ΠΑΝΕΠΙΣΤΗΜΙΟ ΣΤΕΡΕΑΣ ΕΛΛΑΔΑΣ

ΣΧΟΛΗ: ΕΦΑΡΜΟΣΜΕΝΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ: ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗΝ ΒΙΟΙΑΤΡΙΚΗ

Εξόρυξη πληροφορίας από ιατρικά επιστημονικά κείμενα

Διπλωματική Εργασία
της
Ελένης Κουτσιανικούλη

Επιβλέπων
Ηρακλής Βαρλάμης

Λαμία 2009

Περίληψη

Η επιστήμη της ιατρικής εξελίσσεται συνεχώς με αποτέλεσμα να εμφανίζονται συνεχώς νέα πεδία ενδιαφέροντος, νέες τάσεις και προϊόντα. Παράλληλα αυτά τα πεδία φέρνουν μαζί τους και καινούργιες έννοιες που σχετίζονται μ' αυτά, οι όποιες εμφανίζονται όλο και πιο συχνά στην επιστημονική βιβλιογραφία. Σκοπός της εργασίας είναι η επεξεργασία συλλογών ιατρικών κειμένων, και η εξαγωγή απλών ή σύνθετων ιατρικών όρων από αυτά. Επίσης ένας ακόμη σκοπός της εργασίας είναι η δημιουργία ενός συστήματος Ανάκτησης Πληροφορίας (IR), για την ανάκτηση κειμένων που αναφέρονται στην ιατρική. Η πραγματοποίηση αυτών των δύο στόχων αναμένεται να έχει άμεση εφαρμογή στον τομέα της βιοϊατρικής, και να βοηθήσει τους επιστήμονες στον αυτόματο εντοπισμό ιατρικών όρων σε ιατρικά κείμενα.

Λέξεις Κλειδιά

Συστήματα Ανάκτησης Πληροφορίας, ευρετήριο, δεικτοδότηση, αναζήτηση, ερώτηση, εντοπισμός βιοϊατρικών οντοτήτων

Abstract

The science of medicine is constantly evolving and as a result new areas of interest, new trends and products appear. At the same time, these fields bring new concepts relating to them, which appear more frequently in scientific bibliography. The purpose of this work is the processing of collections of medical texts, and the extraction of simple or complex medical conditions than these. Also a further aim of this work is to create an information retrieval system (IR), to retrieve documents referred to medicine. The realization of these two objectives is expected to have immediate application to the field of biomedicine, and to help scientists in the automatic identification of medical terms in medical texts.

Keywords

Information retrieval system, index, indexing, search, query, identification of biomedical entities

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στο τμήμα Πληροφορική με Εφαρμογές στη Βιοϊατρική του Πανεπιστημίου Στερεάς Ελλάδας, στα πλαίσια των προπτυχιακών σπουδών. Η εκκίνηση της διπλωματικής εργασίας τοποθετείται χρονικά τον Οκτώβριο του 2007 και η ολοκλήρωση της τον Φεβρουάριο του 2009.

Στο σημείο αυτό θα ήθελα να ευχαριστήσω θερμά τον Λέκτορα στο Χαροκόπειο Πανεπιστήμιο και Επιβλέποντα αυτής της παρούσας διπλωματικής εργασίας κ. Ηρακλή Βαρλάμη για την συνεχή βοήθεια και καθοδήγησή του κατά τη διάρκεια της εργασίας, καθώς και για την άψογη συνεργασία που είχαμε. Ακόμα ευχαριστώ την Τριμελή Επιτροπή.

Επίσης θα ήθελα να ευχαριστήσω όλα τα μέλη του Πανεπιστημίου για την άψογη συνεργασία που είχαμε τα χρόνια φοίτησης μου σ' αυτό.

Τέλος, ευχαριστώ την οικογένειά μου για την αγάπη και την υποστήριξή τους σε όλη τη διάρκεια της φοιτητικής μου διαδρομής.

ΠΕΡΙΕΧΟΜΕΝΑ

Κεφάλαιο 1

Εισαγωγή

Εδώ και χιλιάδες χρόνια οι άνθρωποι έχουν αντιληφθεί την αξία της πληροφορίας και την ανάγκη για συλλογή και επεξεργασία της πληροφορίας με στόχο την οικοδόμηση της γνώσης. Με την εμφάνιση των υπολογιστών, ήταν δυνατό να αποθηκευτούν μεγάλα ποσά πληροφοριών και η εύρεση των χρήσιμων πληροφοριών από αυτές τις συλλογές κειμένων έγινε ανάγκη. Ο τομέας της ανάκτησης πληροφοριών (Information Retrieval-IR) γεννήθηκε στη δεκαετία του 1950 εξαιτίας της ανάγκης αυτής. Μπορούμε να δώσουμε ένα απλό ορισμό αυτού ως εξής: Η ανάκτηση πληροφορίας αφορά στην αποτελεσματική διαχείριση απέραντων ποσοτήτων πληροφοριών στις οποίες είναι δύσκολη η ακριβής και ταχεία πρόσβαση. Μια επίδραση αυτής της δυσκολίας είναι ότι οι σχετικές πληροφορίες ενδέχεται να περνούν απαρατήρητες δεδομένου ότι δεν αποκαλύπτονται ποτέ, οδηγώντας έτσι στη συνέχεια σε διπλασιασμό της εργασίας στην προσπάθεια να τις ανακαλύψουμε. Κατά τη διάρκεια των τελευταίων σαράντα χρόνων, ο τομέας έχει αναπτυχθεί αρκετά. Ιδιαίτερα τα τελευταία χρόνια ο συνδυασμός της ανάπτυξης και της ολοένα αυξανόμενης χρήσης του Παγκόσμιου Ιστού (Word Wide Web – WWW) και η αύξηση των ηλεκτρονικών συσκευών αποθήκευσης έχει ως συνέπεια την χρήση κάποιου εργαλείου για την παρακολούθηση του τεράστιου όγκου πληροφοριών που είναι διαθέσιμος στην κοινωνία των πληροφοριών. Ένα τέτοιο εργαλείο είναι και η ανάκτηση πληροφοριών. Διάφορα συστήματα ανάκτησης πληροφοριών (Information Retrieval – IR) χρησιμοποιούνται σε καθημερινή βάση από διάφορες ομάδες χρηστών. Σε λιγότερο από δέκα έτη, τα συστήματα ανάκτησης έχουν μετατραπεί από εσωτερικά συστήματα για την χρήση μόνο από μια μικρή κοινότητα ερευνητών σε μια ουσιαστική μέθοδο της ανάκτησης πληροφορίας για εκατομμύρια ανθρώπους.(Oppenheiem, Moris, Mcknight, & Lowley, 2000)

1.1 Ορισμός της Ανάκτησης Πληροφορίας

Ένας άλλος ορισμός της ανάκτησης πληροφορίας που θα μπορούσε κάποιος να δώσει είναι ο εξής: "Η ανάκτηση των πληροφοριών (Information Retrieval – IR) είναι η επιστήμη της έρευνας για πληροφορίες σ' έγγραφα, για έγγραφα, για μετά-δεδομένα (metadata) τα οποία περιγράφουν έγγραφα, ή έρευνας σε βάσεις δεδομένων, είτε σε αυτόνομες σχεσιακές βάσεις δεδομένων, είτε σε δικτυακές βάσεις δεδομένων, όπως είναι ο Παγκόσμιος Ιστός (World Wide Web). Επίσης η ανάκτηση πληροφοριών ασχολείται με την απεικόνιση, αποθήκευση, οργάνωση και πρόσβαση σε αντικείμενα πληροφορίας." (Korfhage, 1997). Η απεικόνιση και η οργάνωση των αντικειμένων εξασφαλίζουν στο χρήστη εύκολη πρόσβαση στις πληροφορίες που χρειάζεται. Όμως, ο χαρακτηρισμός των πληροφοριών που χρειάζεται ένας χρήστης, έτσι ώστε να είναι κατανοητές από το σύστημα, δεν είναι ένα απλό πρόβλημα, γι' αυτό το λόγο ο χρήστης θα πρέπει πρώτα να 'μεταφράζει' τις πληροφορίες που χρειάζεται σε μια *ερώτηση*, η οποία μπορεί να επεξεργαστεί από τις μηχανές αναζήτησης (ή τα συστήματα Ανάκτησης Πληροφοριών). (Baeza-Yates & Ribero-Neto, 1999) Συνηθέστερο είναι να χρησιμοποιούνται λέξεις κλειδιά, οι οποίες συνοψίζουν την περιγραφή των πληροφοριών που επιθυμεί ο χρήστης. Δίνοντας ο χρήστης την ερώτηση, στόχος του συστήματος IR είναι η ανάκτηση πληροφοριών με βάση τις λέξεις κλειδιά που περιέχονται στην ερώτηση, οι οποίες μπορεί να είναι χρήσιμες ή σχετικές με τις πληροφορίες που επιθυμεί ο χρήστης.

1.1.1 Μεταδεδομένα

Οι περισσότερες συλλογές έγγραφων και κειμένων συνδέονται με αυτό που είναι γνωστό ως μεταδεδομένα. Τα μεταδεδομένα είναι πληροφορίες για την οργάνωση των δεδομένων, των διαφορετικών περιοχών δεδομένων, και της σύνδεσης μεταξύ αυτών των δύο. Εν συντομία, τα μεταδεδομένα είναι 'δεδομένα για δεδομένα'. Τα μεταδεδομένα (μερικές φορές γράφονται και 'meta data') χρησιμοποιούνται για να διευκολύνουν την κατανόηση, τη χρήση και τη διαχείριση των δεδομένων· απαιτούνται για την αποτελεσματική διαχείριση των δεδομένων και διαφοροποιούνται ανάλογα με τον τύπο των δεδομένων και το πλαίσιο χρήσης τους. Για παράδειγμα, σε ένα σύστημα διαχείρισης βάσης δεδομένων, το σχήμα καθορίζει

μερικά από τα μεταδεδομένα, όπως το όνομα της συσχέτισης, τα πεδία ή τα χαρακτηριστικά κάθε σχέσης, το πεδίο κάθε χαρακτηριστικού, κλπ.

Υπάρχουν κάποια είδη μεταδεδομένων όπως για παράδειγμα τα Περιγραφικά Μεταδεδομένα, τα οποία είναι μεταδεδομένα που εξάγονται από την έννοια του εγγράφου, και αναφέρονται περισσότερο στο πως αυτά δημιουργούνται. Ένας άλλος τύπος μεταδεδομένων χαρακτηρίζει το περιεχόμενο το οποίο μπορεί να βρεθεί εντός των περιεχομένων του εγγράφου και ονομάζονται Σημασιολογικά Μεταδεδομένα. Αυτά συνδέονται με ένα μεγάλο αριθμό εγγράφων. Για παράδειγμα, στα βιοϊατρικά άρθρα που εμφανίζονται μέσα στο σύστημα της MEDLINE ορίζεται το θέμα των μεταδεδομένων σχετικά με την ασθένεια, την ανατομία, τα φαρμακευτικά είδη, και άλλους ορισμούς.

Με την αύξηση των δεδομένων στον Ιστό, υπάρχουν πολλές πρωτοβουλίες ώστε να προστεθούν οι πληροφορίες των μεταδεδομένων στα έγγραφα του Ιστού. Στον Ιστό, τα μεταδεδομένα μπορούν να χρησιμοποιηθούν για πολλούς λόγους. Μερικοί από αυτούς είναι οι κατάλογοι (ο Bib TeX είναι ένα δημοφιλές σχήμα για αυτήν την περίπτωση), η εκτίμηση του περιεχομένου (παραδείγματος χάριν, για την προστασία των παιδιών από την ανάγνωση κάποιων τύπων εγγράφων), τα πνευματικά δικαιώματα ιδιοκτησίας, οι ψηφιακές υπογραφές (για την επικύρωση), οι εφαρμογές στο ηλεκτρονικό εμπόριο, κλπ. Τα νέα πρότυπα για τα μεταδεδομένα Ιστού είναι τα ‘πλαίσια περιγραφής των πόρων’ (Resource Description Framework – RDF), τα οποία παρέχουν τη διαλειτουργικότητα μεταξύ των εφαρμογών. Αυτό το πλαίσιο επιτρέπει την περιγραφή των πόρων του Ιστού για να διευκολύνει την αυτοματοποιημένη επεξεργασία των πληροφοριών. (Baeza-Yates & Ribero-Neto, 1999)

1.2 Η ιστορία της δημιουργίας του τομέα της Ανάκτησης Πληροφορίας

Η ιδέα της χρήσης των υπολογιστών για αναζήτηση των κατάλληλων πληροφοριών έγινε δημοφιλής στο άρθρο “As We May Think” του Vannevar Bush το 1945. Οι πρώτες υλοποιήσεις των συστημάτων ανάκτησης πληροφορίας παρουσιάστηκαν τις δεκαετίες του ’50 και ’60. Το 1957 περιγράφει μια από τις σημαντικότερες μεθόδους από τον H.P. Luhn. Αυτό που πρότεινε ο Luhn με απλά

λόγια είναι να χρησιμοποιηθούν οι λέξεις ως μονάδες για την τοποθέτηση δεικτών στα έγγραφα και την μέτρηση των επικαλυπτόμενων λέξεων ως κριτήριο για την ανάκτηση.

Οι εξελίξεις που παρατηρήθηκαν κατά τις δεκαετίες του ’70 και του ’80 είχαν ως αποτέλεσμα την δημιουργία νέων συστημάτων, τα οποία αποδείχθηκαν αποτελεσματικά για τις μικρές συλλογές κειμένων που ήταν διαθέσιμες στους ερευνητές εκείνη την περίοδο. Καθώς υπήρχε έλλειψη διαθεσιμότητας μεγάλων συλλογών κειμένων ήταν αδύνατο να απαντήσουν στο ερώτημα εάν αυτά τα συστήματα θα ήταν αποτελεσματικά και σε μεγάλες συλλογές. Αυτό άλλαξε το 1992. (Singhal, 2001)

Το 1992 το Τμήμα Άμυνας των Ηνωμένων Πολιτειών (US Department of Defense), με το Εθνικό Ινστιτούτο Προτύπων και Τεχνολογιών (National Institute of Standards and Technology – NIST), και με χορηγό το Συνέδριο Ανάκτησης Κειμένου (Text Retrieval Conference – TREC) συνεργάστηκαν για το τμήμα του προγράμματος κειμένων TIPSTER. Ο σκοπός αυτού ήταν να εξετάσει την κοινότητα ανάκτησης πληροφορίας, καλύπτοντας παράλληλα την ανάγκη για την υποδομή η οποία χρειάζονταν για την αποτίμηση των μεθοδολογιών ανάκτησης κειμένων σε μια μεγάλη συλλογή κειμένων. Αυτή η καταλυτική έρευνα για τις μεθόδους μετρά το τεράστιο σύνολο των εγγράφων. Η εισαγωγή των μηχανών αναζήτησης στον Ιστό αύξησε την ανάγκη για μεγάλης κλίμακας συστημάτων ανάκτησης.

1.3 Η Ανάκτησης Της Πληροφορίας (IR)

Τα τελευταία 20 χρόνια, ο τομέας της ανάκτησης πληροφορίας έχει αυξηθεί και χρησιμοποιείται και για άλλους σκοπούς πέρα από τους βασικούς που είναι η δημιουργία ευρετηρίων με τα κείμενα και η αναζήτηση χρήσιμων έγγραφων σε μια συλλογή. Πλέον η έρευνα στα IR συμπεριλαμβάνει μοντέλα, ταξινομημένα και κατηγοριοποιημένα έγγραφα, αρχιτεκτονικά συστήματα, αλληλεπιδράσεις χρηστών, οπτικοποίηση δεδομένων, φιλτραρίσματα, και άλλα. Παρά την μεγάλη ανάπτυξη τους, μέχρι πρόσφατα, τα IR παρουσιαζόταν ως μια περιορισμένη περιοχή ενδιαφέροντος, κυρίως για τους βιβλιοθηκονόμους. Αυτή η προκατειλημμένη ιδέα επικρατούσε για πολλά χρόνια, παρά την ταχεία διάδοση, μεταξύ των χρηστών των σύγχρονων προσωπικών υπολογιστών, των IR εργαλείων για τις εφαρμογές των

πολυμέσων και υπερκειμένων. Αυτές οι αντιλήψεις όμως άλλαξαν στις αρχές της δεκαετίας του ’90 με την εισαγωγή του Παγκόσμιου Ιστού. (Baeza-Yates & Ribero-Neto, 1999)

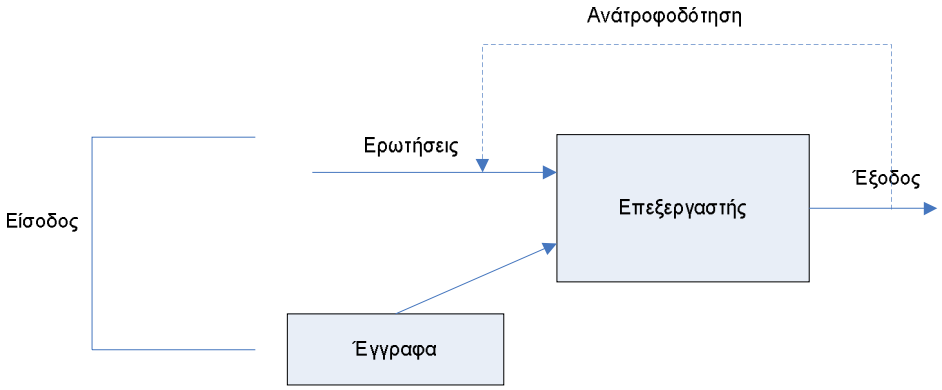
Ο Ιστός έχει γίνει ένας γενικός θησαυρός της ανθρώπινης γνώσης και κουλτούρας γεγονός που επέτρεψε πρωτοφανής ανταλλαγή ιδεών και πληροφοριών σε μια κλίμακα που δεν υπήρχε ποτέ πριν. Η επιτυχία του βασίζεται στην αρχή της καθιερωμένης αλληλεπίδρασης με το χρήστη η οποία είναι πάντα η ίδια, και ανεξάρτητη του υπολογιστικού περιβάλλοντος που χρησιμοποιείται για την εκτέλεση της διεπαφής. Επιπλέον, οποιοσδήποτε χρήστης μπορεί να δημιουργήσει τα δικά του έγγραφα Ιστού και να φτιάξει ευρετήρια σε άλλα έγγραφα του Ιστού χωρίς περιορισμούς. Αυτή είναι μια σημαντική πτυχή διότι μετατρέπει το Διαδίκτυο σε ένα νέο μέσο δημοσίευσης προσιτό σε όλους. Αυτός ο τομέας, χωρίς σύνορα, ελκύει την τεράστια προσοχή των εκατομμυρίων ανθρώπων από την αρχή της δημιουργίας του.

