση των .pdf εγγράφων, στα πλαίσια της παρούσας πτυχιακής εντοπίζεται και εξάγεται το κείμενο του

abstract των επιστημονικών άρθρων ώστε αυτό να περάσει από επεξεργασία και να ευρετηριαστεί.

Επιπλέον στην περίπτωση που κάποια από τα έγγραφα είναι σε κρυπτογραφημένη μορφή πέρα από την PDFBox για την προσπέλασή τους χρησιμοποιείται η βιβλιοθήκη Bouncy Castle. Αυτή συμπεριλαμβάνει μια συλλογή από APIS που χρησιμοποιούνται για την κρυπτογράφηση και αποκρυπτογράφηση (http://bouncycastle.org).

4.3. Αποθήκευση Δεδομένων

Δεν χρησιμοποιούμε κάποιο σύστημα διαχείρισης βάσεων δεδομένων, συνεπώς ως μέσο αποθήκευσης χρησιμοποιούνται αρχεία κειμένου (.txt). Αυτά αποθηκεύονται στα παρακάτω subdirectories του directory στο οποίο βρίσκεται η εφαρμογή :

- Φάκελος exports, εδώ αποθηκεύονται παραγόμενα από την εφαρμογή αρχεία, όπως το Terms.txt, το οποίο περιέχει όλους τους όρους του index, κατά φθίνουσα σειρά tf.

- Φάκελος logs\searches, εδώ αποθηκεύονται τα αποτελέσματα των αναζητήσεων του Lucene. Το όνομα του αρχείου που προκύπτει από κάθε αναζήτηση αποτελείται από τους όρους της αναζήτησης ενωμένους μεταξύ τους με underscore (_). Κάθε γραμμή ενός τέτοιου αρχείου περιέχει το id που δίνει η Lucene σε κάθε έγγραφο, καθώς και την τιμή tf για κάθε έναν από τους όρους της αναζήτησης σε αυτό το έγγραφο.

- Φάκελος logs\skyline, εδώ αποθηκεύονται τα αποτελέσματα των αναζητήσεων του Skyline καθώς και προσωρινά αρχεία που παράγονται κατά την εκτέλεση του αλγορίθμου. Το όνομα του αρχείου που προκύπτει από κάθε αναζήτηση αποτελείται από τους όρους της αναζήτησης ενωμένους μεταξύ τους με underscore (_) και την κατάληξη "skyline_Results_Points.txt". Κάθε γραμμή ενός τέτοιου αρχείου περιέχει το id που έχει λάβει κάθε έγγραφο από τη Lucene, καθώς και την τιμή tf για κάθε έναν από τους όρους της αναζήτησης σε αυτό το έγγραφο.

- Φάκελος Imports\pr_results, από εδώ δίνεται η δυνατότητα στο χρήστη να ορίσει ένα αρχείο με «σωστά αποτελέσματα» το οποίο θα χρησιμοποιηθεί για τον υπολογισμό του presicion – recall, των αποτελεσμάτων που έχουν επιστραφεί από την αναζήτηση μέσω της Lucene.

4.4. Ανάπτυξη των λειτουργιών της εφαρμογής

Σε αυτήν την ενότητα περιγράφεται ο τρόπος υλοποίησης των κυρίων λειτουργιών του λογισμικού.

4.4.1. Δημιουργία ευρετηρίου

Η δημιουργία του ευρετηρίου, όπως αναφέρθηκε διαδραματίζει σημαντικό ρόλο για τη διαδικασία της ανάκτησης πληροφορίας. Το ευρετήριο δημιουργείται χρησιμοποιώντας τη Lucene, οι πληροφορίες που απαιτούνται να αποθηκευθούν ώστε να είναι έτοιμες προς επεξεργασία στη συνέχεια, αποθηκεύονται σε binary αρχεία στο δίσκο και δεν χρησιμοποιείται κάποιο σύστημα βάσεων δεδομένων. Η δομή του ευρετηρίου προσομοιάζει τη δομή ενός σχεσιακού συστήματος διαχείρισης δεδομένων, με τα έγγραφα να αναπαριστούν μια εγγραφή-γραμμή και τα πεδία να αποτελούν τις στήλες αυτής της εγγραφής. Κατά τη διαδικασία ευρετηριοποίησης αποθηκεύουμε από κάθε έγγραφο, το id που του δίνεται αυτόματα, τους όρους που εξάγονται από την προσπέλαση του abstract του, το path του (η τοποθεσία που αυτό βρίσκεται στο δίσκο) και την ονομασία του.

Το ευρετήριο μπορεί να ενημερώνεται με νέα έγγραφα που εισάγει ο χρήστης στη συλλογή. Αυτό επιτυγχάνεται αν είναι επιλεγμένο το checkbox στη διεπαφή χρήστη, με αυτόν τον τρόπο παρέχεται η δυνατότητα για τη δημιουργία νέου ευρετηρίου ή επεξεργασίας του ίδιου.

Για τη διαδικασία της ευρετηρίασης ο χρήστης έχει την επιλογή μεταξύ δύο διαφορετικών analyzers, ο Standard Analyzer είναι υλοποιημένος από τη Lucene, ενώ ο PositionalPorterSopAnalyzer υλοποιήθηκε στα πλαίσια της ανάπτυξης αυτού του λογισμικού. Με την χρήση των παραπάνω η Lucene δημιουργεί το ευρετήριο με βάση τη συλλογή εγγράφων που της έχει δοθεί.

Ο Standard Analyzer χρησιμοποιείται για την μετατροπή ενός κειμένου σε tokens, αφαιρεί συγκεκριμένους χαρακτήρες, όπως τα σημεία στίξης ή τις περιόδους από τα ακρωνύμια (η λέξη ΠΑ.ΠΕΙ. μετατρέπεται σε ΠΑΠΕΙ), επίσης μετατρέπει τα κεφαλαία σε μικρά (lowercase) και αφαιρεί τα stop words.

Ο PositionalPorterStopAnalyzer διαχωρίζει το κείμενο σε tokens, απομακρύνει τα σημεία στίξης κτλ. ,μετατρέπει σε lowercase τα tokens και έπειτα χρησιμοποιεί το english stop set του stop analyzer της Lucene για την αφαίρεση των stop words. Τέλος, πραγματοποιεί την τεχνική stemming εφαρμόζοντας τον αλγόριθμο του Porter. Η κλάση PorterStemFilter περιλαμβάνει την υλοποίηση του αλγόριθμου του Porter, όπως αναφέρθηκε σε προηγούμενο κεφάλαιο η διαδικασία αυτή στοχεύει στην απομάκρυνση των κοινών μορφών της λέξης και των καταλήξεών της ώστε να απομείνει μόνο η ρίζα της. Η υλοποίηση του κώδικα παρουσιάζεται στη σελίδα 67 του παραρτήματος.

4.4.2. Εξαγωγή των όρων του ευρετηρίου

Η διαδικασία που ακολουθεί έπειτα από τη δημιουργία του ευρετηρίου είναι η εξαγωγή των όρων του σε ένα αρχείο κειμένου, το Terms.txt. Αυτή η λειτουργία επιτελείται ώστε κάποιοι όροι από αυτό το αρχείο να χρησιμοποιηθούν από το Query Generator module για την παραγωγή επερωτήσεων.

4.4.3. Εκτέλεση επερώτησης μέσω της βιβλιοθήκης Lucene

Η εκτέλεση μίας επερώτησης μέσω της Lucene, μπορεί να πραγματοποιηθεί χρησιμοποιώντας την κλάση TermQuery η οποία αποτελεί επέκταση της Query. Εναλλακτικά μπορεί να χρησιμοποιηθεί ο Query Parser με τον οποίο καλείται η Query και ανάλογα με το περιεχόμενο της ερώτησης καλεί κάποια από το σύνολο των ειδικών κλάσεων που διαθέτει για να γίνει η αναζήτηση. Η TermQuery χρησιμοποιείται για την εκτέλεση ενός ερωτήματος που περιέχει μόνο έναν όρο, ενώ χρησιμοποιώντας την κλάση Query Parser, η επερώτηση μπορεί να περιέχει πολλούς όρους, φράσεις, τελεστές και συγκεκριμένα πεδία που στοχεύει η αναζήτηση, αφού διασπώνται σε όρους και έπειτα όπως περιγράψαμε παραπάνω καλεί την Query για την περαιτέρω εκτέλεση διαδικασίας. Ο κώδικας του TermQuery περιλαμβάνεται στη σελίδα 68 του παραρτήματος και του QueryParser στην 72 .

4.4.4. Εκτέλεση επερώτησης μέσω του αλγορίθμου Skyline

Για την εκτέλεση μίας επερώτησης μέσω του αλγορίθμου Skyline, απαιτείται η τροφοδότηση του αλγορίθμου με τα αποτελέσματα της ίδιας επερώτησης που έχει εκτελεστεί νωρίτερα μέσω της Lucene. Δηλαδή θεωρούμε ως ground truth το σύνολο των εγγράφων που επιστρέφει η Lucene. Όπως αναφέρεται σε παραπάνω κεφάλαιο, η βαθμολόγηση της ομοιότητας των εγγράφων γίνεται χρησιμοποιώντας ως μέτρο το συνημίτονο της εμπεριεχομένης γωνίας (cosine similarity). Έπειτα, τα αποτελέσματα αποθηκεύονται σε ένα αρχείο κειμένου, όπου κάθε γραμμή περιέχει το id που δίνει η Lucene σε κάθε έγγραφο, καθώς και την τιμή tf για κάθε έναν από τους όρους της επερώτησης. Στη συνέχεια εκτελείται ο αλγόριθμος Skyline και τα αποτελέσματά του αποθηκεύονται σε ένα αρχείο κειμένου.

4.4.5. Η εξαγωγή και σύγκριση αποτελεσμάτων

Το module σύγκρισης αποτελεσμάτων δέχεται ως είσοδο τα αποτελέσματα από τις εκτελέσεις ενός επερωτήματος από δύο μεθόδους (Lucene και Skyline) και τις συγκρίνει υπολογίζοντας τις μετρικές αποτίμησης της απόδοσής τους, τις τιμές της ακρίβειας (Precision) και της ανάκλησης (Recall). Οι τιμές τους θα είναι πάντα ίσες, αφού ως ground truth (relevant documents) χρησιμοποιούμε τα n "καλύτερα" αποτελέσματα που επέστρεψε η Lucene, όπου n ειναι το πλήθος των αποτελεσμάτων που επιστρέφει ο αλγόριθμος Skyline (retrieved documents). Για παράδειγμα, αν έπειτα από την εκτέλεση ενός ερωτήματος η Lucene επιστρέψει ως αποτέλεσμα τα έγγραφα {d1,d2,...,d10} και ο αλγόριθμος Skyline επιστρέψει ως αποτέλεσμα τα έγγραφα {d1,d4,d5}, θα συγκριθούν τα τρία καλύτερα αποτελέσματα που έχουν λάβει την υψηλότερη βαθμολογία σύμφωνα με τη Lucene έστω {d1,d2,d3}. Δηλαδή θα θεωρήσουμε ότι αυτά είναι τα σχετικά έγγραφα της συλλογής και τα έγγραφα που ανακτήθηκαν.