Όμως, παρά την μεγάλη επιτυχία του, ο Ιστός έχει εισάγει καινούργια προβλήματα. Η εύρεση χρήσιμων πληροφοριών στον Ιστό είναι συχνά μια βαρετή και δύσκολη διαδικασία. Για παράδειγμα, για την ικανοποίηση των απαραίτητων πληροφοριών που χρειάζεται ο χρήστης, μπορεί να χρειαστεί να πλοηγηθεί σε θέσεις των συνδέσμων του Ιστού για την εύρεση των πληροφοριών που τον ενδιαφέρουν. Για τους απλοϊκούς χρήστες, το πρόβλημα γίνεται δυσκολότερο, με πιθανό αποτέλεσμα να αποθαρρύνει εντελώς όλες τις προσπάθειες τους. Το κύριο εμπόδιο είναι η απουσία ενός καλά ορισμένου μοντέλου δεδομένων για τον Ιστό, το οποίο υποδηλώνει ότι οι έννοιες και η δομή των πληροφοριών είναι συχνά χαμηλότερης ποιότητας. Αυτές οι δυσκολίες ελκύουν το ανανεωμένο ενδιαφέρον για τα IR και τις τεχνικές τους, με σκοπό να προταθούν λύσεις.

Στην Εικόνα 1.1 φαίνονται οι τρεις συνιστώσες ενός συστήματος IR: είσοδος, επεξεργαστής, και έξοδος. Αυτή η τριχοτόμηση των στοιχείων μπορεί να φαίνεται κάπως κοινότυπη, αλλά είναι ιδιαίτερα εξυπηρετική για την κατανόηση του συστήματος. (Rijsbergen, 1979)

Αρχικά θα αναλύσουμε την διαδικασία της εισόδου. Το κύριο πρόβλημα που υπάρχει σ’ αυτή την φάση είναι η εξασφάλιση μιας κατάλληλης απεικόνισης για κάθε έγγραφο και ερώτηση για τον υπολογιστή που χρησιμοποιείται. Σ’ αυτό που πρέπει να εστιάσουμε είναι το γεγονός ότι τα περισσότερα συστήματα ανάκτησης που βασίζονται σε υπολογιστές αποθηκεύουν μόνο μια απεικόνιση των εγγράφων (ή ερωτήσεων)· αυτό σημαίνει ότι το κείμενο του εγγράφου χάνεται όταν θα έχει

ολοκληρωθεί η επεξεργασία για το σκοπό της δημιουργίας της εκπροσώπησης του. Παραδείγματος χάριν, ένας *αντιπρόσωπος εγγράφων* θα μπορούσε να χρησιμοποιηθεί, και θα περιλάμβανε ένα κατάλογο εξαγόμενων λέξεων που θεωρήθηκαν σημαντικές. Αντί να έχουμε τον υπολογιστή για την επεξεργασία της φυσικής γλώσσας, μια εναλλακτική προσέγγιση είναι να υπάρχει μια τεχνητή γλώσσα μέσα στην οποία όλες οι ερωτήσεις και τα έγγραφα θα μπορούν να διατυπωθούν. Υπάρχουν στοιχεία που δείχνουν ότι αυτό μπορεί να είναι αποτελεσματικό. Βέβαια προϋποθέτει ότι ο χρήστης είναι πρόθυμος να διδαχθεί για να εκφράσει στη γλώσσα αυτή την πληροφορία που χρειάζεται.



Εικόνα 1.1 Ένα τυπικό σύστημα IR[a]

Όταν το σύστημα ανάκτησης είναι σε απευθείας σύνδεση (on-line), είναι δυνατό για το χρήστη να αλλάξει το αίτημά του κατά τη διάρκεια μιας συνόδου αναζήτησης λαμβάνοντας υπόψη ένα δείγμα ανακτημένων εγγράφων, με αυτόν τον τρόπο, αναμένεται βελτίωση στο επόμενο τρέξιμο της ανάκτησης. Μια τέτοια διαδικασία αναφέρεται συνήθως ως *ανατροφοδότηση (ανάδραση)*. Ένα παράδειγμα ενός περίπλοκου συστήματος σε απευθείας σύνδεση ανάκτησης είναι το σύστημα MEDLINE.

Ο επεξεργαστής αποτελεί το δεύτερο τμήματα του συστήματος, και είναι αυτό το τμήμα που χρησιμοποιείται για τη διαδικασία της ανάκτησης. Η διαδικασία μπορεί να περιλάβει τη δόμηση των πληροφοριών με κάποιο κατάλληλο τρόπο, όπως η ταξινόμηση του. Θα περιλαμβάνει επίσης την εκτέλεση της πραγματικής λειτουργίας ανάκτησης, δηλαδή την εκτέλεση της στρατηγικής αναζήτησης ως απάντηση σε μια ερώτηση. Στο διάγραμμα (στην Εικόνα 1.1), τα έγγραφα έχουν τοποθετηθεί σε ένα

[a] Πηγή εικόνας από το κεφάλαιο 1 του βιβλίου ‘Information Retrieval’ του Keith van Rijsbergen (1979).

χωριστό πεδίο για να υπογραμμιστεί το γεγονός ότι δεν αποτελούν απλά μια είσοδο αλλά μπορούν να χρησιμοποιηθούν κατά τη διάρκεια της διαδικασίας ανάκτησης με τέτοιο τρόπο ώστε η δομή τους να φαίνεται πιο σωστά ως μέρος της διαδικασίας ανάκτησης.

Τέλος, υπάρχει το στάδιο της παραγωγής, η οποία συνήθως αποτελείται από ένα σύνολο παραπομπών ή αριθμών εγγράφων. Σε ένα λειτουργικό σύστημα η διαδικασία τελειώνει εδώ. Εντούτοις, σε ένα πειραματικό σύστημα πρέπει να γίνει και αξιολόγηση.

Αν και η ανάκτηση πληροφοριών μπορεί να υποδιαιρεθεί με πολλούς τρόπους, φαίνεται ότι υπάρχουν τρεις κύριοι τομείς της έρευνας που μεταξύ τους αποτελούν ένα σημαντικό μέρος του θέματος. Αυτοί οι τομείς είναι: η *ανάλυση περιεχομένου*, οι *δομές πληροφοριών*, και η *αξιολόγηση*. Εν συντομία ο πρώτος όρος ασχολείται με την περιγραφή του περιεχομένου των εγγράφων σε μορφή κατάλληλη για τη μηχανογραφική επεξεργασία· ο δεύτερος στην αξιοποίηση των σχέσεων μεταξύ των εγγράφων για την βελτίωση της αποδοτικότητας και της αποτελεσματικότητας των στρατηγικών ανάκτησης· και ο τρίτος στη μέτρηση της αποτελεσματικότητας της ανάκτησης.

Κατά καιρούς έχουν χρησιμοποιηθεί από διάφορους επιστήμονες στο χώρο αυτό καταγραφή των συχνοτήτων των λέξεων στο κείμενο εγγράφων για να καθοριστούν ποιες λέξεις είναι αρκετά σημαντικές ώστε να αντιπροσωπεύσουν ή να χαρακτηρίσουν το έγγραφο στον υπολογιστή. Κατά συνέπεια ένας κατάλογος αυτών, που ονομάζονται *λέξεις κλειδιά*, παράγονται για κάθε έγγραφο. Επιπλέον η συχνότητα εμφάνισης αυτών των λέξεων στο σώμα του κειμένου θα μπορούσε επίσης να χρησιμοποιηθεί για να εξαχθεί ο βαθμός μεγάλης σημασίας. Αυτό παρείχε ένα απλό σχέδιο στάθμισης για τις λέξεις κλειδιά σε κάθε κατάλογο και τίθεται διαθέσιμο ένα αντιπροσωπευτικό έγγραφο υπό μορφή “σταθμισμένης περιγραφής λέξης κλειδιού”.

Σε αυτό το σημείο, ίσως θα ήταν χρήσιμο να αναλύσουμε την χρήση της λέξης ‘κλειδί’. Είναι πλέον κοινή πρακτική στη λογοτεχνία της IR να αναφέρονται στα περιγραφικά στοιχεία που εξάγονται από το κείμενο ως *λέξεις κλειδιά* ή *όροι*. Τέτοια στοιχεία είναι συχνά το αποτέλεσμα κάποιων διαδικασιών όπως, παραδείγματος χάριν, η συλλογή των διαφορετικών μορφολογικών παραλλαγών της ίδιας λέξης. Στην συνέχεια της εργασίας αυτής οι όροι ‘κλειδί’ και ‘όρος’ θα χρησιμοποιούνται εναλλακτικά.

Η *δομή του όρου πληροφοριών* καλύπτει συγκεκριμένα μια λογική οργάνωση των πληροφοριών με σκοπό την ανάκτηση πληροφοριών. Η ανάπτυξη στις δομές πληροφοριών είναι αρκετά πρόσφατη. Ο τομέας αυτός της ανάκτησης πληροφοριών άργησε να αναπτυχθεί κυρίως γιατί οι ερευνητές άργησαν να συνειδητοποιήσουν ότι οι υπολογιστές δεν θα έδιναν έναν αποδεκτό χρόνο ανάκτησης για ένα μεγάλο έγγραφο, εκτός και αν κάποια λογική δομή επιβάλλονταν σε αυτό. Η βραδύτητα της αναγνώρισης και υιοθέτησης των νέων τεχνικών οφείλεται κυρίως στην ανεπάρκεια των πειραματικών στοιχείων που υποστηρίζουν. Τα προηγούμενα πειράματα με τα συστήματα ανάκτησης εγγράφων υιοθέτησαν συνήθως μια οργάνωση σειριακών αρχείων, η οποία ήταν αποδοτική όταν υποβλήθηκε σε επεξεργασία ταυτόχρονα ένας αρκετά μεγάλος αριθμός ερωτήσεων με έναν ομαδοποιημένο τρόπο, αποδείχθηκε ανεπαρκής όταν κάθε ερώτηση απαιτούσε μια σύντομη απάντηση σε πραγματικό χρόνο. Η δημοφιλής οργάνωση που υιοθετήθηκε αντί αυτού ήταν το ανεστραμμένο αρχείο. Από μερικούς, βέβαια, αυτή η πρόταση θεωρήθηκε ως περιοριστική.(Rijsbergen K., 1979)

Η *αξιολόγηση* των συστημάτων ανάκτησης έχει αποδειχθεί εξαιρετικά δύσκολη. Παρά το εξαιρετικό πρωτοποριακό έργο των Cleverdon et al. (1966) στον τομέα αυτό, και παρά τα πολυάριθμα μέτρα της αποτελεσματικότητας που έχουν προταθεί δεν έχει προκύψει μια γενική θεωρία της αξιολόγησης, ακόμη. Σήμερα, για τη μέτρηση της αποτελεσματικότητας των συστημάτων, χρησιμοποιούνται κάποια μαθηματικά μέτρα μέτρησης της καταλληλότητας των εγγράφων που ανακτώνται, όπως είναι η *ανάκληση* και η *ακρίβεια*, τα οποία θα αναπτυχθούν παρακάτω.

1.4 Τι ενέργειες πρέπει να κάνει ο χρήστης

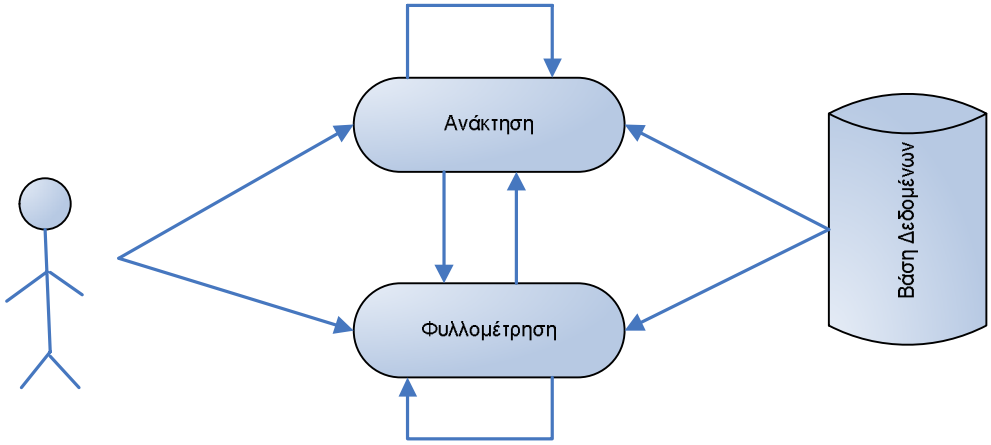
Ο χρήστης του συστήματος ανάκτησης πρέπει να μεταφράσει τις πληροφορίες που χρειάζεται σε μια ερώτηση στην γλώσσα που προβλέπει το σύστημα. Σε ένα σύστημα ανάκτησης πληροφορίας, αυτό που συνεπάγεται κανονικά είναι ο ορισμός ενός συνόλου λέξεων που μεταφέρουν την σημασιολογία των πληροφοριών που χρειάζεται ο χρήστης. Αυτό που μπορεί κανείς να πει σ’ αυτή την περίπτωση είναι ότι ο χρήστης ψάχνει για χρήσιμες πληροφορίες εκτελώντας μια εργασία ανάκτησης.

Έστω ότι κάποιος χρήστης ενδιαφέρεται για κάποιο αντικείμενο το οποίο είναι είτε κακώς καθορισμένο είτε καθορισμένο ευρύτερα. Για παράδειγμα, ο

χρήστης μπορεί να χρησιμοποιήσει μια διαδραστική διεπαφή απλά για να κοιτάξει σε μια συλλογή εγγράφων που αναφέρονται σε αγώνες αυτοκινήτων γενικά. Σε αυτή την περίπτωση, ο χρήστης μπορεί να χρησιμοποιεί μια αλληλεπιδραστική διασύνδεση για να κοιτάξει απλά σε μια συλλογή έγγραφων που αφορούν σε αγώνες αυτοκινήτων. Για παράδειγμα, μπορεί να βρει ενδιαφέροντα έγγραφα για τον αγώνα της Formula 1, για τη βιομηχανία αυτοκινήτων, ή για το "24 Hours of Le Mans". Επιπλέον, ενώ διαβάζει το "24 Hours of Le Mans", μπορεί να στρέψει την προσοχή του σε έγγραφα τα οποία παρέχουν οδηγίες στο Le Mans και, από εκεί, σε έγγραφα τα οποία καλύπτουν ως θέμα τον τουρισμό στην Γαλλία. Σ' αυτή την περίπτωση λέμε ότι ο χρήστης φυλλομέτρησε τα έγγραφα της συλλογής, δηλαδή έψαξε στο σύνολο των πληροφοριών της συλλογής, και δεν αναζήτησε. Βέβαια αυτή παραμένει μια διαδικασία ανάκτησης πληροφοριών, αλλά οι κύριοι στόχοι αυτής δεν καθορίζονται σαφώς από την αρχή και ο σκοπός της μπορεί να αλλάξει κατά τη διάρκεια της αλληλεπίδρασης με το σύστημα. (Baeza-Yates & Ribero-Neto, 1999)

Μια εργασία χρηστών σε ένα σύστημα ανάκτησης μπορεί να έχει ένα από τους δύο τύπους: πληροφορίες ή δεδομένα *ανάκτησης* και *φυλλομέτρηση*. Τα κλασσικά συστήματα ανάκτησης πληροφορίας κανονικά επιτρέπουν την ανάκτηση πληροφοριών ή δεδομένων. Τα συστήματα υπέρ-κειμένων συνήθως συντονίζονται για γρήγορη φυλλομέτρηση. Οι σύγχρονες ψηφιακές βιβλιοθήκες και οι αλληλεπιδράσεις του Ιστού προσπαθούν να συνδυάσουν αυτές τις εργασίες για να εξασφαλίσουν βελτιωμένες ικανότητες ανάκτησης.

Η Εικόνα 1.2 απεικονίζει την αλληλεπίδραση των χρηστών δια μέσω των διαφορετικών εργασιών που ορίστηκαν παραπάνω. Η ανάκτηση πληροφορίας και δεδομένων συχνά εξασφαλίζετε από τα πιο σύγχρονα συστήματα ανάκτησης πληροφορίας (όπως οι διασυνδέσεις του Ιστού). Επιπλέον, αυτά τα συστήματα μπορούν να εξασφαλίσουν κάποια μορφή φυλλομέτρησης.



Εικόνα 1.2 Η αλληλεπίδραση του χρήστη με το σύστημα IR διαμέσου των διαφορετικών εργασιών[b]

1.5 Λογική όψη του εγγράφου

Συχνά τα έγγραφα σε μια συλλογή αναπαρίστανται από ένα σύνολο δεικτοδοτούμενων ορών ή λέξεων κλειδιών. Αυτές οι λέξεις κλειδιά μπορούν να εξαχθούν αμέσως από το κείμενο του εγγράφου ή μπορούν να καθοριστούν από ένα ανθρώπινο υποκείμενο (όπως συνήθως συμβαίνει στην επιστήμη των πληροφοριών). Δεν υπάρχει κάποια διαφορά όσο αναφορά στο τρόπο δημιουργίας τους, δηλαδή αν αυτές οι αντιπροσωπευτικές λέξεις κλειδιά παράγονται αυτόματα ή αν παράγονται από έναν ειδικό, παρέχουν μια *λογική όψη του εγγράφου*. (Baeza-Yates & Ribero-Neto, 1999)

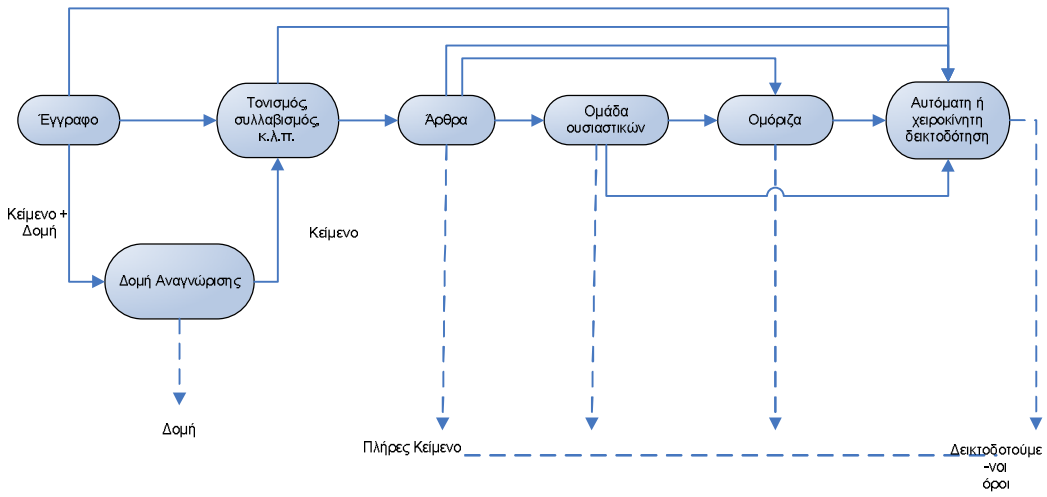
Οι σύγχρονοι υπολογιστές αναπαριστούν ένα έγγραφο με ολόκληρο το σύνολο των λέξεων του. Σ' αυτή την περίπτωση, λέμε ότι το σύστημα ανάκτησης υιοθετεί ένα ολόκληρο κείμενο ως λογική όψη (ή απεικόνιση) των εγγράφων. Σε πολύ μεγάλες συλλογές, ωστόσο, οι σύγχρονοι υπολογιστές μπορούν να μειώσουν το σύνολο των αντιπροσωπευτικών λέξεων κλειδιών. Αυτό μπορεί να επιτευχθεί με τη εξάλειψη των 'κοινών λέξεων' (όπως είναι τα άρθρα και οι σύνδεσμοι), την χρήση των ομόριζων (το οποίο αντιστοιχεί την κάθε λέξη ξεχωριστά στην κοινή γραμματική τους ρίζα), και την αναγνώριση των ομάδων από ουσιαστικά (ενώ αποβάλλει τα επίθετα, τα επιρρήματα, και τα ρήματα). Επιπλέον, μπορεί να χρησιμοποιηθεί και η συμπίεση. Αυτή η λειτουργία καλείται *εφαρμογή κειμένων* (ή μετασχηματισμός). Η

[b] Πηγή της εικόνας από το κεφάλαιο 1 του βιβλίου 'Modern Information Retrieval' των Ricardo Baeza-Yates και Berthier Ribeiro-Neto, (1999).

εφαρμογή κειμένου μειώνει την πολυπλοκότητα της αντιπροσώπευσης των εγγράφων και επιτρέπει μετακίνηση της λογική όψη από ένα πλήρη κείμενο προς ενός συνόλου *όρων δεικτών*.

Το πλήρες κείμενο είναι σαφώς η πιο ολοκληρωμένη λογική όψη ενός εγγράφου αλλά η χρήση της υπονοεί συνήθως υψηλότερες υπολογιστικές δαπάνες. Ένα μικρό σύνολο κατηγοριών (που παράγονται από έναν ειδικό) παρέχει την πιο συνοπτική λογική όψη ενός εγγράφου αλλά η χρήση της μπορεί να οδηγήσει σε ανάκτηση κακής ποιότητας. Διάφορες ενδιάμεσες λογικές όψεις ενός εγγράφου μπορούν να υιοθετηθούν από ένα σύστημα ανάκτησης πληροφοριών όπως απεικονίζεται στην Εικόνα 1.3. Εκτός από την υιοθέτηση οποιωνδήποτε από την ενδιάμεση αντιπροσώπευση, το σύστημα ανάκτησης επίσης μπορεί να αναγνωρίσει εσωτερικές δομές που παρουσιάζονται σε ένα έγγραφο (π.χ., κεφάλαια, τμήματα, υποενότητες, κλπ.).

Από την Εικόνα 1.3 μπορεί κανείς να καταλάβει την διαδικασία αντιπροσώπευσης μιας λογικής όψης του κειμένου από μια πλήρη απεικόνιση του κειμένου σε μια υψηλού επιπέδου απεικόνιση, η οποία έχει καθοριστεί από ένα ανθρώπινο υποκείμενο.



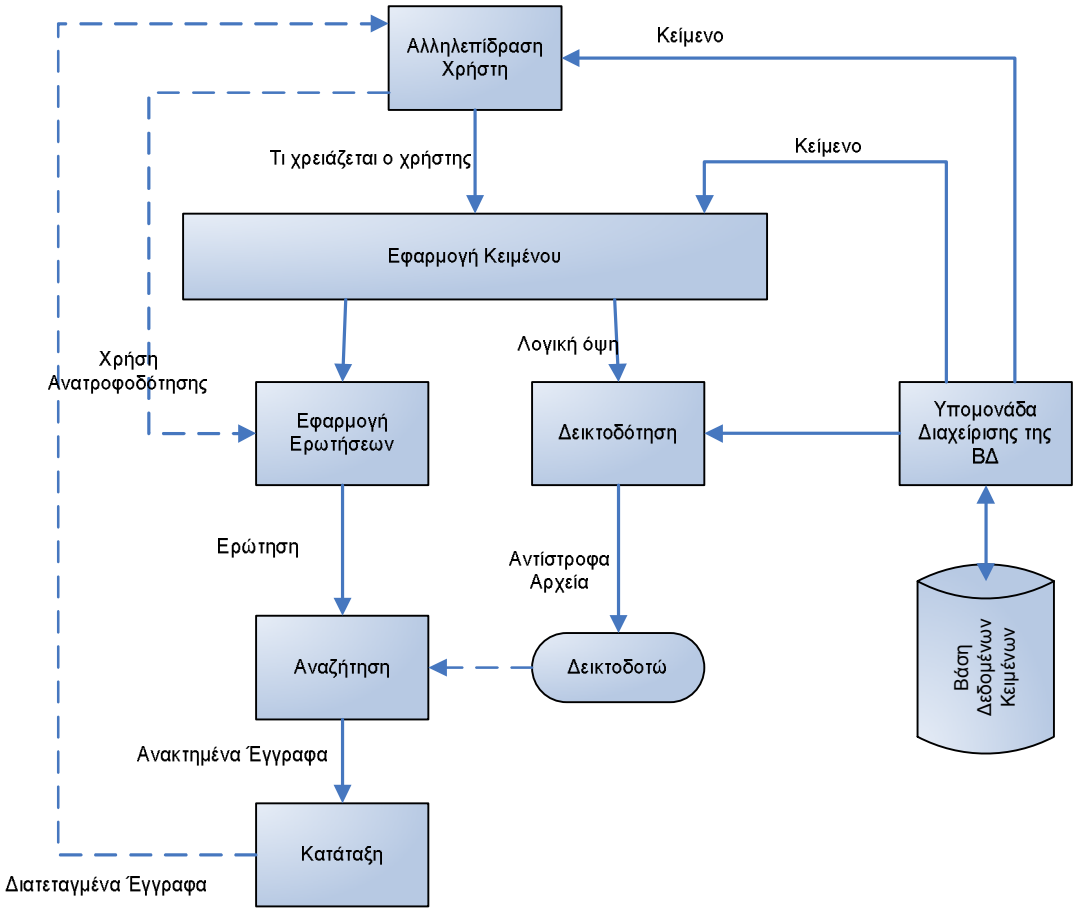
Εικόνα 1.3 Λογική όψη ενός εγγράφου: από ένα πλήρες κείμενο σε ένα σύνολο έκφρασης δεικτών[c]

[c] Πηγή της εικόνας από το κεφάλαιο 1 του βιβλίου 'Modern Information Retrieval' των Ricardo Baeza-Yates και Berthier Ribeiro-Neto (1999).

1.6 Η διαδικασία της Ανάκτησης

Σε αυτό το σημείο, μπορεί να μελετηθεί με λεπτομέρεια η όψη της διαδικασίας ανάκτησης. Μια τέτοια διαδικασία ερμηνεύεται με όρους από τα συστατικά των υποδιαδικασιών.

Για την περιγραφή της διαδικασίας ανάκτησης, χρησιμοποιείται μια απλή και γενική αρχιτεκτονική λογισμικού όπως φαίνεται στην Εικόνα 1.4. Καταρχήν, πριν ακόμη ξεκινήσει η διαδικασία ανάκτησης, είναι απαραίτητο να καθοριστεί η βάση δεδομένων των κειμένων. Αυτό γίνεται συνήθως από το διευθυντή της βάσης δεδομένων, όπου διευκρινίζονται τα εξής: (α) τα έγγραφα που χρησιμοποιούνται, (β) οι διαδικασίες που εκτελούνται στο κείμενο, και (γ) το πρότυπο κειμένων (δηλαδή, η δομή των κειμένων και ποια στοιχεία μπορούν να ανακτηθούν). Οι διαδικασίες κειμένων μετασχηματίζουν τα αρχικά έγγραφα και παράγουν μια λογική όψη τους. (Baeza-Yates & Ribero-Neto, 1999)



Εικόνα 1.4 Η διαδικασία της ανάκτησης πληροφορίας[d]

[d] Πηγή της εικόνας από το κεφάλαιο 1 του βιβλίου 'Modern Information Retrieval' των Ricardo Baeza-Yates και Berthier Ribeiro-Neto (1999).

1.7 Η Ανάκτηση Πληροφορίας και η Ανάκτηση Δεδομένων

Συχνά παρατηρείται μια σύγχυση μεταξύ των διάφορων τύπων ανάκτησης όπως είναι η ανάκτηση των δεδομένων, των εγγράφων, των πληροφοριών και των (αποσπασμάτων) κειμένων. Όμως το καθένα από αυτά αποτελείται από τα δικά του σώματα βιβλιοθηκών, θεωρίες, πράξεις και τεχνολογίες. Η ανάκτηση πληροφοριών (Information Retrieval-IR) περιλαμβάνει τις δικές της αρχές, βασιζόμενη στις επιστήμες: της πληροφορικής, των μαθηματικών, της βιβλιοθήκης, της πληροφορίας, της γνωστικής ψυχολογίας, της γλωσσολογίας, της στατιστικής και της φυσικής.

Η ανάκτηση δεδομένων (Data Retrieval - DR), στο περιβάλλον ενός συστήματος IR, υπάρχει κυρίως για να καθορίζει ποια έγγραφα μιας συλλογής περιέχουν τις λέξεις κλειδιά στην ερώτηση του χρήστη, ενώ παρατηρείτε συχνά ότι δεν είναι αρκετό για να ικανοποιήσει τις πληροφορίες που χρειάζεται ο χρήστης. Στην πραγματικότητα, οι χρήστες των συστημάτων IR ενδιαφέρονται περισσότερο για τις ανακτημένες πληροφορίες για ένα θέμα από ότι για τα ανακτημένα δεδομένα που θα μπορούσαν να ικανοποιούν την ερώτηση που έθεσαν. Η γλώσσα ανάκτησης δεδομένων σκοπεύει στην ανάκτηση όλων των αντικειμένων τα οποία ικανοποιούν ξεκάθαρα τους ορισμένους όρους, όπως σε μια κανονική έκφραση ή σε μια έκφραση σχέσης στην άλγεβρα. Γι' αυτό το λόγο, για ένα σύστημα ανάκτησης δεδομένων, ένα μονό λανθασμένο αντικείμενο μεταξύ χιλίων ανακτημένων αντικειμένων δηλώνει την ολοκληρωτική αποτυχία του συστήματος. Για ένα σύστημα ανάκτησης πληροφορίας, όμως, τα ανακτημένα αντικείμενα μπορεί να είναι ανακριβή και με μικρά λάθη και να μην επηρεάζουν το σύστημα. Ο κύριος λόγος για αυτή την διαφορά είναι ότι η ανάκτηση πληροφορίας συνήθως συναλλάσετε με το κείμενο φυσικής γλώσσας η οποία δεν είναι πάντα καλά δομημένη και μπορεί να είναι σημασιολογικά ασαφή. Από την άλλη πλευρά, ένα σύστημα ανάκτησης δεδομένων (όπως μια σχεσιακή βάση δεδομένων) συναλλάσει δεδομένα τα οποία έχουν μια καλά ορισμένη δομή και σημασιολογία. (Baeza-Yates & Ribero-Neto, 1999)

Η ανάκτηση δεδομένων, ενώ εξασφαλίζει μια λύση στους χρήστες ενός συστήματος βάσεων δεδομένων, δεν λύνει το πρόβλημα της ανάκτησης πληροφορίας για ένα ζήτημα ή θέμα. Για να μπορέσει να είναι αποτελεσματική και σ' αυτό επιχειρεί να ικανοποιήσει τις πληροφορίες που χρειάζεται ένας χρήστης

χρησιμοποιώντας τα συστήματα IR για την εξήγηση του περιεχόμενου των αντικειμένων (εγγράφων) σε μια συλλογή και τη θέση αυτών ανάλογα με την βαθμίδα της εφαρμογής στην ερώτηση του χρήστη. Αυτή η 'μετάφραση' του περιεχομένου του εγγράφου συνδέει τα συντακτικά αποσπάσματα και τις σημασιολογικές πληροφορίες από το κείμενο του εγγράφου και χρησιμοποιεί τις πληροφορίες αυτές για να ταιριάξει τα αποτελέσματα με τις ανάγκες των χρηστών. Η δυσκολία δεν βρίσκεται μόνο στο πώς να εξαχθούν αυτές οι πληροφορίες αλλά επίσης και στο πώς να τις χρησιμοποιεί για να καθορίσει την εφαρμογή. Στην πραγματικότητα, ο βασικός σκοπός ενός συστήματος IR είναι να ανακτήσει όλα τα έγγραφα τα οποία είναι κατάλληλα για μια ερώτηση του χρήστη, ενώ ταυτόχρονά να ανακτά όσο λιγότερα μη σχετικά έγγραφα είναι δυνατόν.