Συνεπώς ο αριθμός των σχετικών εγγράφων που ανακτήθηκαν είναι ίσος με τη μονάδα, αφού μόνο το d1 αποτελεί την τομή των δύο συνόλων. Οπότε σύμφωνα με τους ορισμούς την ακρίβειας και της ανάκλησης για το συγκεκριμένο παράδειγμα οι τιμές και των δύο θα είναι 0,333. Για τη ευκολότερη κατανόηση του παραδείγματος, παρατίθενται οι τύποι της ανάκλησης και της ακρίβειας :

$$\text{Recall} = \frac{αριθμός\ σχετικών\ εγγράφων\ που\ ανακτήθηκαν}{αριθμός\ σχετικών\ εγγράφων\ στη\ συλλογή}$$

$$\text{Precision} = \frac{αριθμός\ σχετικών\ εγγράφων\ που\ ανακτήθηκαν}{αριθμός\ εγγράφων\ που\ ανακτήθηκαν}$$

4.4.6. Αυτόματη δημιουργία επερωτήσεων

Σκοπός του module Query Generator είναι η κατασκευή q queries που περιλαμβάνουν t όρους το καθένα, αντλώντας αυτούς τους όρους από το αρχείο Terms.txt το οποίο παράγεται από τη διαδικασία Export Terms. Για τη δημιουργία αυτόματων επερωτήσεων χρησιμοποιούμε τους περισσότερο εμφανιζόμενους όρους στο ευρετήριο. Ανάλογα με τον αριθμό των όρων n που θέλουμε να περιέχει κάθε ερώτημα, αντλούμε συνολικά έναν αριθμό που ισούται με το γινόμενο 30 * n, ώστε να κατασκευαστούν οι επερωτήσεις. Για παράδειγμα αν θέλουμε τα ερωτήματα να περιέχουν 3 όρους, θα χρησιμοποιήσουμε τους 3 * 30 = 90 όρους του ευρετηρίου με την υψηλότερη συχνότητα (tf). Εισάγουμε στο *ArrayList<String> theTerms = getTerms(maxTerms);* ,τους όρους με την υψηλότερη συχνότητα όπως αναφέρθηκε, από το Terms.txt, όπου εκεί όλοι οι όροι είναι ταξινομημένοι κατά φθίνουσα σειρά. Έπειτα γίνονται τυχαίοι συνδυασμοί μεταξύ αυτών των όρων ώστε να σχηματιστούν αυτόματα επερωτήσεις. Ο κώδικας υλοποίησης περιλαμβάνεται στη σελίδα 74 του παραρτήματος.

4.4.7. Δημιουργία module Query Processing

Το συγκεκριμένο module χρησιμοποιεί τις παραπάνω λειτουργίες με αυτοματοποιημένο τρόπο ώστε να διευκολυνθεί η πειραματική μελέτη των αποτελεσμάτων.

4.5. Δομή της Εφαρμογής

Στο παρακάτω σχήμα παρουσιάζεται ένα διάγραμμα με τις λειτουργίες της εφαρμογής που μπορεί να επιτελέσει ο χρήστης. Ουσιαστικά αποτυπώνεται όλη η λειτουργικότητα του εργαλείου. Δηλαδή ο χρήστης μπορεί :

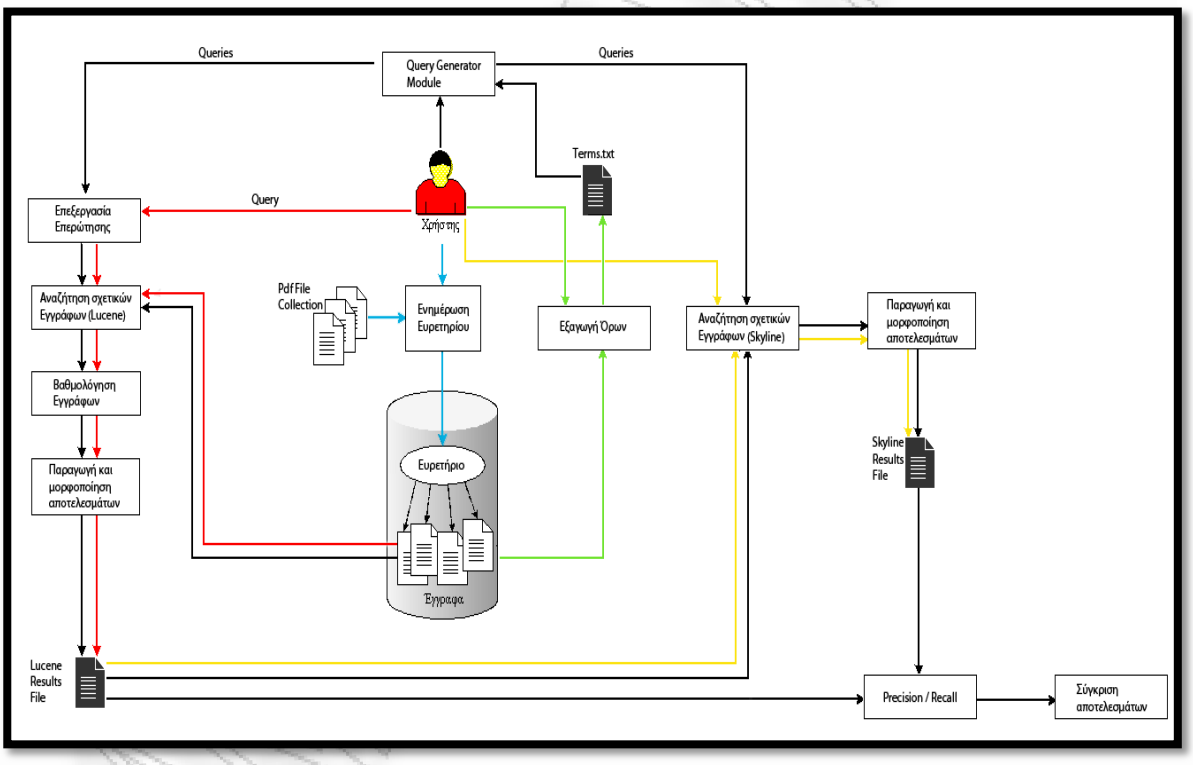
- να δημιουργήσει ή να ενημερώσει το ευρετήριο, η λειτουργία απεικονίζεται με τη ροή γαλάζιου χρώματος.

- να εξάγει τους όρους από το ευρετήριο, αυτή η λειτουργία απεικονίζεται με πράσινο χρώμα.

- να εκτελέσει επερώτηση μέσω της Lucene, η λειτουργία απεικονίζεται με τη ροή κόκκινου χρώματος στο σχήμα.

- να εκτελέσει επερώτηση μέσω του Skyline αλγορίθμου, η λειτουργία παρουσιάζεται ως ροή κίτρινου χρώματος .

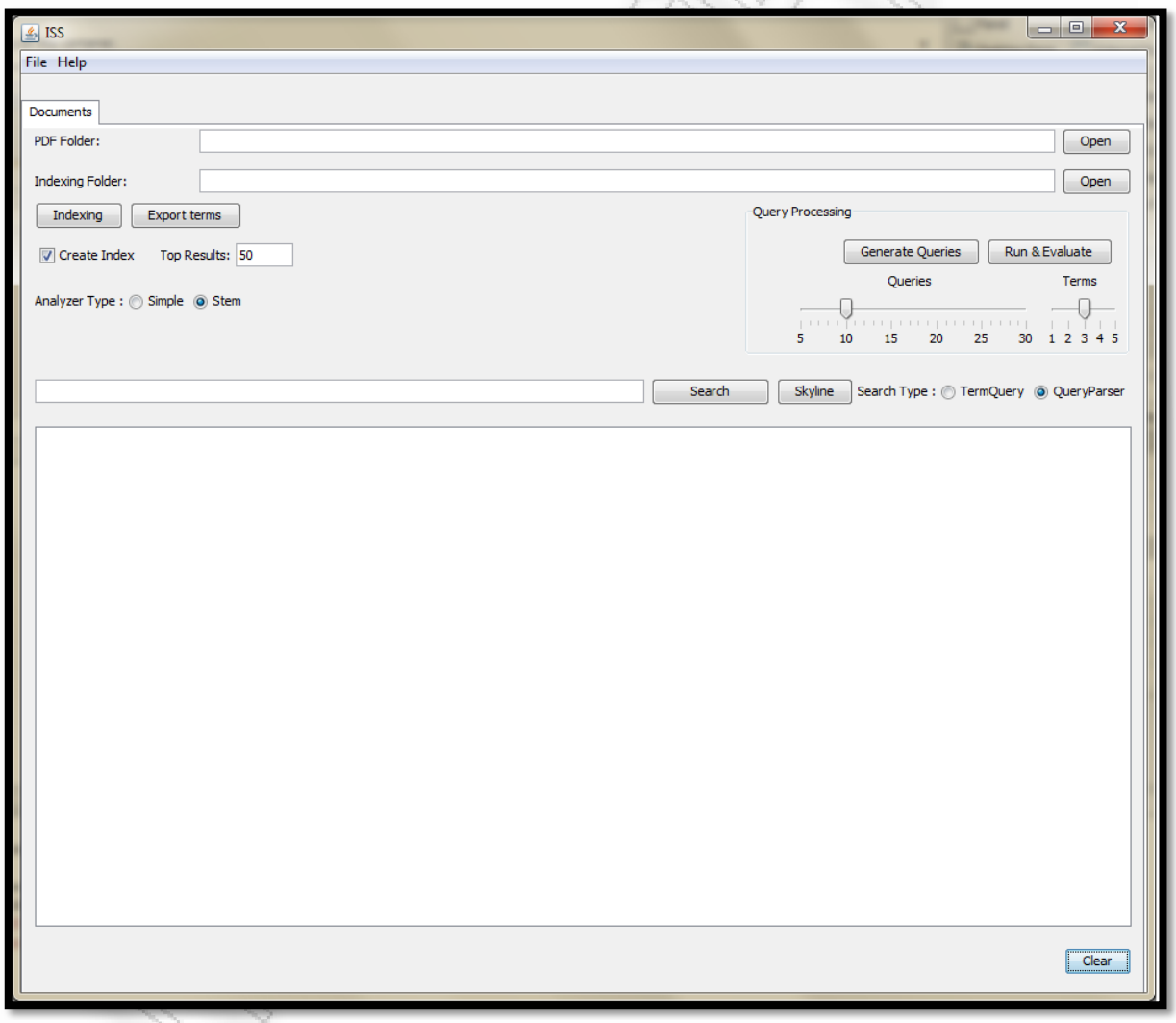
- να δημιουργήσει αυτόματα επερωτήσεις και να τις εκτελέσει μέσω της Lucene και του Skyline αλγορίθμου ώστε να λάβει αποτελέσματα, η διαδικασία απεικονίζεται με τη



ροή μαύρου χρώματος.