Πίνακας 1.1 . Οι διαφορές των δύο συστημάτων[e]

	Ανάκτηση Δεδομένων (DR)	**Ανάκτηση Πληροφοριών(IR)**
Ταίριασμα	Ακριβές ταίριασμα	Μερικό ταίριασμα, Καλύτερο ταίριασμα
Εξαγωγή συμπεράσματος	Αφαιρετικό συμπέρασμα	Επαγωγικό Συμπέρασμα
Μοντέλο	Ντετερμινιστικό	Πιθανοκρατικό
Ταξινόμηση	Μονοθετική	Πολυθετική
Γλώσσα Ερώτησης	Τεχνητή	Φυσική
Προδιαγραφή της Ερώτησης	Πλήρης	Ελλιπής
Απαιτούμενα αντικείμενα	Ισότιμα	Σχετικά
Εσφαλμένη Απόκριση	Ευαίσθητο σύστημα	Απαθές Σύστημα

Στον Πίνακα 1.1 είναι συγκεντρωμένες οι κυριότερες διαφορές των συστημάτων IR και DR. Παρακάτω αναπτύσσονται ένα-ένα αυτά τα χαρακτηριστικά. (Rijsbergen K., 1979) Στην ανάκτηση δεδομένων (Data Retrieval-DR) συνήθως προκύπτει ως αποτέλεσμα ένα *ακριβές ταίριασμα*, δηλαδή, γίνετε αναζήτηση για ένα αντικείμενο το οποίο είτε εμφανίζεται είτε όχι σε ένα φάκελο. Στην ανάκτηση πληροφοριών αυτό μπορεί να μας ενδιαφέρει κάποιες φορές, αλλά γενικά θέλουμε να βρούμε εκείνα τα αντικείμενα που *ταιριάζουν μερικώς* και να επιλέξουμε εκείνα με το καλύτερο σκορ.

[e] Πηγή εικόνας από το κεφάλαιο 1 του ηλεκτρονικού βιβλίου 'Information Retrieval' του Keith van Rijsbergen.

Η *εξαγωγή συμπεράσματος* στην ανάκτηση δεδομένων είναι ένα *απλό συμπερασματικό είδος*, για παράδειγμα αν είναι aRb και bRc, τότε aRc. Στη IR είναι συνηθέστερο να χρησιμοποιείται *επαγωγική εξαγωγή συμπεράσματος*: οι σχέσεις καθορίζονται από το βαθμό βεβαιότητας ή αβεβαιότητας γι' αυτό το λόγο η εμπιστοσύνη για την εξαγωγή των συμπερασμάτων είναι μεταβλητή. Αυτή η διάκριση καθοδηγεί στην περιγραφή της ανάκτησης δεδομένων(DR) *ως ντετερμινιστική* και της ανάκτηση πληροφοριών(IR) ως *πιθανοκρατική*. Συχνά το θεώρημα του Bayes επικαλείται για να εκτελέσει την εξαγωγή συμπεράσματος στο IR.

Μια άλλη διάκριση που μπορεί να γίνει επίσης είναι αυτή της *ταξινόμησης*. Στη DR είναι πιο ενδιαφέρον η *μονοθετική* ταξινόμηση, δηλαδή, σύμφωνα με τις ορισμένες με το υπάρχον αντικείμενο κατηγορίες ταξινομούνται και τα υπόλοιπα στοιχεία για να ανήκουν σε μια ταξινόμηση. Στη IR μια ταξινόμηση δεν είναι γενικά ιδιαίτερα χρήσιμη. Συνήθως χρησιμοποιείται μια *πολυθετική* ταξινόμηση.

Η *γλώσσα ερωτήσεων* στη DR θα μπορούσε να είναι γενικά ένα *τεχνητό είδος*, με περιορισμένη σύνταξη και λεξικό, ενώ στη IR προτιμάτε να χρησιμοποιείται η *φυσική γλώσσα* αν και υπάρχουν κάποιες αξιοσημείωτες εξαιρέσεις. Στη DR γενικά η ερώτηση είναι μια *ολοκληρωμένη δήλωση* για το τι ζητείται, στη IR είναι *ελλιπής*. Αυτή η τελευταία διαφορά ανέρχεται εν μέρει στο γεγονός ότι στην IR η αναζήτηση γίνεται για κατάλληλα δεδομένα που αντιτίθενται στο ακριβές ταίριασμα των αντικειμένων. Ο βαθμός του ταιριάσματος στην IR δηλώνει τη πιθανότητα για τη σχέση των αντικειμένων. Ένα απλό συμπέρασμα για αυτή τη διαφορά είναι ότι η DR είναι πιο *ευαίσθητη σε λάθη* αυτό υπονοεί ότι ένα λάθος στο ταίριασμα έχει ως αποτέλεσμα να μην ανακτηθεί το αναζητούμενο αντικείμενο το οποίο οδηγεί σε μια πλήρη αποτυχία του συστήματος. Στη IR μικρότερα λάθη στο ταίριασμα γενικά *δεν επηρεάζουν την απόδοση* του συστήματος σημαντικά.

Τελικά, το συμπέρασμα που προκύπτει από τα παραπάνω είναι ότι τα αυτοματοποιημένα συστήματα IR χρησιμοποιούνται για να μειώσουν την υπερφόρτωση των πληροφοριών. Πολλά Πανεπιστήμια και δημόσιες βιβλιοθήκες χρησιμοποιούν τα συστήματα IR για να παρέχουν πρόσβαση σε βιβλία, περιοδικά και άλλα έγγραφα. Μηχανές αναζήτησης στον Ιστό όπως οι Google, Yahoo Search, ή Live Search είναι οι πιο αναγνωρίσιμες IR εφαρμογές.

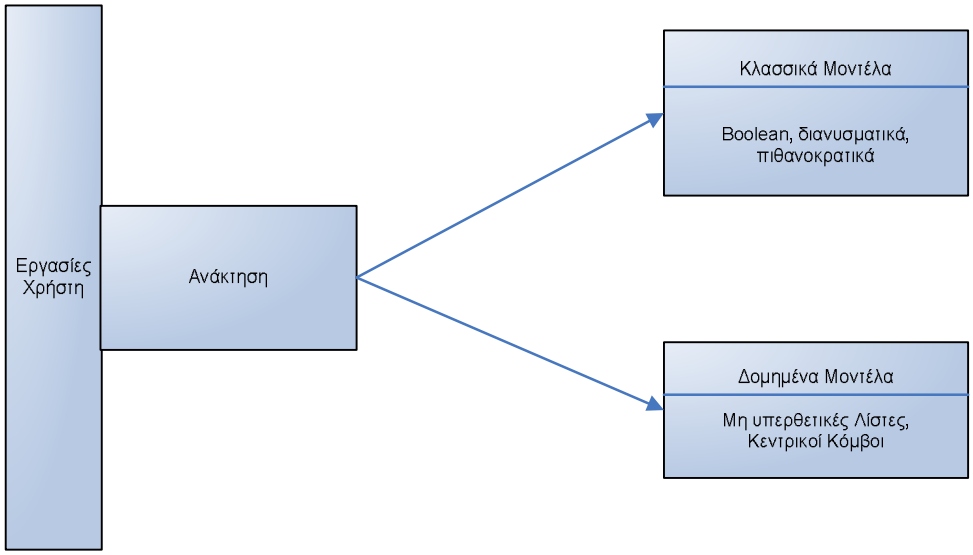
1.8 Μοντέλα των Συστημάτων IR

Τα παραδοσιακά συστήματα ανάκτησης πληροφοριών υιοθετούν συνήθως τους όρους για τους οποίους έχουν δημιουργηθεί ευρετήρια για την ανάκτηση των έγγραφων. Υπό μια περιορισμένη έννοια, ένα ευρετήριο είναι μια λέξη κλειδί (ή ομάδα σχετικών λέξεων) που έχει κάποια σημασία από μόνο του. Στην γενικότερη μορφή του, ένα ευρετήριο είναι απλά οποιαδήποτε λέξη που εμφανίζεται στο κείμενο του εγγράφου στη συλλογή. Οι όροι για τους οποίους έχουν ήδη δημιουργηθεί ευρετήρια χρησιμοποιούνται για την δημιουργία και άλλων ευρετηρίων αλλά και για να συνοψίσουν το περιεχόμενο του κειμένου. Συνήθως οι όροι αυτοί είναι ουσιαστικά, καθώς αυτά έχουν κάποιο νόημα από μόνα τους, και επιπλέον η σημασιολογία τους είναι εύκολη να αναγνωριστεί και να γίνει κατανοητή. Η ανάκτηση με ευρετήρια είναι απλή και συνήθως προβάλλει τις ερωτήσεις κλειδιά λαμβάνοντας υπ' όψιν τις εργασίες της ανάκτησης πληροφοριών. Παραδείγματος χάριν, η ανάκτηση που χρησιμοποιεί όρους ευρετηρίων υιοθετεί ως βασικό θεμέλιο την ιδέα ότι η σημασιολογία των εγγράφων και των πληροφοριών που χρειάζεται ο χρήστης μπορεί να εκφραστεί φυσικά μέσω του συνόλου των όρων που υπάρχουν στα ευρετήρια. Σαφώς, αυτό είναι μια ιδιαίτερη απλοποίηση του προβλήματος επειδή η σημασιολογία σε ένα αίτημα εγγράφων ή χρηστών χάνεται όταν αντικαθιστάτε το κείμενό της με ένα σύνολο λέξεων. Κατά συνέπεια, δεν αποτελεί έκπληξη που τα έγγραφα που ανακτώνται ως απάντηση σε ένα αίτημα του χρηστή, το οποίο εκφράζεται ως σύνολο λέξεων κλειδιών, είναι συχνά άσχετα. Εάν επίσης θεωρηθεί ότι οι περισσότεροι χρήστες δεν έχουν εκπαιδευτεί για να διαμορφώνουν κατάλληλα τις ερωτήσεις τους, το πρόβλημα επιδεινώνεται με ενδεχομένως καταστρεπτικά αποτελέσματα.

Μια παρατήρηση που προκύπτει, επίσης, από τα παραπάνω είναι ότι σε ένα δοθέντος σύνολο από ευρετήρια για ένα έγγραφο, δεν είναι όλοι οι όροι κατάλληλοι για την περιγραφή των περιεχομένων του εγγράφου. Στην πραγματικότητα, υπάρχουν ευρετήρια τα οποία είναι πιο ασαφή από ότι άλλα. Κάτι το οποίο πρέπει να ξεκαθαριστεί είναι εάν τα ευδιάκριτα ευρετήρια παρουσιάζουν ποικίλη σχετικότητα όταν χρησιμοποιούνται για την περιγραφή των περιεχομένων του εγγράφου. Αυτό το αποτέλεσμα πετυχαίνεται μέσο της εντολής του αριθμητικού βάρους για κάθε ευρετήριο του εγγράφου.

Σαφώς, ένα κεντρικό πρόβλημα σχετικά με τα συστήματα ανάκτησης πληροφοριών είναι το ζήτημα της πρόβλεψης για το ποιο έγγραφο είναι σχετικό και ποιο δεν είναι. Μια τέτοια απόφαση εξαρτάται συνήθως από έναν αλγόριθμο ταξινόμησης που προσπαθεί να δημιουργήσει μια απλή διάταξη των εγγράφων που ανακτώνται. Τα έγγραφα που εμφανίζονται στην κορυφή αυτής της διάταξης θεωρούνται πιθανότερο να είναι σχετικά. Κατά συνέπεια, οι αλγόριθμοι ταξινόμησης βρίσκονται στον πυρήνα των συστημάτων ανάκτησης πληροφοριών.

Τα τρία κλασικά πρότυπα στην ανάκτηση πληροφοριών ονομάζονται: Boolean, διανυσματικά, και πιθανολογικά. Στο Boolean πρότυπο, τα έγγραφα και οι ερωτήσεις αντιπροσωπεύονται ως σύνολα δεικτοδοτούμενων όρων. Κατά συνέπεια το πρότυπο τίθεται θεωρητικό. Στο διανυσματικό πρότυπο, τα έγγραφα και οι ερωτήσεις αντιπροσωπεύονται ως διανύσματα σε ένα χώρο t διαστάσεων. Κατά συνέπεια, το πρότυπο μπορεί να ονομαστεί *αλγεβρικό*. Στο πιθανολογικό πρότυπο, το πλαίσιο για τη διαμόρφωση του έγγραφου και την αντιπροσώπευση της ερώτησης είναι βασισμένο στη θεωρία των πιθανοτήτων. Κατά συνέπεια το πρότυπο ονομάζεται *πιθανολογικό*. (Baeza-Yates & Ribero-Neto, 1999)



Εικόνα 1.5 Μια ταξινόμηση των προτύπων ανάκτησης πληροφορίας[f]

Στην συνέχεια θα αναλύσουμε τα τρία κλασσικά πρότυπα που απεικονίζονται στην Εικόνα 1.5 για την ανάκτηση πληροφορίας.

[f] Πηγή της εικόνας από το κεφάλαιο 2 του βιβλίου ‘Modern Information Retrieval’ των Ricardo Baeza-Yates και Berthier Ribeiro-Neto (1999).

1.8.1 Το Boolean πρότυπο

Το Boolean πρότυπο είναι ένα απλό πρότυπο ανάκτησης βασισμένο στην θεωρία των συνόλων και την Boolean άλγεβρα. Δεδομένου ότι η αρχή ενός συνόλου είναι αρκετά διαισθητική, το Boolean πρότυπο παρέχει ένα πλαίσιο που είναι εύκολο να γίνει κατανοητό από ένα κοινό χρήστη ενός συστήματος IR. Επιπλέον, οι ερωτήσεις καθορίζονται ως εκφράσεις Boolean οι οποίες έχουν ακριβή σημασιολογία. Το πρότυπο αυτό ήταν το κύριο μοντέλο που χρησιμοποιήθηκε στο παρελθόν από τα πρόωρα συστήματα.

Δυστυχώς, το Boolean πρότυπο πάσχει από σημαντικά μειονεκτήματα. Κατ' αρχάς, η στρατηγική ανάκτησής της είναι βασισμένη σε ένα δυαδικό κριτήριο απόφασης (δηλαδή ένα έγγραφο προβλέπεται για να είναι είτε σχετικό είτε άσχετο) χωρίς οποιαδήποτε έννοια μιας κλίμακας βαθμολόγησης, η οποία αποτρέπει την καλή εκτέλεση ανάκτησης. Κατά συνέπεια, το Boolean πρότυπο είναι περισσότερο ένα πρότυπο ανάκτησης δεδομένων, αντί των πληροφοριών. Επιπλέον, οι Boolean εκφράσεις, που σχηματίζονται χρησιμοποιώντας τις λογικές συναρτήσεις AND, OR και NOT, ενώ κατέχουν μια ορθή σημασιολογία, συχνά δεν είναι εύκολο για το χρήστη να μετατρέψει τις ανάγκες του, δηλαδή της πληροφορίες που αναζητά, σε μια Boolean έκφραση. Στην πραγματικότητα, οι περισσότεροι χρήστες το βρίσκουν δύσκολο και αδέξιο να εκφράσουν την ερώτησής τους σε μια Boolean έκφραση. Παρόλα αυτά οι Boolean εκφράσεις που διατυπώνονται περισσότερο από τους χρήστες είναι συχνά αρκετά απλές. Όμως παρά αυτά τα μειονεκτήματα, το Boolean πρότυπο είναι ακόμα το κυρίαρχο πρότυπο για τα εμπορικά συστήματα βάσεων δεδομένων εγγράφων και παρέχει μια καλή αφετηρία για εκείνους που είναι καινούργιοι στον τομέα της ανάκτησης.

Ορισμός Για το Boolean πρότυπο, οι μεταβλητές των βαρών όρων δεικτών είναι όλες δυαδικές δηλαδή, $w_{i,j} \in 0,1$. Μια ερώτηση q είναι μια συμβατική έκφραση Boolean. Έστω το $\vec{q}_{dnf}$ είναι η διαζευκτική κανονική μορφή για την ερώτηση q. Περαιτέρω, έστω ότι το $\vec{q}_{cc}$ είναι ένα από τα συνδετικά συστατικά του $\vec{q}_{dnf}$. Η ομοιότητα ενός εγγράφου d_j στην ερώτηση q ορίζεται ως $sim(d_j, q) = \begin{cases} 1 \\ 0 \end{cases}$, όπου 1 αν $\exists \vec{q}_{cc} \left| (\vec{q}_{cc} \in \vec{q}_{dnf}) \wedge (\forall k_i, g_i(\vec{d}_j) = g_i(\vec{q}_{cc})) \right.$ και 0 αλλού.

Εάν $sim(d_j, q) = 1$ τότε το Boolean πρότυπο προβλέπει ότι το έγγραφο d_j είναι σχετικό με την ερώτηση q. Διαφορετικά, η πρόβλεψη είναι ότι το έγγραφο δεν είναι σχετικό. (Baeza-Yates & Ribero-Neto, 1999)

Το κύριο πλεονέκτημα του Boolean προτύπου είναι ο καθαρός φορμαλισμός πίσω από το πρότυπο και η απλότητά του. Το κύριο μειονέκτημα είναι ότι το ακριβές ταίριασμα μπορεί να οδηγήσει στην ανάκτηση πολύ λίγων ή πάρα πολλών εγγράφων. Πλέον, είναι ευρέως γνωστό ότι η στάθμιση ευρετηρίου μπορεί να οδηγήσει σε μια ουσιαστική βελτίωση στην απόδοση ανάκτησης. Η στάθμιση ευρετηρίου οδηγεί στην χρήση του διανυσματικού πρότυπου.

1.8.2 Το Διανυσματικό πρότυπο

Στο χώρο διανύσματος το πρότυπο κείμενο αντιπροσωπεύεται από ένα διάνυσμα *των όρων*. Ο καθορισμός ενός όρου δεν είναι έμφυτος στο πρότυπο, αλλά οι όροι είναι χαρακτηριστικές λέξεις και φράσεις. Εάν οι λέξεις επιλέγονται ως όροι, κατόπιν κάθε λέξη στο λεξιλόγιο γίνεται μια ανεξάρτητη διάσταση σε ένα υψηλής διάστασης διανυσματικό διάστημα. Οποιοδήποτε κείμενο μπορεί έπειτα να αντιπροσωπευθεί από ένα διάνυσμα σε αυτό το υψηλής διάστασης διάστημα. Εάν ένας όρος ανήκει σε ένα κείμενο, παίρνει μια διαφορετική από το μηδέν τιμή στο κείμενο-διάνυσμα κατά μήκος της διάστασης που αντιστοιχεί στον όρο. Δεδομένου ότι οποιοδήποτε κείμενο περιέχει ένα περιορισμένο σύνολο όρων (το λεξιλόγιο μπορεί να περιέχει εκατομμύρια όρους), τα περισσότερα διανύσματα κειμένων είναι πολύ αραιά. Τα περισσότερα συστήματα που βασίζονται στο διανυσματικό πρότυπο λειτουργούν στο θετικό τεταρτημόριο του διανυσματικού διαστήματος, δηλαδή, κανένας όρος δεν ορίζεται από μια αρνητική τιμή.

Για να οριστεί ένα αριθμητικό αποτέλεσμα σε ένα έγγραφο για μια ερώτηση, το πρότυπο μετρά την *ομοιότητα* μεταξύ του διανύσματος ερώτησης (δεδομένου ότι η ερώτηση είναι επίσης κείμενο και μπορεί να μετατραπεί σε ένα διάνυσμα) και το διάνυσμα εγγράφων. Η ομοιότητα μεταξύ δύο διανυσμάτων δεν είναι έμφυτη στο πρότυπο. Χαρακτηριστικά, η γωνία μεταξύ δύο διανυσμάτων χρησιμοποιείται ως μέτρο της απόκλισης μεταξύ των διανυσμάτων, και το συνημίτονο της γωνίας χρησιμοποιείται ως αριθμητική ομοιότητα, και αυτό γιατί το συνημίτονο έχει την εξής εξαιρετικά χρήσιμη ιδιότητα: παίρνει την τιμή (1, 0) για ταυτόσημα διανύσματα

και την τιμή (0, 0) για τα ορθογώνια διανύσματα. Εάν είναι d το διάνυσμα εγγράφων και s το διάνυσμα ερώτησης, τότε η ομοιότητα του εγγράφου - ερώτησης (ή τα αποτέλεσμα για το q) μπορεί να αντιπροσωπευθεί ως:

$$sim(\vec{D},\vec{Q}) = \sum_{t_i \in Q,D} w_{t_{iQ}} \cdot w_{t_{iD}}$$

όπου $w_{t_{iQ}}$ είναι η τιμή της i-στή συνιστώσας του διανύσματος ερώτησης $\vec{Q}$, και το $w_{t_{iD}}$ είναι η i-στη συνιστώσα του διανύσματος εγγράφου $\vec{D}$.

Ορισμός Για το διανυσματικό πρότυπο, το βάρος $w_{i,j}$ που συνδέεται με ένα ζευγάρι (k_i, d_j) είναι θετικός και μη-δυαδικός. Περαιτέρω, οι όροι δεικτών στην ερώτηση είναι επίσης σταθμισμένοι. Έστω τα $w_{i,q}$ είναι το βάρος για το ζευγάρι [k_i,q], όπου $w_{i,q} >= 0$. Κατόπιν, το διάνυσμα ερώτησης q ορίζεται ως $q=(w_{1,q}, w_{2,q}\ldots,w_{t,q})$ όπου το t είναι ο συνολικός αριθμός όρων δεικτών στο σύστημα. Επομένως, το διάνυσμα για ένα έγγραφο d_j αντιπροσωπεύεται από το $d_j=(w_{1,j}, w_{2,j}..... w_{t,j})$. (Baeza-Yates & Ribero-Neto, 1999)

1.8.3 Το Πιθανολογικό πρότυπο

Η οικογένεια των συστημάτων IR είναι βασισμένη στην γενική αρχή ότι τα έγγραφα σε μια συλλογή θα πρέπει να κατατάσσονται σύμφωνα με την ελάχιστη πιθανότητα που παρουσιάζουν σε μια εφαρμογή ερώτησης. Αυτό συχνά καλείται *Αρχή της πιθανοκρατικής κατάταξης* (*The Probabilistic Ranking Principle* - PRP). Καθώς οι πραγματικές πιθανότητες δεν είναι διαθέσιμες σε ένα IR σύστημα, τα πιθανοκρατικά συστήματα IR *εκτιμούν* την πιθανότητα της εφαρμογής των εγγράφων για μια ερώτηση. Αυτή η εκτίμηση είναι το κλειδί του τμήματος του μοντέλου, και αυτό είναι που κάνει το πιθανοκρατικό μοντέλο διαφορετικό από τα άλλα. Η αρχική ιδέα της πιθανοκρατικής ανάκτησης προτάθηκε από τους Maron και Kuhns σε μια διατριβή που δημοσιεύθηκε το 1960. Από τότε, πολλά πιθανοκρατικά μοντέλα έχουν προταθεί. Το κλασσικό, όμως, πιθανοκρατικό πρότυπο περιγράφηκε το 1976 από τους Roberston και Spack Jones, το οποίο στην συνέχεια έγινε γνωστό ως το πρότυπο της δυαδικής ανεξάρτητης ανάκτησης.

Η ιδέα λειτουργίας αυτού του μοντέλου είναι η ακόλουθη: Δίνοντας ο χρήστης μια ερώτηση, υπάρχει ένα σύνολο από έγγραφα που περιέχουν ακριβώς τα σχετικά έγγραφα και όχι κάποια επιπλέον. Συχνά αυτό το σύνολο των εγγράφων

αποκαλείται και ιδεατό σύνολο απαντήσεων. Από την στιγμή που δίνεται αυτό το σύνολο ιδεατών απαντήσεων, δεν υπάρχει κάποιο πρόβλημα στην ανάκτηση των εγγράφων. Η διαδικασία της ερώτησης μπορεί να θεωρηθεί ως μια διαδικασία καθορισμού των ιδιοτήτων του ιδεατού συνόλου απαντήσεων. Το πρόβλημα, όμως, είναι ότι δεν μπορεί κάποιος να γνωρίζει αυτές τις ιδιότητες ακριβώς. Αυτό όμως που είναι γνωστό είναι ότι υπάρχουν ευρετήρια που η σημασιολογία τους μπορεί να χρησιμοποιηθεί για την περιγραφή αυτών των ιδιοτήτων. Καθώς αυτές οι ιδιότητες δεν είναι γνωστές κατά την διάρκεια της ερώτησης, μια προσπάθεια που πρέπει να γίνει αρχικά είναι να μαντεύσουν αυτές οι ιδιότητες. Αυτή η αρχικά εκτίμηση επιτρέπει να παραχθούν προπαρασκευαστικές πιθανολογικές περιγραφές για το ιδεατό σύνολο απαντήσεων.

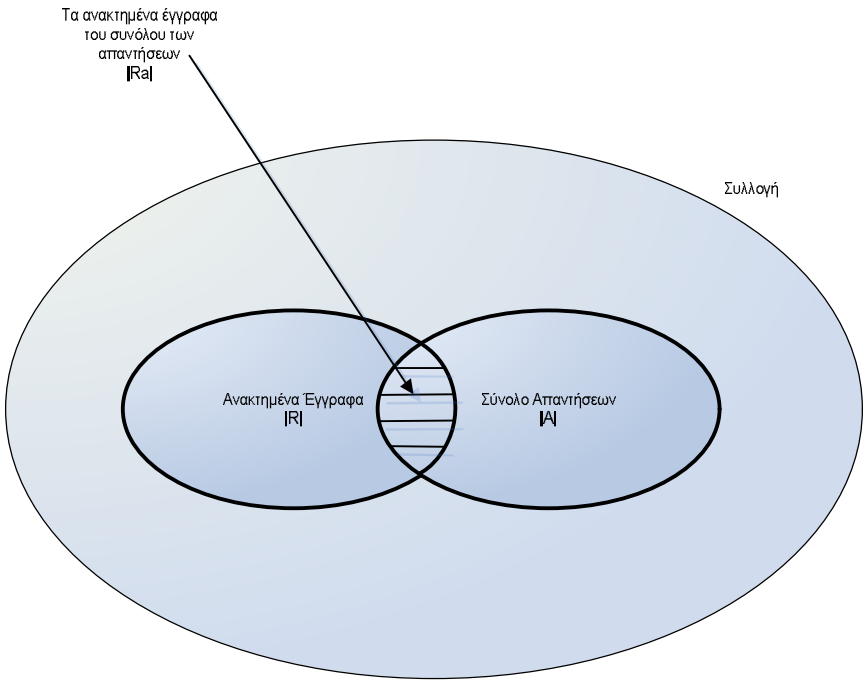
Ορισμός Για το πιθανολογικό πρότυπο, οι μεταβλητές είναι όλες δυαδικές δηλαδή, $w_{i,j} \in \{0,1\}$, $w_{i,q} \in \{0,1,\}$. Μια ερώτηση q είναι ένα υποσύνολο των όρων από τα ευρετήρια. Έστω το R να είναι το σύνολο εγγράφων που το μοντέλο γνωρίζει (ή αρχικά υποθέτει) ως σχετικό. Έστω το $\overline{R}$ να είναι το συμπλήρωμα του R (δηλαδή, το σύνολο των άσχετων εγγράφων). Έστω ότι το $P(R \mid \vec{d}_j)$ είναι η πιθανότητα το έγγραφο d_j να είναι σχετικό με την ερώτηση q και το $P(\overline{R} \mid \vec{d}_j)$ είναι η πιθανότητα το d_j να είναι άσχετο στην q. Η ομοιότητα sim(d_j,q) του εγγράφου d_j με την ερώτηση q ορίζεται ως η αναλογία $sim(d_j, q) = \frac{P(R \mid \vec{d}_j)}{P(\overline{R} \mid \vec{d}_j)}$. (Baeza-Yates & Ribero-Neto, 1999)

1.9 Μέτρα Αξιολόγησης

Ένα σύστημα ανάκτησης πληροφορίας αποτελείται από ένα σύνολο εγγράφων $D = \{D_1, D_2, ..., D_n\}$. Η είσοδος είναι μια ερώτηση, q, που συχνά αναφέρεται ως λίστα λέξεων κλειδιών. Η ομοιότητα μεταξύ της ερώτησης και του εγγράφου υπολογίζεται ως $sim(q, D_i)$. Αυτό το μέτρο ομοιότητας είναι μια συνάρτηση συμμετοχής που περιγράφει την πιθανότητα το έγγραφο να είναι σχετικό, δηλαδή να είναι ενδιαφέρον για το χρήστη σύμφωνα με τα κριτήρια του χρήστη, τα οποία δηλώθηκαν μέσω της ερώτησης του. Η αποτελεσματικότητα του συστήματος συχνά μετριέται με την χρήση

της ανάκλησης και της ακρίβειας. Αν και έχουν προταθεί και άλλα μέτρα για τον έλεγχο της απόδοσης του συστήματος, αυτά τα δύο είναι τα πιο κοινά και ευρέως χρησιμοποιούμενα. (Dunham, 2004)

Έστω ένα παράδειγμα αιτημάτων πληροφοριών *I* (μιας δοκιμαστικής συλλογής αναφοράς) και το σύνολο R των σχετικών εγγράφων του. Έστω ότι |R| είναι ο αριθμός των εγγράφων σε αυτό το σύνολο. Υποθέστε ότι μια δεδομένη στρατηγική ανάκτησης επεξεργάζεται το αίτημα πληροφοριών *I* και παράγει ένα σύνολο εγγράφων που αποτελούν την απάντηση, Α. Έστω ότι |A| είναι ο αριθμός των εγγράφων σε αυτό το σύνολο. Περαιτέρω, έστω ότι $|R_a|$ είναι ο αριθμός εγγράφων στη διατομή των συνόλων R και Α. (Baeza-Yates & Ribero-Neto, 1999) Η Εικόνα 1.6 επεξηγεί αυτά τα σύνολα.



Εικόνα 1.6 Η ακρίβεια και η ανάκληση για μια ερώτηση[g]

Τότε τα μέτρα ανάκλησης και ακρίβειας καθορίζονται ως εξής.

Η ανάκληση είναι το μέρος των σχετικών εγγράφων (το σύνολο R) που έχει ανακτηθεί δηλαδή,

$$\mathrm{Re}\,call = \frac{|R_a|}{|R|} \qquad \text{ή} \qquad \text{Ανάκληση} = \frac{|\,\Sigma\chi\varepsilon\tau\iota\kappa\acute{\alpha}\_\kappa\alpha\iota\_\text{Ανακτημένα}\,|}{|\,\Sigma\chi\varepsilon\tau\iota\kappa\acute{\alpha}\,|}.$$

[g] Πηγή της εικόνας από το κεφάλαιο 3 του βιβλίου 'Modern Information Retrieval' των Ricardo Baeza-Yates και Berthier Ribeiro-Neto (1999).

Η ανάκληση αφ' ετέρου είναι η ικανότητα ενός συστήματος ανάκτησης να αποκτήσει όλα ή τα περισσότερα από τα σχετικά έγγραφα της συλλογής. Κατά συνέπεια απαιτεί τη γνώση όχι μόνο των σχετικών και ανακτημένων αλλά και εκείνων που δεν ανακτώνται. (Clarke & Willet, 1997) Δεν υπάρχει καμία κατάλληλη μέθοδος για την απόλυτη ανάκληση των μηχανών αναζήτησης καθώς είναι αδύνατο να είναι γνωστός ο συνολικός αριθμός σχετικότητας στις τεράστιες βάσεις δεδομένων.

Στη δυαδική ταξινόμηση, η ανάκληση ονομάζεται ευαισθησία.

Είναι τετριμμένο να επιτευχθεί η ανάκληση 100% με την επιστροφή όλων των εγγράφων ως απάντηση σε οποιαδήποτε ερώτηση. Επομένως η ανάκληση δεν είναι από μόνη της αρκετή αλλά πρέπει να μετρηθεί και ο αριθμός των μη σχετικών εγγράφων επίσης, παραδείγματος χάριν με τον υπολογισμό της ακρίβειας.

Η ακρίβεια είναι το μέρος των ανακτημένων εγγράφων (το σύνολο A) που είναι σχετικά δηλαδή,

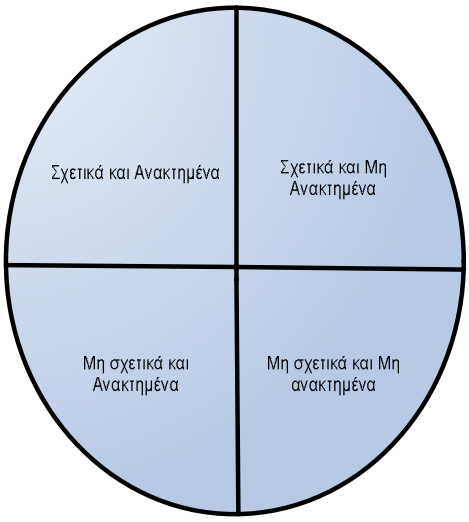
$$\Pr ecission = \frac{|R_a|}{|A|} \quad \text{ή} \quad \text{Α}κρίβεια = \frac{|\,\Sigma χετικά\_και\_Ανακτημένα\,|}{|\,\text{Α}νακτημένα\,|}.$$

Η ακρίβεια είναι το κλάσμα της εξόδου της αναζήτησης που είναι σχετική για μια συγκεκριμένη ερώτηση. Ο υπολογισμός της, ως εκ τούτου, απαιτεί τη γνώση των σχετικών και μη-σχετικών ταιριασμάτων στο αξιολογημένο σύνολο εγγράφων. (Clarke & Willet, 1997) Κατά συνέπεια είναι δυνατό να υπολογιστεί η απόλυτη ακρίβεια των μηχανών αναζήτησης που αποτελεί μια ένδειξη της σχετικότητας του συστήματος.

Στη δυαδική ταξινόμηση, η ακρίβεια είναι ανάλογη με τη θετική προφητική αξία. Η ακρίβεια λαμβάνει όλα τα ανακτημένα έγγραφα υπόψη.

Η ανάκληση και η ακρίβεια, όπως καθορίζονται ανωτέρω, υποθέτουν ότι όλα τα έγγραφα στην απάντηση του συνόλου A έχουν εξεταστεί (ή αναγνωριστεί). Εντούτοις, στο χρήστη δεν παρουσιάζονται συνήθως όλα τα έγγραφα του συνόλου A αμέσως. Αντί αυτού, τα έγγραφα του A ταξινομούνται αρχικά σύμφωνα με την σχετικότητα τους (δηλαδή παράγεται μια ταξινόμηση). Ο χρήστης εξετάζει έπειτα αυτόν τον ταξινομημένο κατάλογο που αρχίζει από το κορυφαίο έγγραφο. Σε αυτήν την κατάσταση, τα μέτρα ανάκλησης και ακρίβειας ποικίλλουν καθώς ο χρήστης συνεχίζει με την εξέτασή του στο σύνολο των απαντήσεων A.