4.6. Διεπαφή χρήστη

Σε αυτήν την ενότητα παρατίθεται ένα screenshot της διεπαφής χρήστη, μέσω της οποίας επιτελούνται όλες οι λειτουργίες της.



Εικόνα 6: Διεπαφή Χρήστη

Στο πεδίο που αντιστοιχεί η ετικέτα PDF Folder ο χρήστης έχει την επιλογή να ορίσει τους φακέλους τους οποίους θα χρησιμοποιηθούν για την ευρετηρίαση και γενικά για την ανάκτηση πληροφορίας. Στο πεδίο που αντιστοιχεί η ετικέτα Indexing Folder ο χρήστης επιλέγει τη

directory στο δίσκο όπου θα τοποθετηθούν τα αρχεία που δημιουργεί η Lucene κατά την ευρετηρίαση. Αν θέλουμε να δημιουργήσουμε ένα εντελώς νέο ευρετήριο *Create Index* διαφορετικά αν επιθυμείται να ενημερωθεί το υπάρχον ευρετήριο τότε αποεπιλέγουμε το *Create Index*, έπειτα πατάμε το κουμπί *Indexing* για να ξεκινήσει η διαδικασία ευρετηρίασης. Η διαδικασία που περιγράφηκε απεικονίζεται ως ροή γαλάζιου χρώματος στο σχήμα της ενότητας 4,5.

Αφού έχει δημιουργηθεί το ευρετήριο, ο χρήστης έχει τη δυνατότητα να εξάγει τους όρους του ευρετηρίου σε ένα αρχείο κειμένου με την ονομασία Terms.txt, πατώντας το κουμπί *Export Terms*. Αυτή η λειτουργία επιτελείται ώστε κάποιοι όροι από αυτό το αρχείο να χρησιμοποιηθούν από το Query Generator module για την παραγωγή επερωτήσεων. Η παραπάνω λειτουργία απεικονίζεται με πράσινο χρώμα στο σχήμα της προηγουμένης ενότητας.

Η εκτέλεση μίας επερώτησης μέσω της Lucene, μπορεί να πραγματοποιηθεί χρησιμοποιώντας την κλάση TermQuery ή χρησιμοποιώντας τον QueryParser, συνεπώς επιλέγουμε ανάλογα με τις ανάγκες της αναζήτησης ανάμεσα στις δύο επιλογές, επίσης επιλέγουμε πόσα αποτελέσματα επιθυμούμε να επιστραφούν εισάγοντας την επιθημητή τιμή στο textbox *Top Results,*και εισάγουμε την επερώτηση στο πεδίο της αναζήτησης και πατάμε *Search*. Αυτή η διαδικασία απεικονίζεται με τη ροή κόκκινου χρώματος στο σχήμα της προηγουμένης ενότητας.

Για την εκτέλεση μίας επερώτησης μέσω του αλγορίθμου Skyline πρέπει να επαναληφθεί η παραπάνω διαδικασία και έπειτα η εκτέλεση γίνεται πατώντας το κουμπί *Skyline*. Η λειτουργία αυτή παρουσιάζεται ως ροή κίτρινου χρώματος.

Για τη δημιουργία αυτόματων επερωτήσεων επιλέγουμε τον αριθμό που επιθυμούμε και το πλήθος των key-words που θα χρησιμοποιηθούν. Υπάρχει η επιλογή δημιουργίας μέχρι 30 ερωτημάτων χρησιμοποιώντας μέχρι πέντε όρους. Η παραγωγή τους πραγματοποιείται πατώντας το κουμπί *Generate Queries*, έπειτα η εκτέλεση τους μέσω της Lucene και του Skyline αλγορίθμου και η παραγωγή των αποτελεσμάτων τους γίνεται πατώντας το κουμπί Run & Evaluate. Η διαδικασία απεικονίζεται στο σχήμα με τη ροή μαύρου χρώματος.

5. Παράδειγμα - Παρουσίαση Εφαρμογής

Σε αυτό το κεφάλαιο παρουσιάζεται ένα παράδειγμα αναζήτησης με τρείς λέξεις κλειδιά οι οποίες όταν περικλείονται από εισαγωγικά ο Query Parser τις αντιμετωπίζει ως μία φράση. Χρησιμοποιούμε τον Query Parser διότι θέλουμε να γίνει αναζήτηση για μία ολόκληρη φράση και όχι για έναν όρο, περίπτωση στην οποία θα επιλέγαμε το TermQuery.

Για την εκτέλεση του παραδείγματος ακολουθούμε τη ροή γαλάζιου χρώματος όπως φαίνεται στην εικόνα της ενότητας 4.5, δηλαδή καθορίζουμε ότι οι φάκελοι που θα χρησιμοποιηθούν για την ευρετηρίαση είναι οι SIGMOD,VLDB,ICDE και EDBT, επίσης καθορίζουμε τον φάκελο στο δίσκο όπου θα τοποθετηθούν τα αρχεία που δημιουργεί η Lucene κατά την ευρετηρίαση, έπειτα επιλέγουμε *Create Index* και πατάμε το κουμπί *Indexing* ώστε να δημιουργήσουμε το ευρετήριο.

Η διαδικασία ευρετηριοποίησης διήρκησε 366 δευτερόλεπτα και το παραγόμενο αρχείο έχει μέγεθος 11,3 MB. Αφού δημιουργήθηκε το ευρετήριο, επιλέγουμε την αυτόματη δημιουργία επερώτησης χρησιμοποιώντας τρείς όρους πατώντας το κουμπί *Generate Queries*, δεν εκτελούμε τα ερωτήματα που προέκυψαν απλά επιλέγουμε ένα από αυτά, για παράδειγμα το *present on gener*.

Επιλέγουμε να επιστραφούν έως 100 έγγραφα, εισάγοντας αυτήν την τιμή στο textbox *Top Results*, να γίνει αναζήτηση χρησιμοποιείται ο *Query Parser*, για τους λόγους που αναφέρθηκαν παραπάνω, έπειτα εισάγουμε στο textbox αναζήτησης την επερώτηση "present on gener" και εκτελούμε την επερώτηση πατώντας το κουμπί *Search*, οι λέξεις κλειδιά περικλείονται από τα εισαγωγικά ώστε να αντιμετωπιστούν ως μία πρόταση.

5.1. Ανάλυση Αποτελεσμάτων

Τα αποτελέσματα που επιστρέφονται μέσω της Lucene μαζί με το βαθμό ομοιότητας κάθε εγγράφου, παρουσιάζονται παρακάτω:

1328717319920	-	P347.PDF	-	Score	:	0.49862358
1328717393554	-	p1243-houkjar.pdf	-	Score	:	0.49862358
1329055428356	-	P347.PDF	-	Score	:	0.49862358
1329055533314	-	p1243-houkjar.pdf	-	Score	:	0.49862358
1329056132435	-	P347.PDF	-	Score	:	0.49862358
1329056206130	-	p1243-houkjar.pdf	-	Score	:	0.49862358
1328717312806	-	P165.PDF	-	Score	:	0.43629563
1328717336955	-	P225.PDF	-	Score	:	0.43629563
1328717347438	-	P686.PDF	-	Score	:	0.43629563
1328717364099	-	p476.pdf	-	Score	:	0.43629563
1329055414597	-	P165.PDF	-	Score	:	0.43629563
1329055474236	-	P225.PDF	-	Score	:	0.43629563
1329055485843	-	P686.PDF	-	Score	:	0.43629563
1329055503627	-	p476.pdf	-	Score	:	0.43629563
1329056125135	-	P165.PDF	-	Score	:	0.43629563
1329056149845	-	P225.PDF	-	Score	:	0.43629563

1329056160531 - P686.PDF - Score : 0.43629563
1329056177239 - p476.pdf - Score : 0.43629563
1328717086715 - 22870052.pdf - Score : 0.37396768
1328717142781 - P568.pdf - Score : 0.37396768
1328717145340 - 00710542.pdf - Score : 0.37396768
1328717147477 - 05060547.pdf - Score : 0.37396768
1328717219908 - P080.PDF - Score : 0.37396768
1328717313508 - P316.PDF - Score : 0.37396768
1328717322244 - P195.PDF - Score : 0.37396768
1328717391151 - p958-fagin.pdf - Score : 0.37396768
1328717424645 - R16.pdf - Score : 0.37396768
1329055174558 - 22870052.pdf - Score : 0.37396768
1329055230219 - P568.pdf - Score : 0.37396768
1329055233012 - 00710542.pdf - Score : 0.37396768
1329055235539 - 05060547.pdf - Score : 0.37396768
1329055312199 - P080.PDF - Score : 0.37396768
1329055415299 - P316.PDF - Score : 0.37396768
1329055435938 - P195.PDF - Score : 0.37396768
1329055530006 - p958-fagin.pdf - Score : 0.37396768
1329055565122 - R16.pdf - Score : 0.37396768
1329055887155 - 22870052.pdf - Score : 0.37396768
1329055944782 - P568.pdf - Score : 0.37396768
1329055947980 - 00710542.pdf - Score : 0.37396768
1329055950757 - 05060547.pdf - Score : 0.37396768
1329056027697 - P080.PDF - Score : 0.37396768
1329056125821 - P316.PDF - Score : 0.37396768
1329056134807 - P195.PDF - Score : 0.37396768
1329056203634 - p958-fagin.pdf - Score : 0.37396768
1329056237798 - R16.pdf - Score : 0.37396768
1328717102580 - P072.pdf - Score : 0.31163973
1328717119007 - P470.pdf - Score : 0.31163973
1328717152235 - 2004319144630.20710037.pdf - Score : 0.31163973
1328717157071 - 200517126400.01319983.pdf - Score : 0.31163973
1328717259469 - 2005130133690.R-581.pdf - Score : 0.31163973
1328717310310 - P026.PDF - Score : 0.31163973
1328717359123 - P406.PDF - Score : 0.31163973
1328717415675 - vldb09-75.pdf - Score : 0.31163973
1328717417718 - vldb09-pvldb19.pdf - Score : 0.31163973
1329055190876 - P072.pdf - Score : 0.31163973
1329055206336 - P470.pdf - Score : 0.31163973
1329055240843 - 2004319144630.20710037.pdf - Score : 0.31163973
1329055246241 - 200517126400.01319983.pdf - Score : 0.31163973
1329055354475 - 2005130133690.R-581.pdf - Score : 0.31163973

1329055411883	-	P026.PDF	-	Score	:	0.31163973
1329055498011	-	P406.PDF	-	Score	:	0.31163973
1329055556090	-	vldb09-75.pdf	-	Score	:	0.31163973
1329055558102	-	vldb09-pvldb19.pdf	-	Score	:	0.31163973
1329055905469	-	P072.pdf	-	Score	:	0.31163973
1329055920679	-	P470.pdf	-	Score	:	0.31163973
1329055956436	-	2004319144630.20710037.pdf	-	Score	:	0.31163973
1329055962317	-	200517126400.01319983.pdf	-	Score	:	0.31163973
1329056069645	-	2005130133690.R-581.pdf	-	Score	:	0.31163973
1329056122607	-	P026.PDF	-	Score	:	0.31163973
1329056172418	-	P406.PDF	-	Score	:	0.31163973
1329056228797	-	vldb09-75.pdf	-	Score	:	0.31163973
1329056230840	-	vldb09-pvldb19.pdf	-	Score	:	0.31163973
1328717236147	-	p355.pdf	-	Score	:	0.24931179
1328717242356	-	influencesetsbafls.pdf	-	Score	:	0.24931179
1328717243339	-	p201-korn.pdf	-	Score	:	0.24931179
1328717293930	-	p403-neumann.pdf	-	Score	:	0.24931179
1328717336206	-	P103.PDF	-	Score	:	0.24931179
1328717336643	-	P175.PDF	-	Score	:	0.24931179
1328717373834	-	S11P02.pdf	-	Score	:	0.24931179
1329055328875	-	p355.pdf	-	Score	:	0.24931179
1329055336207	-	influencesetsbafls.pdf	-	Score	:	0.24931179
1329055337361	-	p201-korn.pdf	-	Score	:	0.24931179
1329055394598	-	p403-neumann.pdf	-	Score	:	0.24931179
1329055473316	-	P103.PDF	-	Score	:	0.24931179
1329055473799	-	P175.PDF	-	Score	:	0.24931179
1329055513408	-	S11P02.pdf	-	Score	:	0.24931179
1329056043656	-	p355.pdf	-	Score	:	0.24931179
1329056050988	-	influencesetsbafls.pdf	-	Score	:	0.24931179
1329056052033	-	p201-korn.pdf	-	Score	:	0.24931179
1329056106461	-	p403-neumann.pdf	-	Score	:	0.24931179
1329056148800	-	P103.PDF	-	Score	:	0.24931179
1329056149252	-	P175.PDF	-	Score	:	0.24931179
1329056186770	-	S11P02.pdf	-	Score	:	0.24931179

Η πρώτη στήλη είναι κωδικός id του κάθε αρχείου, η δεύτερη είναι η ονομασία του και η τρίτη είναι ο βαθμός ομοιότητας του εγγράφου με την επερώτηση. Παρατηρείται ότι κάποια έγγραφα έχουν επιστραφεί πολλές φορές, αυτό συμβαίνει διότι περιέχονται στη συλλογή στην οποία εκτελέστηκε η επερώτηση αλλά σε διαφορετικούς υποφακέλους και συνεπώς η Lucene τα αντιμετώπισε ως διαφορετικά και τους έδωσε ξεχωριστά id.

Τα αποτελέσματα που επιστρέφει ο Skyline αλγόριθμος παρουσιάζονται παρακάτω:

id Αρχείου	gener (tf)	present (tf)	on (tf)

1328717393554	6.0	1.0	0.0
1329055533314	6.0	1.0	0.0
1329056206130	6.0	1.0	0.0
1328717086715	4.0	2.0	0.0
1329055174558	4.0	2.0	0.0
1329055887155	4.0	2.0	0.0
1328717102580	1.0	3.0	0.0
1328717310310	1.0	3.0	0.0
1329055190876	1.0	3.0	0.0
1329055411883	1.0	3.0	0.0
1329055905469	1.0	3.0	0.0
1329056122607	1.0	3.0	0.0

Η πρώτη στήλη είναι το id του κάθε αρχείου, η δεύτερη στήλη είναι η συχνότητα εμφάνισης (tf) του όρου "gener" στο abstract του κάθε αρχείου, η τρίτη του όρου "present" και η τέταρτη του όρου "on". Στη συνέχεια παρατίθεται το abstract του πρώτου εγγράφου που επέστρεψε ο Skyline αλγόριθμος με id 1328717393554, στο οποίο οι όροι του έχουν γίνει highlighted.

This paper presents a generic, DBMS independent, and highly extensible relational data generation tool. The tool can efficiently generate realistic test data for OLTP, OLAP, and data streaming applications. The tool uses a graph model to direct the data generation. This model makes it very simple to generate data even for large database schemas with complex inter and intra table relationships. The model also makes it possible to generate data with very accurate characteristics.

Για την παραπάνω αναζήτηση, οι τιμές της ακρίβειας και της ανάκλησης είναι 0,25 .

6. Πειράματα

Σε αυτό το κεφάλαιο αυτό διεξάγονται πειράματα για την αποτίμηση της απόδοσης της νέας μεθόδου, αυτά εκτελούνται σε υπολογιστή ο οποίος διαθέτει περιβάλλον windows, επεξεργαστή PhenomIIx4 965 @3.4Ghz και μνήμη RAM 4GB. Όπως περιγράφηκε στην προηγούμενη ενότητα, για να γίνει η αναζήτηση θα πρέπει αρχικά να εκτελεστούν μία σειρά από βήματα. Αρχικά, επιλέγονται οι φάκελοι των συνεδρίων που επιθυμούμε ως νέα συλλογή προς ευρετηρίαση, αφού ολοκληρωθεί αυτή η διαδικασία, από το παραγόμενο ευρετήριο εξάγουμε τους όρους στο αρχείο Terms.txt και αντλούμε από αυτό τους n * 30 όρους με την υψηλότερη συχνότητας εμφάνισης ώστε να σχηματιστούν, με τυχαίους συνδυασμούς, οι επερωτήσεις προς εκτέλεση. Έπειτα εκτελούνται οι επερωτήσεις με τις μεθόδους ανάκτησης πληροφορίας που ενσωματώνει η Lucene και ο αλγόριθμος Skyline.

6.1. Πείραμα Πρώτο

Για το πρώτο πείραμα θα χρησιμοποιηθούν τα άρθρα από τα επιστημονικά συνέδρια SIGMOD,VLDB,ICDE και EDBT, ο συνολικός αριθμός των αρχείων είναι 5042. Η διαδικασία της ευρετηριοποίησης για αυτά διήρκησε 366 δευτερόλεπτα και το μέγεθος του παραγόμενου ευρετηρίου είναι 11,3 MB.

6.1.1. Δέκα επερωτήσεις χρησιμοποιώντας δύο όρους

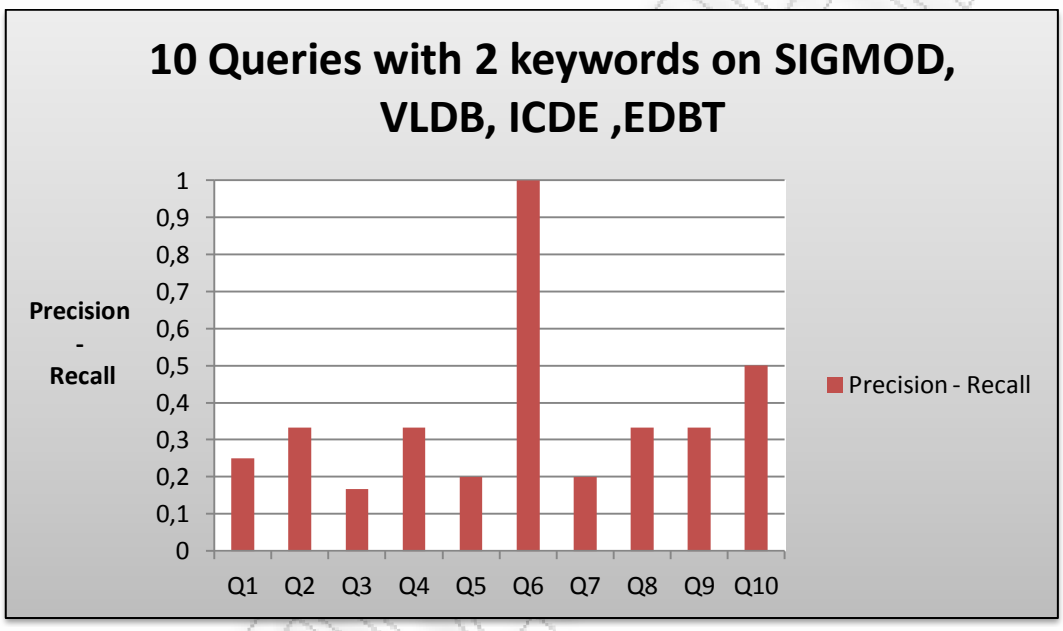
Αρχικά δημιουργούνται δέκα επερωτήσεις που περιέχουν δύο όρους, από το αρχείο Terms.txt αντλούμε τους 2*30 όρους με την υψηλότερη συχνότητας εμφάνισης. Έγινε προσπάθεια ώστε οι όροι που θα συγκροτήσουν τις επερωτήσεις να είναι σχετικοί με τον τομέα των βάσεων δεδομένων και να έχουν νόημα.

Στον παρακάτω πίνακα παρουσιάζονται οι επερωτήσεις μαζί με τα αποτελέσματα που επιστρέφει η Lucene, ο αλγόριθμος Skyline. Η απόδοση του αλγορίθμου φαίνεται από τις τιμές της ακρίβειας και της ανάκλησης.