Η ανάκληση απαντά στην ερώτηση "Έχουν ανακτηθεί όλα τα σχετικά έγγραφα;", ενώ η ακρίβεια δίνει απάντηση στην ερώτηση "Είναι όλα τα ανακτημένα έγγραφα αυτά που ενδιαφέρουν το χρήστη;". Ένα έγγραφο θεωρείται σχετικό αν θα έπρεπε να ανακτηθεί από την ερώτηση. Η Εικόνα 1.7 απεικονίζει τα τέσσερα πιθανά αποτελέσματα των ερωτήσεων. Από αυτά τα τέσσερα τεταρτημόρια, τα δύο αντιπροσωπεύουν επιθυμητά αποτελέσματα: σχετικά και ανακτημένα, και άσχετα και μη ανακτημένα. Τα άλλα δυο τεταρτημόρια αντιπροσωπεύουν λανθασμένες καταστάσεις. Έγγραφα που είναι σχετικά και μη ανακτημένα θα έπρεπε να ανακτηθούν αλλά δεν ανακτήθηκαν. Έγγραφα που είναι ανακτημένα και μη σχετικά δεν έπρεπε να ανακτηθούν αλλά ανακτήθηκαν.



Εικόνα 1.7 Αποτελέσματα Ερωτήσεων[h]

Κάποια ακόμη μέτρα που μπορούν να χρησιμοποιηθούν είναι τα παρακάτω, αλλά δε συναντιόνται συχνά.

Η ισορροπία (συμμετρία - Fall-Out) για μη σχετικά έγγραφα που ανακτώνται, από όλα τα μη σχετικά έγγραφα που είναι διαθέσιμα:

$$fall-out = \frac{|\{\mu\eta - \sigma\chi\varepsilon\tau\iota\kappa\acute{\alpha}\_\acute{\varepsilon}\gamma\gamma\rho\alpha\varphi\alpha\} \cap \{\alpha\nu\alpha\kappa\tau\eta\mu\acute{\varepsilon}\nu\alpha\_\acute{\varepsilon}\gamma\gamma\rho\alpha\varphi\alpha\}|}{|\{\mu\eta - \sigma\chi\varepsilon\tau\iota\kappa\acute{\alpha}\_\acute{\varepsilon}\gamma\gamma\rho\alpha\varphi\alpha\}|}.$$

Ο σταθμισμένος αρμονικός μέσος όρος της ακρίβειας και της ανάκλησης, του παραδοσιακού F-μέτρου ή του ισορροπημένου F-αποτελέσματος είναι:

$F = 2 \cdot (\alpha\kappa\rho\acute{\iota}\beta\varepsilon\iota\alpha \cdot \alpha\nu\acute{\alpha}\kappa\lambda\eta\sigma\eta) / (\alpha\kappa\rho\acute{\iota}\beta\varepsilon\iota\alpha + \alpha\nu\acute{\alpha}\kappa\lambda\eta\sigma\eta)$. (Korfhage, 1997)

[h] Πηγή της εικόνας από το κεφάλαιο 2 του βιβλίου 'Data Mining' της Margaret H. Dunham (2004).

1.10 Συναρτήσεις Κατάταξης Αποτελεσμάτων

Οι συναρτήσεις κατάταξης των αποτελεσμάτων είναι συναρτήσεις που χρησιμοποιούνται από τις μηχανές αναζήτησης για την κατάταξη των εγγράφων που επιστρέφονται κατά την αναζήτηση σύμφωνα με τη συνάφεια τους ως προς την ερώτηση αναζήτησης. Υπάρχουν δύο κύριες συναρτήσεις: 1. η TF-IDF και 2. η Okapi BM25.

1.10.1 TF-IDF

Το βάρος TF-IDF είναι ένα βάρος που χρησιμοποιείται συχνά στην ανάκτηση πληροφοριών και τη εξόρυξη κειμένων. Αυτό το βάρος είναι ένα στατιστικό μέτρο που χρησιμοποιείται για να αξιολογήσει πόσο σημαντική είναι μια λέξη σε ένα έγγραφο σε μια συλλογή κειμένων. Το TF είναι τα αρχικά των λέξεων term frequency, που σημαίνει η συχνότητα των όρων και το IDF είναι τα αρχικά των λέξεων inverse document frequency, αντίστροφη συχνότητα εγγράφων. (Hand, Mannila, & Smyth , 2001)

Η σημαντικότητα αυξάνεται αναλογικά με τον αριθμό των φορών που εμφανίζεται μια λέξη στο έγγραφο, αλλά αντισταθμίζεται από τη συχνότητα της λέξης στη συλλογή. Το μέτρο μεταβολής της συνάρτησης TF-IDF χρησιμοποιείται συχνά από τις μηχανές αναζήτησης ως κεντρικό εργαλείο στην επιτυχία και τη ταξινόμηση της σχετικότητας ενός εγγράφου δεδομένης μιας ερώτησης του χρήστη.

Παρακάτω θα αναλύσουμε μαθηματικά τον υπολογισμό αυτών των όρων.

Η συχνότητα του όρου (TF) στο δεδομένο έγγραφο είναι απλά ο αριθμός των φορών που ένας δεδομένος όρος εμφανίζεται σ' αυτό το έγγραφο. Αυτή η εκτίμηση είναι συνήθως ομαλοποιημένη για να αποτραπεί μια στατιστική απόκλιση προς τα μεγαλύτερα έγγραφα, τα οποία μπορούν να έχουν μια υψηλότερη συχνότητα όρου ανεξάρτητα από την πραγματική σημασία του όρου στο έγγραφο, έτσι ώστε να δώσει ένα μέτρο σπουδαιότητας του όρου t_i μέσα στο έγγραφο d_j. Ο τύπος που χρησιμοποιείται για τον υπολογισμό της TF είναι ο:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}},$$

όπου το $n_{i,j}$ είναι ο αριθμός των εμφανίσεων του εξεταζόμενου όρου στο έγγραφο d_j, και ο παρονομαστής είναι ο αριθμός των εμφανίσεων όλων των όρων στο έγγραφο d_j.

Η αντίστροφη συχνότητα εγγράφων (IDF) είναι ένα μέτρο γενικής σπουδαιότητας του όρου. Αυτή η σπουδαιότητα λαμβάνεται με τη διαίρεση του αριθμού όλων των εγγράφων προς τον αριθμό των εγγράφων που περιέχουν τον όρο, και στην συνέχεια παίρνουμε το λογάριθμο του πηλίκου αυτού. Οπότε ο μαθηματικός τύπος για τον υπολογισμό της IDF είναι ο εξής:

$$idf_i = \log \frac{|D|}{|\{d_j : t_i \in d_j\}|}$$

όπου το $|D|$ είναι ο συνολικός αριθμός των εγγράφων στην συλλογή και ο παρανομαστής είναι ο αριθμός των εγγράφων όπου εμφανίζεται ο όρος t_i, και ισχύει ότι $n_{i,j} \neq 0$.

Τελικά, ο υπολογισμός της συνάρτησης, tf – idf, γίνεται ως εξής:

$tf - idf_{i,j} = tf_{i,j} * idf_i$.

Από τα παραπάνω μπορεί κανείς να συμπεράνει ότι για να επιτευχθεί μια υψηλή τιμή της συνάρτησης TF-IDF θα πρέπει να υπάρξει μια υψηλή συχνότητα εμφάνισης του όρου (στο δεδομένο έγγραφο) και μια χαμηλή συχνότητα εγγράφων που να περιέχουν τον όρο σε ολόκληρη τη συλλογή των εγγράφων.

1.10.2 Okapi BM25

Το Okapi BM25 είναι μια συνάρτηση ταξινόμησης που χρησιμοποιείται από τις μηχανές αναζήτησης για να ταξινομήσει τα έγγραφα, που επιστρέφονται ως αποτελέσματα σε μια αναζήτηση, σύμφωνα με τη σχετικότητά τους σε μια δεδομένη ερώτηση αναζήτησης. Είναι βασισμένο στο πιθανολογικό πλαίσιο ανάκτησης που αναπτύχθηκε στις δεκαετίες του '70 και του '80 από τους Stephen E. Robertson, Karen Spärck Jones, και άλλους.

Το όνομα της πραγματικής αυτής συνάρτησης είναι BM25. Εντούτοις, συνήθως αναφέρεται ως ‘Okapi BM25’, από το σύστημα ανάκτησης πληροφοριών Okapi, που εφαρμόστηκε στο Πανεπιστήμιο London’s City στη δεκαετία του '80 και τη δεκαετία του '90, και πρόκειται για το πρώτο σύστημα που εφαρμόσει αυτήν την λειτουργία.

Το BM25, και οι νεότερες παραλλαγές του, όπως το BM25F, αντιπροσωπεύει τη συνάρτηση ανάκτησης ανώτατου επιπέδου τεχνικής που χρησιμοποιείται στην ανάκτηση εγγράφων, όπως η αναζήτηση στον Ιστό.

Το BM25 είναι μια συνάρτηση ανάκτησης λέξεων που ταξινομεί ένα σύνολο εγγράφων βασισμένων στους όρους ερώτησης που εμφανίζονται σε κάθε έγγραφο, ανεξάρτητα από την αλληλεξάρτηση μεταξύ των όρων ερώτησης μέσα σε ένα έγγραφο, όπως είναι για παράδειγμα η σχετική εγγύτητά τους. Δεν πρόκειται για μια ενιαία συνάρτηση, αλλά για μια ολόκληρη οικογένεια συναρτήσεων βαθμολογήσεων, οπού υπάρχουν ελαφρώς διαφορετικά συστατικά και παράμετροι. Ένα από τα πιο σημαντικά σημεία της συνάρτησης είναι το ακόλουθο.

Λαμβάνοντας υπόψη μια ερώτηση *q*, που περιέχει τις λέξεις κλειδιά $q_1 ... , q_n$, το αποτέλεσμα του BM25 για ένα έγγραφο *D* είναι:

$$score(D,Q) = \sum_{i-1} IDF(q_i) * \frac{f(q_i, D) * (k_1 + 1)}{f(q_i, D) + k_1 * (1 - b + b * \frac{|D|}{avgdl})},$$

όπου το $f(q_i, D)$ είναι συχνότητα του όρου q_i στο έγγραφο D, το $|D|$ είναι το μήκος του εγγράφου D (δηλαδή ο αριθμός των λέξεων που περιέχει), και το *avgdl* είναι το μέσο μήκος εγγράφων στη συλλογή κειμένων από την οποία προέρχονται τα έγγραφα. Τα k_1 και το *b* είναι ελεύθερες παράμετροι, συνήθως επιλέγονται ως $k_1 = 2,0$ και $b = 0,75$. Το $IDF(q_i)$ είναι το βάρος IDF για τον όρο της ερώτησης q_i και υπολογίζεται συνήθως ως: $idf(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$, όπου το *N* είναι ο συνολικός αριθμός εγγράφων στη συλλογή, και το $n(q_i)$ είναι ο αριθμός εγγράφων που περιέχουν το q_i.

Η πιθανολογική φύση του BM25 γίνεται προφανής από τη IDF. Έστω ότι ένας όρος *q* ερώτησης που εμφανίζεται στα έγγραφα *n(q)*. Κατόπιν ένα τυχαία επιλεγμένο έγγραφο *D* περιέχει τον όρο με πιθανότητα $n(q)/N$, όπου το *N* είναι πάλι το σύνολο εγγράφων στη συλλογή. Επομένως, το περιεχόμενο πληροφοριών του μηνύματος "D περιέχει το q" είναι: $-\log \frac{n(q)}{N} = \log \frac{N}{n(q)}$. Έστω ότι τώρα υπάρχουν δύο όροι ερώτησης q_1 και q_2. Εάν οι δύο όροι εμφανίζονται στα έγγραφα εξ' ολοκλήρου ανεξάρτητα ο ένας από τον άλλον, τότε η πιθανότητα της ύπαρξης και του q_1 και του q_2 σε ένα τυχαία επιλεγμένο έγγραφο *D* είναι: $\frac{n(q_1)}{N} * \frac{n(q_2)}{N}$ και το περιεχόμενο

πληροφοριών ενός τέτοιου γεγονότος είναι: $\sum_{i=1}^{2} \log \frac{N}{n(q_i)}$. Αυτό αντιπροσωπεύει το τι εκφράζει ο όρος IDF στο BM25.

1.11 Λογισμικά για IR

Καθώς ο τομέας της ανάκτησης πληροφορίας έχει αναπτυχθεί πολύ τα τελευταία χρόνια, έχουν δημιουργηθεί και πολλά λογισμικά για αυτά τα συστήματα. Κάποια από αυτά είναι ελεύθερου κώδικα ενώ κάποια άλλα όχι. Παρακάτω θα δοθούν κάποια χαρακτηριστικά διαφόρων ελεύθερων λογισμικών.

1.11.1 Egothor

Η Egothor είναι μια ανοικτού κώδικα, υψηλής απόδοσης, πλήρως χαρακτηρισμένων κειμένων μηχανή αναζήτησης γραμμένη σε Java. Η τεχνολογία της είναι κατάλληλη για σχεδόν κάθε εφαρμογή που απαιτεί αναζήτηση σε πλήρη κείμενα, ειδικά σε υβριδικές πλατφόρμες. Μπορεί να σχηματιστεί ως μια ξεχωριστή μηχανή, μετά-αναζήτησης, peer-to-peer HUB, και, επί πλέον, μπορεί να χρησιμοποιηθεί σαν μια βιβλιοθήκη για μια εφαρμογή η οποία χρειάζεται να εκτελέσει αναζήτηση σε πλήρες κείμενο.

Τα χαρακτηριστικά γνωρίσματα της Egothor είναι τα εξής:

- Χρησιμοποιεί ως γλώσσα προγραμματισμού την Java , η οποία της μεταβιβάζει την δυνατότητά της για χρήση σε διαφορετικές πλατφόρμες.
- Είναι ένας νέος δυναμικός αλγόριθμος για γρήγορη ενημέρωση των δεικτών.
- Έχει πυρήνα 64-bit.
- Υποστηρίζει την βελτιωμένη έκδοση εγγράφων, Xdelta.
- Οι ερωτήσεις μπορούν να επιλυθούν με ένα παράλληλο τρόπο.
- Έχει την δυνατότητα να αναγνωρίζει τις πιο κοινές μορφές αρχείων, δηλαδή HTML, PDF, PS, DOC και XLS.
- Υποστηρίζει ρομπότ υψηλής χωρητικότητας, τα οποία υποστηρίζουν την σύσταση robots.txt.
- Χρησιμοποίει τις καλύτερες μεθόδους συμπίεσης, όπως Golomb, Elias-Gamma, Block coding.

- Βασίζεται στο εκτεινόμενο πρότυπο Boolean το οποίο μπορεί να χρησιμοποιήσει πρότυπα Boolean ή διανυσματικά πρότυπα.

Υπάρχουν δυο κλάδοι του προϊόντος: ο κλάδος 1.x είναι γενικής χρήσης και αναπτύχθηκε από την egothor.sf.net, ενώ ο κλάδος 2.x μπορεί να είναι διαθέσιμος για το γενικό κοινό όταν θα ολοκληρωθεί πλήρως. (http://www.egothor.org/index.shtml)

1.11.2 ht://Dig

Το σύστημα ht://Dig είναι ένα ολοκληρωμένο σύστημα Ιστού έρευνας και τοποθέτησης δεικτών σε μια περιοχή ή ένα ενδοδίκτυο. Αυτό το σύστημα δεν έχει ως στόχο του να αντικαταστήσει την ανάγκη για τα αποτελεσματικά συστήματα αναζήτησης του Ιστού όπως είναι τα Lycos, Infoseek, Google και AltaVista. Αντίθετα έχει ως στόχο να καλύψει τις ανάγκες αναζήτησης για μια ενιαία εταιρεία, πανεπιστημιούπολη, ή ακόμη και για ένα συγκεκριμένο υποτμήμα ενός ιστοχώρου.

Σε αντιδιαστολή με μερικές μηχανές αναζήτησης που βασίζονται σε WAIS και στον Ιστό, ο ht://Dig μπορεί εύκολα να επεκταθεί σε αρκετούς υπολογιστές εξυπηρέτησης δικτύου. Ο τύπος αυτών των διαφορετικών εξυπηρετητών δικτύου δεν επηρεάζονται αφού υποστηρίζουν τα συνήθη πρωτόκολλα όπως το http.

Το ht://Dig αναπτύχθηκε στο Πανεπιστήμιο του San Diego ως ένας τρόπος για έρευνα των διαφορετικών εξυπηρετητών ιστού για το δίκτυο της πανεπιστημιούπολης.

Παρακάτω παρουσιάζονται κάποια από τα χαρακτηριστικά αυτού του συστήματος.