No	Query	Actual Results	Skyline Results	Precision	Recall
1	data set	50	4	0.25	0.25
2	minimum element	50	3	0.333333	0.333333
3	contain algorithm	50	6	0.166666	0.166666
4	minimum data	50	3	0.333333	0.333333
5	custom data	50	5	0.2	0.2
6	allow problem	50	1	1	1
7	pattern defin	50	5	0.2	0.2
8	data life	50	3	0.333333	0.333333
9	minimum time	50	3	0.333333	0.333333

10	maximum time	50	4	0.5	0.5

Παρατηρούμε ότι εκτελώντας το ερώτημα No6, ο αλγόριθμος Skyline, επιστρέφει ένα έγγραφο, και επιτυγχάνει να επιστρέψει το ίδιο έγγραφο που έχει λάβει την υψηλότερη βαθμολογία και από τη βιβλιοθήκη Lucene. Γι' αυτόν το λόγο η ακρίβεια και η ανάκληση λαμβάνουν τιμή ίση με τη μονάδα. Για να υπάρχει άλλο ένα μέτρο σύγκρισης με τα πειράματα που ακολουθούν, υπολογίσαμε το μέσο όρο της ακρίβειας (είναι ο ίδιος και για την ανάκληση) ο οποίος είναι 0,365.



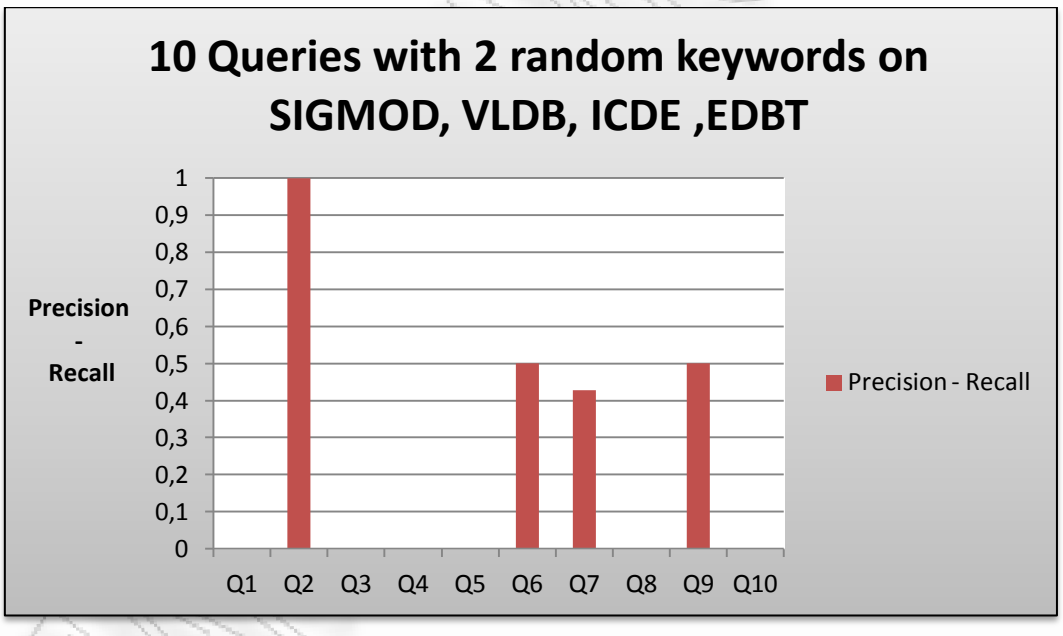
6.1.2. Δέκα επερωτήσεις χρησιμοποιώντας δύο τυχαίους όρους

Στο πείραμα που ακολουθεί δημιουργήσαμε δέκα επερωτήσεις που περιέχουν δύο τυχαίους όρους, ώστε να ελεγχθεί και αυτή η παράμετρος.

No	Query	Actual Results	Skyline Results	Precision	Recall
1	gsp bought	3	1	0	0
2	pattern given	50	1	1	1
3	item across	50	4	0	0
4	adjac scale	50	2	0	0
5	foundat faster	50	4	0	0
6	number element	50	2	0.5	0.5

7	period linearli	50	7	0.428574	0.428574
8	sent time	50	1	0	0
9	maximum set	50	2	0.5	0.5
10	empir ha	50	2	0	0

Παρατηρούμε ότι ο αλγόριθμος skyline αποτυγχάνει να επιστρέψει τις περισσότερες φορές επιτυχώς τα ίδια έγγραφα με αυτά που επιστρέφει η Lucene. Γι' αυτόν το λόγο η ακρίβεια και η ανάκληση λαμβάνουν τις περισσότερες φορές τιμή ίση με το μηδέν, ο μέσος όρος τους είναι 0,243, αριθμός αυτός είναι χαμηλότερος από τον μέσο όρο του αντίστοιχου πειράματος (0,365), όπου επιλέχθηκαν πρώτα οι όροι με την υψηλότερη συχνότητα και έπειτα συνδυάστηκαν με τυχαίο τρόπο. Για αυτά τα αποτελέσματα ίσως να ευθύνεται ο τρόπος δημιουργίας των επερωτήσεων, που όπως αναφέραμε έγινε με εντελώς τυχαίο συνδυασμό όρων χωρίς να επιλεγούν αρχικά αυτοί με την υψηλότερη συχνότητα.

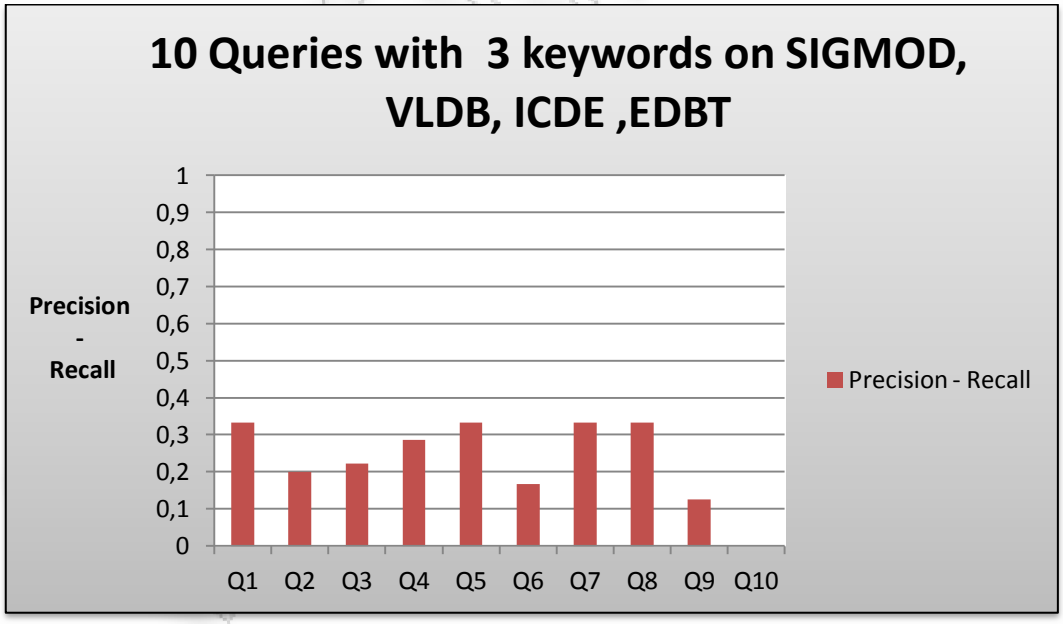


6.1.3. Δέκα επερωτήσεις χρησιμοποιώντας τρείς όρους

Στο πείραμα που ακολουθεί χρησιμοποιούνται οι ίδιες επιστημονικές συλλογές και δημιουργούνται δέκα επερωτήσεις που περιέχουν τρείς όρους η κάθε μία. Δηλαδή από το αρχείο Terms.txt αντλούμε τους 3 * 30 όρους με την υψηλότερη συχνότητας εμφάνισης. Τα αποτελέσματά του παρουσιάζονται στον παρακάτω πίνακα :

No	Query	Actual Results	Skyline Results	Precision	Recall
1	databas taxonomi sequenti	50	3	0.333333	0.333333
2	maximum number problem	50	5	0.2	0.2
3	list hierarchi sequenti	50	9	0.222222	0.222222
4	includ pattern tree	50	7	0.285714	0.285714
5	faster size restric	50	6	0.333333	0.333333
6	size data allow	50	6	0.166666	0.166666
7	present abstract order	50	3	0.333333	0.333333
8	data list pattern	50	12	0.333333	0.333333
9	pre order pattern	50	8	0.125	0.125
10	databas abstract item	50	3	0	0

Παρατηρούμε ότι εκτελώντας την επερώτηση No10 ο αλγόριθμος Skyline, παρότι επιστρέφει τρία έγγραφα, εντούτοις αποτυγχάνει να επιστρέψει έστω και ένα κοινό έγγραφο με τα πρώτα τρία έγγραφα που έχουν την υψηλότερη βαθμολογία από τη βιβλιοθήκη Lucene. Γι' αυτόν το λόγο η ακρίβεια και η ανάκληση λαμβάνουν τιμή μηδέν. Γενικά η χρησιμοποίηση τριών όρων για την αυτόματη παραγωγή ερωτημάτων παρατηρούμε ότι λειτούργησε ανασταλτικά στην επίτευξη υψηλότερης απόδοσης, αφού κατά οι τιμές της ακρίβειας και της ανάκλησης είναι πολύ χαμηλές σχεδόν για όλα τα ερωτήματα. Όντως ο μέσος όρος της απόδοσης από την εκτέλεση αυτής της σειράς επερωτήσεων είναι χαμηλότερος από την αντίστοιχη στην οποία χρησιμοποιήθηκαν δύο όροι, και είναι 0,233293.



6.2. Πείραμα Δεύτερο

Σε αυτό το πείραμα θα χρησιμοποιηθούν τα επιστημονικά άρθρα από τα συνέδρια KDD και ICDM, ο συνολικός αριθμός των αρχείων είναι 1935. Η διαδικασία της ευρετηριοποίησης διήρκησε 132 δευτερόλεπτα και το μέγεθος του παραγόμενου ευρετηρίου είναι 4,54 MB.