- Αναζήτηση σε ενδοδίκτυο: Το ht://Dig έχει τη δυνατότητα να ερευνά μέσω πολλών κεντρικών υπολογιστών ένα δίκτυο δρώντας ως ένας διαφυλλιστή WWW.
- Είναι ελεύθερο: Ολόκληρο το σύστημα έχει άδεια από το GNU General Public License.
- Υποστηρίζει τη σύσταση robots.txt.
- Αναζήτηση με εκφράσεις Boolean: Οι αναζητήσεις μπορούν να είναι σύνθετες χρησιμοποιώντας τις εκφράσεις Boolean.
- Διαμόρφωση των αποτελεσμάτων της αναζήτησης: Η έξοδος μιας αναζήτησης μπορεί εύκολα να προσαρμοστεί στις ανάγκες του χρήστη με τη βοήθεια της παροχής των προτύπων HTML.

- Ασαφή αναζήτηση: Οι αναζητήσεις μπορούν να εκτελεστούν χρησιμοποιώντας τους διάφορους διαμορφώσιμους αλγορίθμους. Αυτήν την περίοδο οι ακόλουθοι αλγόριθμοι υποστηρίζονται (με οποιοδήποτε συνδυασμό):
    - μεθοδικό (exact)
    - ηχητικό (soundex)
    - μετακουστικό (metaphone)
    - κοινές καταλήξεις λέξεων (ρίζες)
    - συνώνυμες λέξεις
    - συλλαβισμός
    - υπό-συμβολοσειρές και προθέματα
- Αναζήτηση για αρχεία κειμένων και HTML: Και τα έγγραφα HTML και τα απλά αρχεία κειμένων μπορούν να αναζητηθούν.
- Λέξεις κλειδιά μπορούν να προστεθούν σε έγγραφα HTML: Οποιοσδήποτε αριθμός λέξεων κλειδιών μπορεί να προστεθεί στα έγγραφα HTML. Αυτή η εισαγωγή των λέξεων κλειδιών χρησιμοποιείται για να βρίσκετε πιο εύκολα το έγγραφο και επίσης για να εμφανίζεται ψηλότερα στον κατάλογο των αποτελεσμάτων.
- Κοινοποίηση ηλεκτρονικού ταχυδρομείου των εγγράφων που έχουν λήξει: Οι ειδικές μετά-πληροφορίες μπορούν να προστεθούν στα έγγραφα HTML που μπορούν να χρησιμοποιηθούν για να ειδοποιήσουν το συντηρητή για αυτά τα έγγραφα σε έναν ορισμένο χρόνο.
- Ένας προστατευμένος κεντρικός υπολογιστής μπορεί να συνταχθεί: Το ht://Dig μπορεί να ειπωθεί ότι χρησιμοποιεί ένα συγκεκριμένο όνομα χρήστη και έναν κωδικό πρόσβασης όταν ανακτά τα έγγραφα. Αυτό μπορεί να χρησιμοποιηθεί για να συντάξει το ευρετήριο ένας κεντρικός υπολογιστής.
- Αναζητήσεις στις υποενότητες της βάσης δεδομένων: Είναι εύκολο να οργανωθεί μια αναζήτηση που επιστρέφει μόνο τα έγγραφα των οποίων το URL ταιριάζει με ένα ορισμένο σχέδιο. Αυτό είναι ειδικότερα χρήσιμο για τους ανθρώπους που θέλουν να καταστήσουν τα στοιχεία τους εξερευνήσιμα χωρίς να πρέπει να χρησιμοποιηθεί μια χωριστή μηχανή ή μια βάση δεδομένων αναζήτησης.

- Συμπεριλαμβάνεται πλήρης κώδικας πηγής: Η μηχανή αναζήτησης έρχεται με τον πλήρη κώδικα πηγής. Ολόκληρο το σύστημα απελευθερώνεται υπό τους όρους και τις διατάξεις της GNU Public License έκδοση 2.0.
- Το βάθος της αναζήτησης μπορεί να περιοριστεί: Αντί του περιορισμού της αναζήτησης σε ένα σύνολο μηχανών, μπορεί επίσης να περιορίσει την αναζήτηση σε έγγραφα που είναι ορισμένα κλικ του ποντικιού μακριά από το έγγραφο έναρξης.
- Πλήρης υποστήριξη για το σύνολο χαρακτήρα ISO-Latin-1: Και οι δύο οντότητες SGML όπως το "&agrave" και οι χαρακτήρες ISO-Latin-1 μπορούν να συνταχθούν και να αναζητηθούν.

Το σύστημα αυτό έχει και κάποιες απαιτήσεις. Το ht://Dig αναπτύχθηκε από το Unix χρησιμοποιώντας τη C ++. Για αυτόν τον λόγο, όποιος θελήσει να εργαστεί με αυτό το σύστημα θα χρειαστεί μια μηχανή Unix, το μεταγλωττιστή της C και ένα μεταγλωττιστή C++. (Ο μεταγλωττιστής C απαιτείται για να συντάξει μερικές από τις βιβλιοθήκες της GNU).

Δυστυχώς οι υπεύθυνοι για την ανάπτυξη έχουν μόνο πρόσβαση σε μερικές διαφορετικές μηχανές Unix. Η περισσότερη ανάπτυξη γίνεται σε συστήματα Linux με gcc/g++, αλλά το ht://Dig έχει ελεγχθεί στις μηχανές (και τους μεταγλωττιστές) που φαίνονται παρακάτω:

- Sun Solaris SPARC 2.X (που χρησιμοποιεί gcc/g ++)
- Sun SunOS 4.1.4 SPARC (που χρησιμοποιεί gcc/g ++ 2.7.0)
- HP/UX 10.X (που χρησιμοποιεί gcc/g ++)
- IRIX 5,3 και 6.X (μεταγλωττιστής SGI g ++.)
- Οι περισσότερες διανομές Linux (που χρησιμοποιούν gcc/g ++)
- Οι περισσότερες πλατφόρμες BSD, συμπεριλαμβανομένου BSDI και της MAC OS X (που χρησιμοποιούν gcc/g ++)

Υπάρχουν εκθέσεις ht://Dig που λειτουργούν σε διάφορες άλλες πλατφόρμες.

Το σύστημα ht://Dig έχει και απαιτήσεις στο χώρο του δίσκου του υπολογιστικού συστήματος. Η μηχανή αναζήτησης θα απαιτήσει μέρη από το χώρο του δίσκου για να αποθηκεύσει τις βάσεις δεδομένων της. Δυστυχώς, δεν υπάρχει κανένας ακριβής τύπος για να υπολογίσει τις απαιτήσεις του χώρου. Εξαρτάται από τον αριθμό εγγράφων που πρόκειται να χρησιμοποιηθούν για την σύνταξη του ευρετήριου αλλά και στις διάφορες επιλογές που χρησιμοποιεί ο χρήστης. (http://www.htdig.org/main.html)

1.11.3 Lemur

Το Lemur είναι ένα σύνολο εργαλείων που στόχος του είναι να διευκολύνει την έρευνα στη γλωσσική διαμόρφωση και την ανάκτηση πληροφοριών (IR), όπου ο όρος IR ερμηνεύεται ευρέως για να περιλάβει τεχνολογίες όπως η ad hoc (συγκεκριμένου σκοπού) και η διανεμημένη ανάκτηση, με δομημένες ερωτήσεις, το διαγλωσσικό IR, την περιληπτική παρουσίαση της πληροφορίας, το φιλτράρισμα, και την κατηγοριοποίηση. Η ελλοχεύουσα αρχιτεκτονική του συστήματος χτίστηκε για να υποστηρίξει τις παραπάνω τεχνολογίες. Παρέχονται πολλές χρήσιμες εφαρμογές δειγμάτων, αλλά έχει σχεδιαστεί έτσι ώστε το σύνολο εργαλείων του να επιτρέπει στους χρήστες του να προγραμματίζουν εύκολα τις προσαρμογές και τις αιτήσεις τους.

Το σύνολο εργαλείων Lemur υποστηρίζει την κατασκευή των βασικών συστημάτων ανάκτησης κειμένων χρησιμοποιώντας τις μεθόδους γλωσσικής διαμόρφωσης, καθώς επίσης και τις παραδοσιακές μεθόδους όπως εκείνες που βασίζονται στο διανυσματικό πρότυπο και το Okapi. Δεδομένου ότι το σύνολο εργαλείων εξελίσσεται, αναμένεται ότι θα υποστηρίζει πλέον και την αναζήτηση στην ευρύτερη σειρά των τεχνολογιών πληροφοριών όπως το φιλτράρισμα, και ακόμη την απάντηση σε ερώτηση.

Το Lemur είναι ιδιαίτερα χρήσιμο για τους ερευνητές στη γλωσσική διαμόρφωση και την ανάκτηση πληροφοριών που δεν θέλουν να χρησιμοποιήσουν τους καταχωρητές τους αλλά θα εστίαζαν στην ανάπτυξη των νέων τεχνικών και των αλγορίθμων. Εντούτοις, εκτός από την ευρετηρίαση, παρέχει και μερικούς αλγορίθμους ανάκτησης βασικών γραμμών.

Κάποιος μπορεί να χρησιμοποιήσει το Lemur για να χτίσει τα συστήματα αναζήτησής του. Έχει εφαρμοστεί και περιέχει το βασικό IR, το διανεμημένο IR, το IR που χρησιμοποιεί τις δομημένες ερωτήσεις, το IR που χρησιμοποιεί διανεμημένα ευρετήρια, που συγκεντρώνονται στα έγγραφα, και την περιληπτική παρουσίαση της πληροφορίας. Άλλοι έχουν χρησιμοποιήσει το Lemur για φιλτράρισμα, εύρεση ιστοσελίδας, εύρεση μεταβάσεων, και μηχανές αναζήτησης Ιστού.

Το Lemur έχει πολλές εφαρμογές για την ευρετηρίαση και την ανάκτηση που είναι πλήρως λειτουργικές για πολλούς λόγους, έτσι μπορεί να τις χρησιμοποιήσει

κάποιος εκτός του συστήματος. Επιπλέον, δεδομένου ότι το Lemur γράφτηκε για να διευκολύνει την έρευνα για LM και IR, το σχέδιο επιτρέπει στους χρήστες να δοκιμάσουν νέες μεθόδους ανάκτησης με αφηρημένες διεπαφές, ή να γράψουν τις νέες εφαρμογές βασισμένες στις υπάρχουσες μεθόδους.

Το Lemur 'έρχεται' με όλο τον κώδικα πηγής και φτιάχνει τα απαραίτητα αρχεία για να χτίσει τις βιβλιοθήκες για την ευρετηρίαση και την ανάκτηση (στο πλαίσιο της συμφωνίας μιας χορήγησης αδειών CMU και UMass). Για τα Windows, μπορεί, κάποιος, να μεταφορτώσει τις προ-μεταγλωτιστικές βιβλιοθήκες και να τις εκτελέσει.

Το Lemur υποστηρίζει αυτήν την περίοδο τα ακόλουθα χαρακτηριστικά γνωρίσματα:

- Συντάσσει ευρετήριο:
  - αγγλικό, κινεζικό και αραβικό κείμενο
  - ομόριζα
  - παραλείπει τις συνήθεις λέξεις (stopword, όπως άρθρα, συνδέσμους)
  - αναγνωρίζει τα αρκτικόλεξα
  - συμβολικές ιδιότητες επιπέδων, όπως το μέρος του λόγου και τις ονομασμένες οντότητες
  - ευρετηρίαση μεταβάσεων
  - επαυξητική ευρετηρίαση
  - ευθύγραμμη και αντισταθμισμένη υποστήριξη σχολιασμών
- Ανάκτηση: ειδική ανάκτηση (TF/IDF, Okapi, και InQuery)
  - ανάκτηση μεταβάσεων
  - διαγλωσσική ανάκτηση
  - γλωσσική διαμόρφωση (KL-απόκλιση)
  - ανατροφοδότηση σχετικότητας
  - δομημένη γλώσσα διατύπωσης ερωτήσεων
  - βασισμένο σε παραγωγικές καταλήξεις ταίριασμα όρου μπαλαντέρ
- Κατανεμημένο IR:
  - ερώτηση βασισμένη στη δειγματοληψία
  - ταξινόμηση βάσεων δεδομένων (CORI)
  - συγχώνευση αποτελεσμάτων (CORI, ενιαίες οπισθοδρόμηση και συγχώνευση πολυ-οπισθοδρόμησης)
- Συγκέντρωση εγγράφων

- Περιληπτική παρουσίαση της πληροφορίας
- Απλή επεξεργασία κειμένων

Το σύνολο εργαλείων Lemur που είναι διαθέσιμο για μεταφόρτωση επίσης περιλαμβάνει ένα μικρό αρχείο στοιχείων δειγμάτων με τα χειρόγραφα δοκιμής που χρησιμοποιούν τις αιτήσεις του.

Το Lemur γράφτηκε πρώτιστα στη C++, (το GUI γράφεται με την Java/Swing.). Είναι συμβατό με τα Windows XP, Linux (συμπεριλαμβανομένου x86_64), MAC OS/X και συστήματα βασισμένα σε Solaris. Αν και δεν υποστηρίζονται αυτήν την περίοδο επίσημα, πληροφορίες από κάποια άτομα δηλώνουν ότι έχει τρέξει επίσης το σύνολο εργαλείων Lemur σε Cygwin, τα Windows 2000, και τα Windows NT. (http://www.lemurproject.org/lemur/)

1.11.4 Sphinx

Το Sphinx είναι μια μηχανή αναζήτησης ολόκληρων κειμένων, που διανέμεται από την έκδοση GPL 2. Οι εμπορικές άδειες (π.χ. για την ενσωματωμένη χρήση) είναι επίσης διαθέσιμες κατόπιν αιτήσεως.

Γενικά, είναι αυτόνομη μηχανή αναζήτησης, εξασφαλίζει ότι θα τρέξει γρήγορα, η απόδοσή της είναι ανάλογη του μεγέθους και είναι κατάλληλη για συναρτήσεις αναζήτησης ολόκληρων κειμένων σε άλλες εφαρμογές. Το Sphinx είχε σχεδιαστεί ειδικά έτσι ώστε να ενσωματώνεται σε βάσεις δεδομένων SQL και περιγραφικές γλώσσες.

Αυτήν την περίοδο οι ενσωματωμένοι οδηγοί δεδομένων του κώδικα υποστηρίζουν τα εξαγόμενα δεδομένα είτε μέσω της άμεσης σύνδεσης με MySQL, είτε PostgreSQL, είτε χρησιμοποιώντας ένα μηχανισμό σωλήνα με ένα σχήμα XML. Η προσθήκη των νέων οδηγών έχει ως σκοπό να είναι όσο το δυνατόν ευκολότερη.

Η αναζήτηση API είναι ένα μέσο εισόδου-εξόδου σε PHP, Python, Perl, Ruby, Java, και επίσης είναι διαθέσιμο ως pluggable μηχανή αποθήκευσης MySQL. Όσον αφορά στο όνομα, Sphinx είναι ένα αρκτικόλεξο που αποκωδικοποιείται επίσημα ως SQL Phrase Index.

Τα χαρακτηριστικά γνωρίσματα του Sphinx είναι τα εξής:

- Υψηλή ταχύτητα ευρετηρίασης (μέχρι 10 MB/SEC σε σύγχρονο CPUs)
- Υψηλή ταχύτητα αναζήτησης (avg η ερώτηση είναι κάτω από 0,1 sec σε 2-4GB συλλογές κειμένων)

- Υψηλή εξέλιξη (μέχρι 100 GB του κειμένου, μέχρι 100 M έγγραφα σχετικά με μια ενιαία CPU)
- Παρέχει καλή συμπεριφορά ταξινόμησης μέσω του συνδυασμού φράσεων εγγυάται την ταξινόμηση και τη στατιστική ταξινόμηση
- Παρέχει διανεμημένες ικανότητες αναζήτησης
- Παρέχει την παραγωγή εγγράφων
- Παρέχει την έρευνα μέσω MySQL μέσω της pluggable μηχανής αποθήκευσης
- Υποστηρίζει ερωτήσεις τύπου boolean, φράσεων, και λέξεων
- Υποστηρίζει πολλαπλούς τομείς ολοκληρωμένων κειμένων ανά έγγραφο
- Υποστηρίζει πολλαπλές πρόσθετες ιδιότητες ανά έγγραφο (όπως ομάδες, timestamps, κλπ.)
- Υποστηρίζει την χρήση των κοινών λέξεων (stopwords)
- Υποστηρίζει και τις κωδικοποιήσεις ενιαίων-byte και UTF-8
- Υποστηρίζει τις ομόριζες λέξεις για την αγγλική γλώσσα και τη ρωσική, και Soundex για τη μορφολογία
- Υποστηρίζει MySQL
- Υποστηρίζει PostgreSQL.

Το Sphinx είναι διαθέσιμο μέσω του επίσημου ιστοχώρου του στην http://www.sphinxsearch.com/.

Η ανάπτυξη του Sphinx άρχισε το 2001, επειδή ο δημιουργός της δεν κατόρθωσε να βρει μια αποδεκτή λύση αναζήτησης (για έναν προσανατολισμένο προς τη βάση δεδομένων ιστοχώρο) που θα κάλυπτε τις απαιτήσεις του. Πραγματικά, κάθε σημαντική πτυχή ήταν ένα πρόβλημα:

- ποιότητα αναζήτησης (δηλ. καλή σχετικότητα)
- ταχύτητα αναζήτησης
- μέτριες απαιτήσεις για το δίσκο και τη CPU κατά την ευρετηρίαση

Τα περισσότερα σύγχρονα συστήματα Unix με το μεταγλωττιστή C++ πρέπει να είναι σε θέση να συντάξουν και να τρέξουν Sphinx χωρίς οποιεσδήποτε τροποποιήσεις.