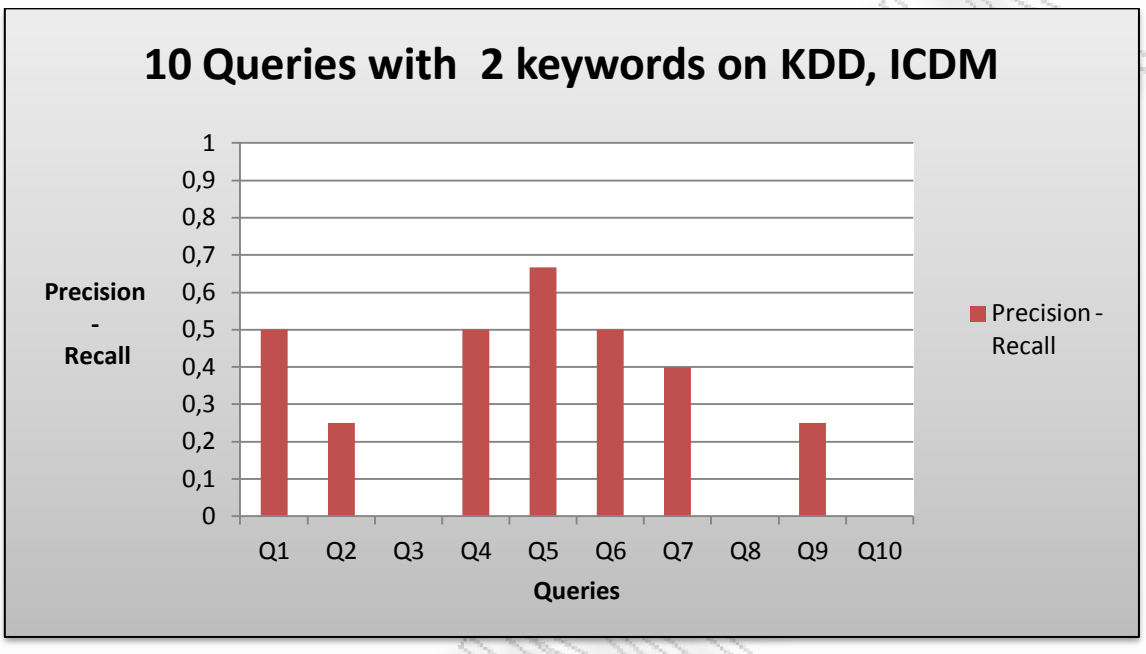
6.2.1. Δέκα επερωτήσεις χρησιμοποιώντας δύο όρους

Δημιουργούνται δέκα επερωτήσεις που περιέχουν δύο όρους, από το αρχείο Terms.txt αντλούμε τους 2 * 30 όρους με την υψηλότερη συχνότητας εμφάνισης. Έπειτα εκτελούνται τα ερωτήματα με τις μεθόδους ανάκτησης πληροφορίας που ενσωματώνει η Lucene και ο αλγόριθμος Skyline.

Στον παρακάτω πίνακα παρουσιάζονται οι επερωτήσεις μαζί με τα αποτελέσματα των δύο μεθόδων και οι τιμές της ακρίβειας και της ανάκλησης

No	Query	Actual Results	Skyline Results	Precision	Recall
1	small set	50	4	0.5	0.5
2	document larg	50	4	0.25	0.25
3	data space	50	4	0	0
4	document text	50	2	0.5	0.5
5	search qualiti	50	3	0.666666	0.666666
6	user method	50	2	0.5	0.5
7	search queri	50	5	0.4	0.4
8	textual represent	50	2	0	0
9	small queri	50	4	0.25	0.25
10	index search	50	3	0	0

Παρατηρούμε ότι εκτελώντας τις επερωτήσεις No3, No8 και No10 ο αλγόριθμος Skyline, παρότι επιστρέφει κάποια έγγραφα, εντούτοις αποτυγχάνει να επιστρέψει έστω και ένα κοινό έγγραφο με τα έγγραφα που έχουν την υψηλότερη βαθμολογία από τη βιβλιοθήκη Lucene για τις αντίστοιχες αναζητήσεις. Γι' αυτόν το λόγο η ακρίβεια και η ανάκληση λαμβάνουν τιμή μηδέν. Ο μέσος όρος της ακρίβειας (ίσος με της ανάκλησης) είναι 0,306667 , διακρίνουμε ότι βρίσκεται κοντά στο μέσο όρο του πρώτου πειράματος (0,365) στο οποίο εκτελέστηκαν δέκα επερωτήσεις και χρησιμοποιήθηκαν δύο όροι σε καθένα από αυτά.

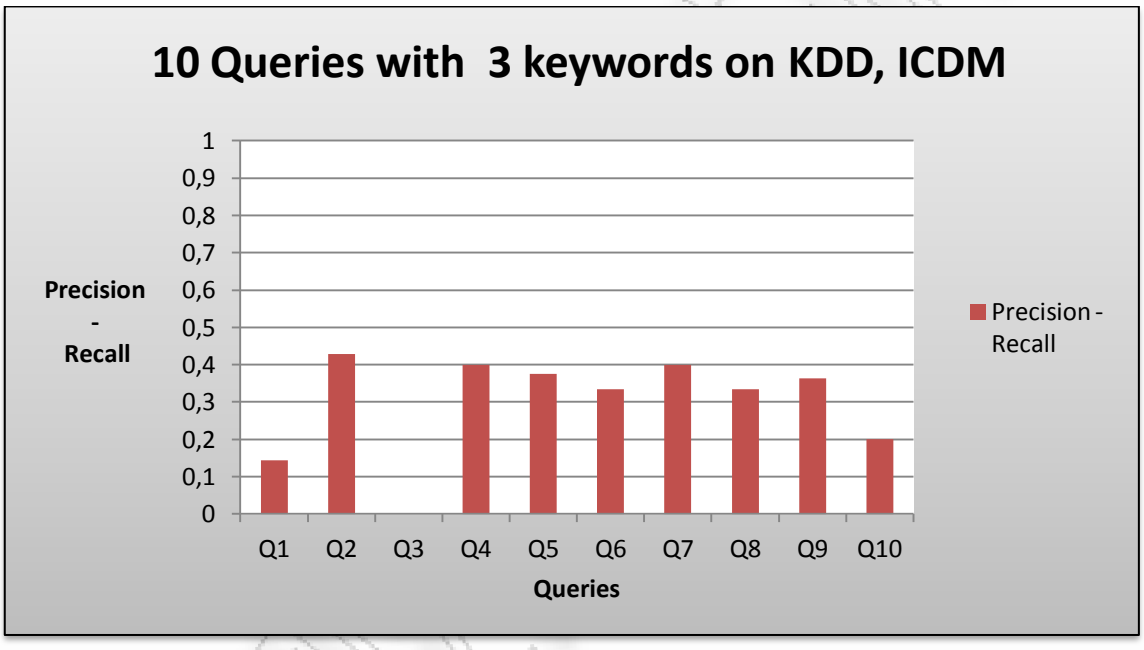


6.2.2. Δέκα επερωτήσεις χρησιμοποιώντας τρείς όρους

Στο πείραμα που ακολουθεί χρησιμοποιούνται οι ίδιες επιστημονικές συλλογές και δημιουργούνται δέκα επερωτήσεις που περιέχουν τρείς όρους η κάθε μία, από το αρχείο Terms.txt αντλούμε τους 3 * 30 όρους με την υψηλότερη συχνότητας εμφάνισης. Τα αποτελέσματά του παρουσιάζονται στον παρακάτω πίνακα :

No	Query	Actual Results	Skyline Results	Precision	Recall
1	map larg text	50	7	0.142857	0.142857
2	space larg search	50	7	0.428571	0.428571
3	ambigu problem data	50	7	0	0
4	keyword nois effect	50	10	0.4	0.4
5	among queri target	50	8	0.375	0.375
6	measur keyword size	50	3	0.333333	0.333333
7	similar search problem	50	5	0.4	0.4
8	provid qualit search	50	6	0.333333	0.333333
9	data sampl measur	50	11	0.363636	0.363636
10	larg text techniqu	50	5	0.2	0.2

Παρατηρούμε ότι εκτελώντας την επερώτηση No3 ο αλγόριθμος Skyline, παρότι επιστρέφει εφτά έγγραφα, εντούτοις αποτυγχάνει να επιστρέψει έστω και ένα κοινό έγγραφο με τα έγγραφα που έχουν την υψηλότερη βαθμολογία από τη βιβλιοθήκη Lucene για τις αντίστοιχες αναζητήσεις. Γι' αυτόν το λόγο η ακρίβεια και η ανάκληση λαμβάνουν τιμή μηδέν. Ο μέσος όρος της ακρίβειας (ίσος με της ανάκλησης) είναι 0,297673. Διακρίνουμε ότι ο μέσος όρος του πειράματος βρίσκεται κοντά με το μέσο όρο του πειράματος (0,233293) στο οποίο εκτελέστηκαν δέκα επερωτήσεις με τρείς όρους για διαφορετική συλλογή.



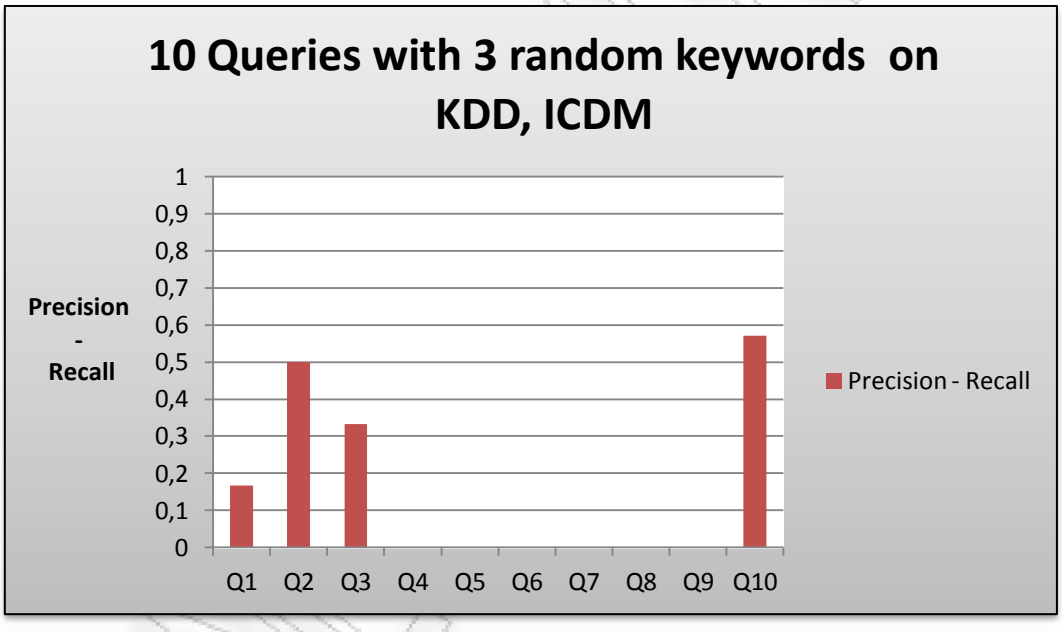
6.2.3. Δέκα επερωτήσεις χρησιμοποιώντας τρείς τυχαίους όρους

Στο πείραμα που ακολουθεί δημιουργήσαμε δέκα επερωτήσεις που περιέχουν τρείς τυχαίους όρους, χρησιμοποιώντας τις συλλογές KDD και ICDM.

No	Query	Actual Results	Skyline Results	Precision	Recall
1	inher even semant	50	6	0.166666	0.166666
2	iti data lsi	50	2	0.5	0.5
3	suitabl scheme ef?cienc	50	6	0.333333	0.333333
4	space tual tation	50	1	0	0
5	similar topic on	50	4	0	0
6	tex queri invert	50	1	0	0

7	wai allow order	50	4	0	0
8	unfortun engin possibl	50	3	0	0
9	both document outperform	50	3	0	0
10	us nois effect	50	7	0.571428	0.571428

Παρατηρούμε ότι ο αλγόριθμος skyline και σε αυτήν την περίπτωση αποτυγχάνει να επιστρέψει τις περισσότερες φορές επιτυχώς τα ίδια έγγραφα με αυτά που επιστρέφει η Lucene. Γι' αυτόν το λόγο η ακρίβεια και η ανάκληση λαμβάνουν τις περισσότερες φορές τιμή ίση με το μηδέν, ο μέσος όρος τους είναι 0,157143, αριθμός χαμηλότερος από το αντίστοιχο πείραμα (0,297673), όπου επιλέχθηκαν πρώτα οι όροι με την υψηλότερη συχνότητα και έπειτα συνδυάστηκαν με τυχαίο τρόπο. Για αυτά τα αποτελέσματα ίσως να ευθύνεται η δημιουργία των επερωτήσεων με εντελώς τυχαίο συνδυασμό όρων.



7. Συμπέρασμα – Επεκτάσεις

Ο κύριος στόχος της εργασίας αυτής ήταν η δημιουργία ενός εύχρηστου desktop εργαλείου που θα δημιουργεί αυτόματα επερωτήσεις χρησιμοποιώντας τους όρους του ευρετηρίου που δημιουργείται από τις μεθόδους της βιβλιοθήκης Lucene και θα ενσωματώνει την τεχνική εύρεσης της κορυφογραμμής (Skyline) ώστε να εκτελεί τις παραπάνω επερωτήσεις και να προσφέρει αποτελέσματα που αφορούν την απόδοση της νέας μεθόδου.

Από τα πειράματα που έγιναν διαπιστώθηκε ότι αλγόριθμος κορυφογραμμής που χρησιμοποιείται στην εφαρμογή, για τις περισσότερες εκτελέσεις επερωτήσεων δεν επιστρέφει τα ταυτόσημα αποτελέσματα με τη Lucene, σε κάποιες αποτυγχάνει εντελώς και σε κάποιες επιστρέφει με επιτυχία τα ίδια έγγραφα τα οποία έχουν λάβει την υψηλότερη βαθμολογία από την μέθοδο βαθμολόγησης της Lucene. Ειδικότερα όταν οι επερωτήσεις έχουν δημιουργηθεί με τη χρήση εντελώς τυχαίων όρων η απόδοση του είναι χαμηλότερη.

Η απόδοση του αλγορίθμου όμως δεν μπορεί να κριθεί αυστηρά διότι χρησιμοποιείται για την εκτέλεση επερωτήσεων που έχουν δημιουργηθεί με σχεδόν τυχαίο τρόπο. Πέρα από τη χρήση των όρων με τη μεγαλύτερη συχνότητα οι οποίοι χρησιμοποιούνται στην παρούσα εργασία, θα πρέπει να επινοηθούν τρόποι για την βελτίωση της ποιότητας της ανάκτησης. Προς αυτήν την κατεύθυνση, έγιναν κάποιες δοκιμές με τις τεχνικές της προεπεξεργασίας του κειμένου των εγγράφων, δηλαδή των τεχνικών που λαμβάνουν μέρος κατά της ευρετηρίαση. Ένα παράδειγμα αποτελεί η ενημέρωση του συνόλου των stop words ώστε να μην ευρετηριοποιούνται λέξεις που δεν έχουν ιδιαίτερη αξία κατά την ανάκτηση. Επιπλέον, οι διαδικασίες της ανάλυσης έχουν εφαρμογή όταν το κείμενο είναι στην αγγλική γλώσσα, αν επιθυμούμε την ανάκτηση εγγράφων που περιέχουν κείμενο σε άλλη γλώσσα θα πρέπει να χρησιμοποιηθούν ή να υλοποιηθούν άλλοι analyzer (π.χ. SnowballAnalyzer της Lucene) και να διαφοροποιηθεί το σύνολο των stop words.

Με στόχο να γίνει αποδοτικότερη η αναζήτηση και να αποκτήσουν καλύτερη εννοιολογική σημασία οι επερωτήσεις, πέρα από τους όρους του ευρετηρίου που χρησιμοποιούνται ως λέξεις κλειδιά για τη κατασκευή των επερωτήσεων θα πρέπει να βρεθούν τεχνικές για την αυτοματοποιημένη δημιουργία φράσεων. Αυτό μπορεί επιτευχθεί μέσω της χρήσης τεχνικών επεξεργασίας κειμένου που δεν διασπούν τις λέξεις αλλά ευρετηριάζουν ολόκληρες προτάσεις ή άλλων τεχνικών οι οποίες θα διατηρούν τις αποστάσεις των λέξεων ώστε να είναι δυνατή αργότερα η συγκρότησή τους σε προτάσεις.

Επίσης, από τη στιγμή που χρησιμοποιείται ως ground truth, δηλαδή ως σχετικά έγγραφα χρησιμοποιείται ο n αριθμός των εγγράφων που έχουν λάβει την υψηλότερη βαθμολογία, όπου το n καθορίζεται από τον αριθμό των εγγράφων που επιστρέφει η Lucene, θεωρούμε ότι θα πρέπει να δοκιμαστεί η βαθμολόγηση μέσω εναλλακτικών μεθόδων ανάκτηση πληροφορίας. Στην βιβλιοθήκη Lucene χρησιμοποιείται ένα μείγμα του δυαδικού και του διανυσματικού μοντέλου. Μία εναλλακτική μέθοδος βαθμολόγησης αποτελεί η χρήση του πιθανοτικού μοντέλου (Okapi BM25), η ενσωμάτωσή του στη βιβλιοθήκη Lucene μπορεί να γίνει χρησιμοποιώντας την προσέγγιση του Joaquin Perez-Iglesias από το http://nlp.uned.es/~jperezi/Lucene-BM25/ .

Συνοψίζοντας, θεωρούμε ότι έγινε κάποιο βήμα προόδου για την ενσωμάτωση νέων τεχνικών για την ανάκτηση πληροφορίας ωστόσο απαιτούνται νέες μεθόδοι οι οποίες θα βελτιώσουν την απόδοσή τους. Τέλος, όσον αφορά τη διεπαφή, μια μελλοντική επέκταση με στόχο να γίνει η εφαρμογή φιλικότερη προς το χρήστη μπορεί να είναι η εμφάνιση των αποτελεσμάτων με τη χρήση διαγραμμάτων.

Βιβλιογραφία

1. Ευρετηριοποίηση και Αναζήτηση με Λέξεις Κλειδιά σε Συλλογές Κειμένων, Βανός, Κακοσίμος, Μαθιουδάκης, Εργασία στα πλαίσια του μαθήματος Διαχείριση Δεδομένων, Μεταπτυχιακό Δικτυοκεντρικά Συστήματα, Πανεπιστήμιο Πειραιώς, Τμήμα Ψηφιακών Συστημάτων.

2. Σημειώσεις μαθήματος Ανάκτησης Πληροφορίας, Πανεπιστήμιο Αιγαίου, Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων.

3. Ανάκτηση Πληροφορίας, Σημειώσεις Διαλέξεων, Ιωάννης Μανωλόπουλος, Απόστολος Ν. Παπαδόπουλος, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης , Τμήμα Πληροφορικής.

4. Ανάκτηση Πληροφοριών, Σημειώσεις Μαθήματος, Κωνσταντόπουλος Χάρης, Πανεπιστήμιο Πειραιώς, Τμήμα Πληροφορικής .

5. Introduction to Information Retrieval By Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, 2008 Cambridge University Press.

6. S. Borzsonyi, D. Kossmann, K. Stocker, "The skyline operator", Proceedings of the International Conference on Data Engineering (ICDE), pp. 421-430, 2001.

7. Integrating the Probabilistic Model BM25/BM25F into Lucene, Joaquın Perez-Iglesias, Jose R. Perez-Aguera, Vıctor Fresno and Yuval Z. Feinstein

8. Υλοποίηση και βελτίωση τυχαίου αλγορίθμου εύρεσης κορυφογραμμής, Πτυχιακή εργασία, Σανιδά Σταυρούλα, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, Σχολή Θετικών Επιστημών, Τμήμα Πληροφορικής.

9. Αυτόματη Θεματική Κατηγοριοποίηση και Σημασιολογική Διεύρυνση Ερωτημάτων για Μηχανή Αναζήτησης με Οντολογίες, Διπλωματική εργασία της Αμαλίας Κούρτη, Εθνικό Μετσόβιο Πολυτεχνείο Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

10. Αποσαφήνιση Λέξεων με Βάση τα Google 5-grams, Διπλωματική Εργασία, Πολυξένη Π. Κατσιούλη, Οικονομικό Πανεπιστήμιο Αθηνών, Τμήμα Πληροφορικής, Μεταπτυχιακό Πρόγραμμα Σπουδών Επιστήμη των Υπολογιστών.

11. Ταξινόμηση Συνόψεων Αντικειμένων με τη Χρήση Τεχνικών Εξόρυξης Δεδομένων, Ατομική Διπλωματική Εργασία, Ιουλία Βανέζη Πανεπιστήμιο Κύπρου, Τμήμα Πληροφορικής.

12. Αποδοτικό Ξεφύλλισμα σε Ιατρικές Βάσεις Δεδομένων, Διπλωματική Εργασία, Σάββας Πέτρου Πολυτεχνείο Κρήτης, Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών.

13. Μηχανισμοί και Τεχνικές Διαχείρισης, Επεξεργασίας, Ανάλυσης, Κατηγοριοποίησης, Εξαγωγής Περίληψης και Προσωποποίησης Συχνά Ανανεώσιμων Δεδομένων του Παγκοσμίου Ιστού για Παρουσίαση Σε Σταθερές και Κινητές Συσκευές, Διδακτορική Διατριβή, Βασίλειος Ν. Πουλόπουλος, -Πανεπιστήμιο Πατρών, Πολυτεχνική Σχολή Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών & Πληροφορικής.

14. Υλοποίηση και βελτίωση τυχαίου αλγορίθμου εύρεσης κορυφογραμμής , Πτυχιακή Εργασία, Σανιδά Σταυρούλα, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, Σχολή Θετικών Επιστημών, Τμήμα Πληροφορικής.

15. Συνεχής Υπολογισμός Γενικευμένων Skyline Ερωτημάτων σε Κατανεμημένα Συστήματα, Μεταπτυχιακή Εργασία Εξειδίκευσης , Αικατερίνη Φωτιάδου, Πανεπιστήμιο Ιωαννίνων, Τμήμα Πληροφορικής.

16. Αναζήτηση Κορυφογραμμής, Νικόλαος Γ. Παπαβασιλείου, Διπλωματική Εργασία, Πανεπιστήμιο Θεσσαλίας, Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών Τηλεπικοινωνιών και Δικτύων.

17. Παράλληλος Υπολογισμός Ερωτημάτων Κορυφογραμμής, Διπλωματική Εργασία, Βαλκανάς Γεώργιος, Πρόγραμμα Μεταπτυχιακών Σπουδών, Κατεύθυνση Πληροφοριακών Συστημάτων, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, Τμήμα Πληροφορικής.

18. QUERYING, EXPLORING AND MINING THE EXTENDED DOCUMENT, Nikolaos Sarkas, Thesis, Department of Computer Science,University of Toronto.

19. On Finding the Maxima of a Set of Vectors, H.T. KUNG, F. LUCCIO, F.P. PREPARATA, Journal of the ACM (JACM) JACM Homepage archive Volume 22 Issue 4, Oct. 1975.

Ηλεκτρονικές πηγές

1. http://lucene.apache.org/java/docs/index.html
2. http://www-csli.stanford.edu/~hinrich/information-retrieval.html
3. http://nlp.stanford.edu/IR-book/html/htmledition/irbook.html
4. http://documents.cfar.umd.edu/resources/ir/
5. http://www.glue.umd.edu/~dlrg/clir/
6. http://www.daviddlewis.com/resources/
7. http://www.cs.umbc.edu/~crowder/pubs/IR.html
8. http://www.google.com/Top/Computers/Software/Information_Retrieval/
9. http://ir.dcs.gla.ac.uk/resources.html
10. http://www.searchtools.com/info/info-retrieval.html
11. http://bitsearch.blogspot.com/2011/01/vector-space-model-for-scoring.html
12. http://blogs.msdn.com/b/spt/archive/2008/03/05/information-retrieval-search-basic-ir-models.aspx
13. http://www.lucidimagination.com/blog/2009/03/18/exploring-lucenes-indexing-code-part-2/
14. http://lucene.apache.org/java/3_0_2/api/core/org/apache/lucene/search/Similarity.html
15. http://lucene.apache.org/java/3_5_0/scoring.html#Introduction
16. http://nlp.uned.es/~jperezi/Lucene-BM25/

Παράρτημα

Κλάση PorterStemFilter

```
public class PositionalPorterStopAnalyzer extends Analyzer {

   private Set stopWords;

   public PositionalPorterStopAnalyzer() {

      this(StopAnalyzer.ENGLISH_STOP_WORDS_SET);

   }

   public PositionalPorterStopAnalyzer(Set stopWords) {

      this.stopWords = stopWords;

   }

   public TokenStream tokenStream(String fieldName, Reader reader) {

      StopFilter stopFilter = new StopFilter(true, new LowerCaseTokenizer(

            reader), stopWords);

      stopFilter.setEnablePositionIncrements(true);

      return new PorterStemFilter(stopFilter);

   }

}
```

Υλοποίηση του κώδικα του TermQuery

```
/**
    * Υλοποίηση της αναζήτησης QuerySearch.
    * Η μέθοδος υλοποιεί αναζήτηση,scoring και εξαγωγή αποτελεσμάτων
    * σε αρχείο κειμένου.
    * @param queryExpression Ο όρος αναζήτησης
    * @param directory Ο φάκελος που περιέχονται τα indexing αρχεία
    * @param searcher
    * @param explanation Αντικείμενο για όλες τις λεπτομέρειες των αποτελεσμάτων
    * @param list Λίστα με τα τελικά αποτελέσματα
    * @param topResults Το μέγιστο πλήθος αποτελεσμάτων με το υψηλότερο score
    * @throws IOException
    */
  protected HashMap termQuerySearch(String queryExpression, Directory directory, IndexSearcher searcher,
          Explanation explanation, ArrayList list, int topResults) throws IOException {
      HashMap resultMap = new HashMap<String,ResultDoc>();
      String[] term = queryExpression.split(" ");
      PrintWriter wr = createtTxt(queryExpression, true);
      int iteration = 0;
      for (String t : term) {
          Term terms = new Term("contents", t);
          Query termQuery = new TermQuery(terms);
          searcher = new IndexSearcher(directory);
          TopDocs topDocs = searcher.search(termQuery, topResults);
```

```
//Finds the top n hits for query.
        HashMap hash = new HashMap();
        for (ScoreDoc match : topDocs.scoreDocs) {
           explanation = searcher.explain(termQuery, match.doc);
           Document doc = searcher.doc(match.doc);
           Explanation[] exp = explanation.getDetails();
           float tf = 0;
           int finalTF = 0;
           for (Explanation e : exp) {
              tf = e.getValue();
              finalTF = Math.round((float) Math.pow(tf, 2));
              if (!resultMap.containsKey(doc.get("id"))) {
                 resultMap.put(doc.get("id"), new ResultDoc(doc, finalTF));
              }
              if (iteration == 0) {
                 setReport("#" + doc.get("id") + " Name: "+doc.get("name")+" Score: " +
match.score + "\n");
                 StringBuilder sb = new StringBuilder();
                 sb.append(String.format("%s \t %d \r\n", doc.get("id"), finalTF).trim());
                 for (int i = 0; i < term.length - 1; i++) {
                    sb.append("\t 0");
                 }
                 sb.append("\r\n");
                 list.add(sb);
```

```
                } else {
                    hash.put(doc.get("id"), finalTF);
                }
                break;
            }
        }
        if (iteration > 0) {
            for (int i = 0; i < list.size() - 1; i++) {//size 50
                String k[] = list.get(i).toString().split("\t"); //key size =5
                //Find same keys an altes (append values)/resize array
                if (hash.containsKey(k[0].trim())) {
                    String old = list.get(i).toString().trim();
                    int lastTad = old.lastIndexOf("\t");
                    list.add(i, String.format("%s \t %s \r\n", old.substring(0, lastTad),
hash.get(k[0].trim())));
                    list.remove(i + 1);
                    hash.remove(k[0].trim());
                }
            }
            Object[] values = hash.values().toArray();
            Object[] keys = hash.keySet().toArray();
            StringBuilder sb = new StringBuilder();
            for (int j = 0; j < keys.length; j++) {
                sb.append(String.format("%s \t ", keys[j]));
```

```
                for (int i = 0; i < iteration; i++) {
                    sb.append("0");
                      sb.append("\t");
                }
                sb.append(values[j]);
                sb.append("\r\n");
            }
            list.add(sb);
        }
        iteration++;
        hash.clear();
        end = System.currentTimeMillis();
    }
    /*Write to report*/
    for (int i = 0; i < list.size(); i++) {
        writeTxt(wr, String.format("%s", list.get(i)));
    }
    closeTxt(wr);
    return resultMap;
}
```

Υλοποίηση του QueryParser

```
/**
    * Υλοποίηση της αναζήτησης ParserSearch.
    * Η μέθοδος υλοποιεί αναζήτηση, scoring και εξαγωγή αποτελεσμάτων
    * σε αρχείο κειμένου.
    * @param queryExpression Ο όρος αναζήτησης
    * @param directory Ο φάκελος που περιέχονται τα indexing αρχεία
    * @param searcher
    * @param explanation Αντικείμενο για όλες τις λεπτομέρειες των αποτελεσμάτων
    * @param topResults Το μέγιστο πλήθος αποτελεσμάτων με το υψηλότερο score
    * @throws IOException
    * @throws org.apache.lucene.queryParser.ParseException
    */

   protected HashMap queryParserSearch(String queryExpression, Directory directory, IndexSearcher searcher,
         Explanation explanation, int topResults) throws IOException, org.apache.lucene.queryParser.ParseException {
      HashMap resultMap = new HashMap<String,ResultDoc>();

      QueryParser parser = new QueryParser(Version.LUCENE_31, "contents",
            new PositionalPorterStopAnalyzer());
      Query query = parser.parse(queryExpression);

      searcher = new IndexSearcher(directory);
      TopDocs topDocs = searcher.search(query, topResults); //Finds the top n hits for query.

      IndexReader ir = searcher.getIndexReader();
      query.rewrite(ir);

      HashSet<Term> termSet = new HashSet<Term>();
      query.extractTerms(termSet);

      String term;
      Document doc;
      TermDocs termDocs = null;
      StringBuilder sb = new StringBuilder();
      PrintWriter wr = createtTxt(queryExpression, true);

      for (ScoreDoc match : topDocs.scoreDocs) {
```

```
        doc = searcher.doc(match.doc);
        sb.append(doc.get("id"));
        //System.out.println(doc.get("id") + " - " +doc.get("name") + " - Score : " +
match.score);

        for (Term termObj : termSet) {
          term = ((Term)termObj).text();

          termDocs = ir.termDocs(new Term("contents", term));
          termDocs.skipTo(match.doc);
          //System.out.println("Term : " + term + " tf=" + termDocs.freq());
          sb.append(" ").append(termDocs.freq());
        }

        writeTxt(wr, sb.toString());
        if (!resultMap.containsKey(doc.get("id"))) {
          resultMap.put(doc.get("id"), new ResultDoc(doc, 0));
        }

        sb.setLength(0);
      }
      termDocs.close();

      closeTxt(wr);
      end = System.currentTimeMillis();

      return resultMap;
    }
```

Υλοποίηση αυτοματοποιημένων ερωτημάτων

```
//Διαδικασία  δημιουργίας ερωτημάτων

public void generateQueries(int queryNum, int termNum, int maxTerms) {
      //get the termsList
      ArrayList<String> theTerms = getTerms(maxTerms);

      String[] queryTerms =  new String[termNum];
      int queryTermsCount;
      String newQuery;

      for (int q=0;q<queryNum;q++) {
        queryTermsCount = 0;

        //init termsList array
        for (int i=0;i<termNum;i++)
          queryTerms[i] = "";

        String newTerm = "";
        boolean found = false;

        if (theTerms == null) return;
        Random rnd = new Random();

        while (queryTermsCount < termNum) {
          // get a random term
          newTerm = theTerms.remove(rnd.nextInt(theTerms.size()));

          queryTerms[queryTermsCount] = newTerm;
          queryTermsCount++;
        }

        // concat termsList into a query
        newQuery = "";
        for (int i=0;i<termNum;i++)
          newQuery += queryTerms[i] + " ";
```

```
        // add new query to query list
        this.queryList.add(newQuery);
    }
}
```