Αυτήν την περίοδο γνωστά συστήματα Sphinx έχουν τρέξει επιτυχώς επάνω σε:

- Linux 2.4.x, 2.6.x
- Windows 2000, XP
- FreeBSD 4.x, 5.x, 6.x

- NetBSD 1.6, 3.0
- Solaris 9, 11
- MAC OS X

(http://sphinxsearch.com/)

1.11.5 MG

Το MG είναι ένα ελεύθερα-διαθέσιμο σύστημα ανάκτησης ολοκληρωμένων κειμένων που καθιστά την αποδοτική χρήση των πόρων του δίσκου με την αποθήκευση του ευρετηρίου, και το κείμενο στο οποίο δημιουργούνται ευρετήρια, σε συμπιεσμένη μορφή. Το σύστημα MG περιγράφεται σε επώνυμα βιβλία Managing Gigabytes. Για να ικανοποιηθούν οι ανάγκες της ψηφιακής βιβλιοθήκης της Νέας Ζηλανδίας και του λογισμικού Greenstone έχουν γίνει διάφορες βελτιώσεις στην τυποποιημένη διανομή MG. Αυτές οι βελτιώσεις παρέχονται ως MG-1.3.

Το MG-1.3 προσφέρει:

- Εύκαμπτες μεθόδους ομόριζων

  Διαφορετικές μέθοδοι ομόριζων μπορούν να χρησιμοποιηθούν στην ίδια συλλογή. Στην αρχική έκδοση που μια συλλογή θα μπορούσε μόνο να χρησιμοποιήσει μια ενιαία μέθοδο ομόριζων, τώρα και οι τέσσερις συνδυασμοί ομορίζων μπορούν να χρησιμοποιηθούν στις ερωτήσεις. Αυτό μπορεί να εφαρμοστεί σε ολόκληρες ερωτήσεις καθώς επίσης και σε μεμονωμένες λέξεις.

- Στάθμιση των όρων

  Η στάθμιση μπορεί να εφαρμοστεί στους μεμονωμένους όρους σε μια ταξινομημένη ερώτηση. Αυτό επιτρέπει σε έναν χρήστη να επηρεάζει τη θέση των έγγραφων που έχουν υψηλότερες εκτιμώμενες λέξεις έτσι ώστε να επιστρέφονται πλησιέστερα στην κορυφή.

- Συχνότητες όρου

  Ο αριθμός χρόνων που κάθε όρος ερώτησης εμφανίζεται στη συλλογή μπορεί να επιδειχθεί κατά τη διάρκεια μιας ερώτησης.

- Αποσυμπίεση του κειμένου

  Ένα μικρό πρόγραμμα συμπεριλαμβάνεται στη διανομή που επιτρέπει στους χρήστες να αποσυμπιέσουν όλο το κείμενο με τα έγγραφα και τις

παραγράφους που παρεμβάλλονται σε όλες τις σωστές θέσεις. Με ελάχιστο προγραμματισμό θα μπορέσουν έπειτα να ξαναφτιάξουν τη συλλογή από το συμπιεσμένο κείμενο.

- Ανεξάρτητα ευρητήρια μηχανών

  Τα ευρετήρια πάντα γράφονται χρησιμοποιώντας τη τάξη δικτύων byte, η οποία είναι μεγάλη endian. Αυτό σημαίνει ότι τα ευρετήρια που στηρίζονται στις μικρές endian μηχανές μπορούν να χρησιμοποιηθούν στις μεγάλες endian μηχανές που παρέχουν ότι οι δύο μηχανές έχουν τύπο δεδομένων ίδιου μήκους και χρησιμοποιούν τα ίδια σχήματα κινητής υποδιαστολής.

- Μέσο εισόδου-εξόδου στο MS-DOS

  Ένα μέσο εισόδου-εξόδου του MG στο MS-DOS με επεξεργαστή 32bit έχει ενσωματωθεί επίσης στην έκδοση. Αυτό το μέσο επιτρέπει στο MG να συνταχθεί σε πλατφόρμες Windows των 32bit.

(http://www.nzdl.org/html/mg.html)

1.11.6 DataparkSearch

Το DataparkSearch είναι μια μηχανή αναζήτησης Ιστού πλήρως χαρακτηρισμένη. Το DataparkSearch αποτελείται από δύο μέρη. Το πρώτο μέρος παρέχει το μηχανισμό για την σύνταξη των ευθρετηρίων. Ο καταχωρητής ‘περπατά’ πέρα από τις αναφορές υπερκειμένων και αποθηκεύει τις λέξεις και τις νέες αναφορές στη βάση δεδομένων. Το δεύτερο μέρος είναι front-end της CGI Ιστού για να παρέχει την αναζήτηση, που χρησιμοποιεί τα δεδομένα που συλλέχθηκαν από τον καταχωρητή.

Το DataparkSearch κλωνοποιήθηκε από την έκδοση CVS 3.2.16 του mnoGoSearch στις 27 Νοεμβρίου 2003 ως η DataparkSearch 4.16. Η πρώτη έκδοση mnoGoSearch πραγματοποιήθηκε τον Νοέμβριο του 1998. Η μηχανή αναζήτησης είχε το όνομα UDMSearch μέχρι τον Οκτώβριο του 2000 όταν αποκτήθηκε το πρόγραμμα από την εταιρία COM Lavtech και άλλαξε το όνομά του σε mnoGoSearch. Το πιο πρόσφατο changelog μπορεί να βρεθεί στον εξής ιστοχώρο http://www.dataparksearch.org/ChangeLog.

Τα κύρια χαρακτηριστικά γνωρίσματα του DataparkSearch είναι τα ακόλουθα:

- MySQL (απαιτείται η libz βιβλιοθήκη), PostgreSQL, iODBC, unixODBC, γέφυρα EasySoft ODBC-ODBC, InterBase, Oracle, υποστήριξη MS SQL back-ends.
- Υποστήριξη HTTP.
- Υποστήριξη proxy HTTP.
- Υποστήριξη HTTPS.
- Υποστήριξη FTP.
- Υποστήριξη NNTP
- Υποστήριξη σχεδίου HTDB εικονική URL. Κάποιος μπορεί να χτίσει το ευρετήριο και την αναζήτηση μέσω των μεγάλων τομέων κειμένων από τη βάση δεδομένων SQL.
- Αντανακλά τα χαρακτηριστικά γνωρίσματα.
- Παρέχει ενσωματωμένη υποστήριξη για text/html, text/xml, text/plain, audio/mpeg (MP3) και image/gif .
- Υποστήριξη εξωτερικών κατατμητών για άλλους τύπους εγγράφων.
- Δυνατότητα να συνταχθούν πολύγλωσσες περιοχές που χρησιμοποιούν την ικανοποιημένη διαπραγμάτευση.
- Αναζήτηση όλων των μορφών λέξεων που χρησιμοποιούν ispell επισυνάψεις και λεξικά
- Βασική υποστήριξη έγκρισης. Κάποιος μπορεί να δημιουργήσει ευρετήρια προστατευμένα με κωδικό πρόσβασης σε κεντρικούς υπολογιστές HTTP ενδοδικτύου.
- Ικανότητα επανεισόδου. Κάποιος μπορεί να χρησιμοποιήσει διάφορες διαδικασίες σύνταξης ευρετηρίων και έρευνας συγχρόνως ακόμα και στην ίδια βάση δεδομένων.
- Πολύπλοκη υποστήριξη δημιουργίας ευρετηρίων.
- Υποστήριξη κοινών λέξεων (stop list).
- Υποστήριξη <META NAME= “robots” content= “...”> και robots.txt
- Η γλώσσα C του front-end Ιστού της CGI.
- Υποστήριξη της γλώσσας ερωτήσεων Boolean.

- Τα αποτελέσματα της ταξινόμησης εμφανίζονται ταξινομημένα ως εξής σύμφωνα με την εφαρμογή, τη δημοφιλή θέση, τη τελευταία ημερομηνία τροποποίησης και τη σημασία.
- Συγκεχυμένη αναζήτηση: διαφορετικές μορφές λέξης, διορθώσεις συλλαβισμού, συνώνυμα, αρκτικόλεξα και συντμήσεις.
- Υποστήριξη διάφορων συνόλων χαρακτήρων.
- Πρότυπα HTML για να προσαρμόσει εύκολα τα αποτελέσματα αναζήτησης.
- Προηγμένες επιλογές αναζήτησης όπως τα χρονικά όρια, την κατηγορία και τα όρια ετικετών κλπ.
- Υποστηρίζει την τμηματοποίηση των φράσεων για τις εξής γλώσσες: κινέζικα, ιαπωνικά, κορεάτικα και ταϊλανδέζικα.
- mod_dpsearch - ενότητα αναζήτησης για τον κεντρικό υπολογιστή δικτύου Apache (http://httpd.apache.org/).
- Υποστήριξη των Διεθνοποιημένων περιοχών ονομάτων (Internationalized Domain Names) .
- Συνοπτικός αλγόριθμος εξαγωγής (Summary Extraction Algorithm (SEA)).

(http://www.dataparksearch.org/)

1.11.7 Wumpus

Το Wumpus είναι ένα σύστημα ανάκτησης πληροφοριών που αναπτύχθηκε στο Πανεπιστήμιο του Βατερλώ. Ο κύριος σκοπός του είναι να μελετήσει τα ζητήματα που προκύπτουν στα πλαίσια της σύνταξης ευρετηρίων στις δυναμικές συλλογές κειμένων στο περιβάλλον πολλών χρηστών.

Η προοριζόμενη χρήση Wumpus είναι διπλή. Μπορεί να χρησιμοποιηθεί:

- ως συνηθισμένο σύστημα ανάκτησης πληροφοριών, με την υποστήριξη της ενεργοποίησης ή απενεργοποίησης των πολλών χρηστών
- ως ένα σύστημα αρχείων δημιουργίας ευρετηρίων που παρακολουθεί αυτόματα όλες τις αλλαγές στο σύστημα αρχείων και ενημερώνει το ευρετήριο αναλόγως.

Το Wumpus είναι εξελικτικό σύστημα και έχει χρησιμοποιηθεί στις συλλογές κειμένων που αποτελούνται από πολλές εκατοντάδες gigabytes κειμένου και που περιέχουν δωδεκάδες εκατομμυρίων έγγραφα.

Το Wumpus είναι ελεύθερα διαθέσιμο υπό την άδεια του GNU.(http://www.wumpus-search.org/)

1.11.8 Zettair

Το Zettair είναι μια συμπαγής και γρήγορη μηχανή αναζήτησης κειμένων που σχεδιάστηκε από την ομάδα μηχανών αναζήτησης στο πανεπιστήμιο RMIT. Ήταν γνωστός στο παρελθόν ως Lucy.

Το Zettair επιτρέπει στους χρήστες να συντάσσουν ευρετήρια και να ψάχνουν στις συλλογές HTML (ή TREC). Έχει σχεδιαστεί για την απλότητα καθώς επίσης και την ταχύτητα και την ευελιξία, και το αρχικό χαρακτηριστικό γνώρισμά του είναι η δυνατότητα να αντιμετωπίζει τα μεγάλα ποσά κειμένου. Έχει έναν ενιαίο εκτελεστή, ο οποίος εκτελεί και την σύνταξη των ευρετηρίων και την αναζήτηση: όταν ένα ευρετήριο δεν υπάρχει, το Zettair θα δημιουργήσει ένα βασισμένο στις παραμέτρους που παρέχει ο χρήστης, και όταν υπάρχει ένα, το Zettair θα το χρησιμοποιήσει για να ψάξει τα στοιχεία για τα οποία έχουν ήδη δημιουργηθεί ευρετήρια. Έχει μια απλή διεπαφή εντολών, και υποστηρίζει τις ταξινομημένες, απλές Boolean, και φράσεων ερωτήσεις.

Στα χαρακτηριστικά γνωρίσματα του Zettair περιλαμβάνονται:

- Ταχύτητα και εξελιξιμότητα
- Ερωτήσεις τύπου boolean, ταξινομήσεις και φράσεις
- Αποτελούμενο από υπομονάδες C API για το συνυπολογισμό σε άλλα προγράμματα
- Εγγενής υποστήριξη για τα πειράματα TREC

Από τη 0.6.1, η μηχανή αναζήτησης Zettair έχει εξεταστεί στις ακόλουθες πλατφόρμες:

- Linux
- FreeBSD
- MAC OS X (Δαρβίνος)
- Solaris

- Win32 (WINDOWS 95, 98, NT, κ.λ.π.)
- Cygwin

Το Zettair γράφτηκε στη C και έχει χορηγημένη άδεια από το BSD. (http://www.seg.rmit.edu.au/zettair/)

1.11.9 Xapian

Το Xapian είναι μια ανοικτού κώδικα μηχανή αναζήτησης, που εκδίδεται κάτω από την επίβλεψη του GPL. Γράφεται σε C++, με επιπλέον διασυνδέσεις που επιτρέπουν τη χρήση της και στις εξής γλώσσες Perl, Python, PHP, Java, Tcl, C# και Ruby.

Το Xapian είναι ένα ιδιαίτερα προσαρμόσιμο σύνολο εργαλείων που επιτρέπει στους επιστήμονες να προσθέσουν εύκολα τις προηγμένες συντάξεις ευρετηρίων και αναζήτησης στις εφαρμογές τους. Υποστηρίζει το πιθανολογικό πρότυπο ανάκτησης πληροφοριών και επίσης ένα πλούσιο σύνολο ερωτήσεων Boolean.

Εάν κάποιος θέλει μια συσκευασμένη μηχανή αναζήτησης για τον ιστοχώρο του, πρέπει να ρίξει μια ματιά στο Omega: μια εφαρμογή που χτίζεται επάνω στην Xapian. Σε σύγκριση με τις άλλες λύσεις που παρέχονται για την αναζήτηση σε ένα ιστόχωρο, η μεταβλητότητα της Xapian επιτρέπει στους χρήστες να επεκτείνουν το Omega για να ικανοποιήσει τις αυξανόμενες ανάγκες τους.

Το Xapian είναι μια σύγχρονη βιβλιοθήκη, αλλά έχει εξελιχθεί χάρις την πολύχρονη εμπορική και ακαδημαϊκή εμπειρία του.

Στα αξιοσημείωτα χαρακτηριστικά γνωρίσματα του Xapian περιλαμβάνονται τα εξής:

- Ελεύθερο λογισμικό/ανοικτού κώδικα, που η άδεια του χορηγείται από την GPL.
- Υποστηρίζει Unicode και αποθηκευμένα δεδομένα σε ευρετήρια σε UTF-8.
- Ιδιαίτερα φορητός, καθώς τρέχει σε Linux, MAC OS X, FreeBSD, NetBSD, OpenBSD, Solaris, HP -HP-UX, Tru64, IRIX, και πιθανώς σε άλλες πλατφόρμες Unix, καθώς επίσης και στα Windows της Microsoft.
- Γράφετε σε C++. Οι συνδέσεις Perl είναι διαθέσιμες στην ενότητα Search::Xapian σε CPAN. Οι συνδέσεις της Java JNI

συμπεριλαμβάνονται στην ενότητα συνδέσεων Xapian. Υποστηρίζει επίσης SWIG που μπορεί να παραγάγει τις συνδέσεις για πολλές γλώσσες. Αυτή τη στιγμή λειτουργεί για Python, PHP, TCL, C#, και Ruby.

- Ταξινομημένη πιθανολογική αναζήτηση: οι σημαντικές λέξεις παίρνουν περισσότερο βάρος από τις ασήμαντες λέξεις, έτσι τα πιο σχετικά έγγραφα βρίσκονται στην αρχή του καταλόγου με τα αποτελέσματα.
- Ανατροφοδότηση σχετικότητας, λαμβάνοντας υπόψη ένα ή περισσότερα έγγραφα: η Xapian μπορεί να προτείνει τους πιο σχετικούς όρους ευρετηρίων για να επεκτείνει μια ερώτηση, να προτείνει τα σχετικά έγγραφα, να ταξινομήσει τα έγγραφα, κλπ.
- Έρευνα φράσης και εγγύτητας.
- Πλήρες φάσμα δομημένων χειριστών αναζήτησης τύπου Boolean. Τα αποτελέσματα της boolean αναζήτησης ταξινομούνται από τα πιθανοκρατικά βάρη. Τα boolean φίλτρα μπορούν επίσης να εφαρμοστούν για να περιορίσουν μια πιθανολογική αναζήτηση.
- Υποστηρίζει τους όρους αναζήτησης (π.χ. μια αναζήτηση για την λέξη 'του ποδοσφαίρου' θα ταίριαζε με τα έγγραφα που αναφέρουν 'το ποδόσφαιρο' ή 'τον ποδοσφαιριστή'). Αυτό βοηθά να βρεθούν τα σχετικά έγγραφα, που ειδάλλως δε θα εμφανίζονταν. Η χρήση των ομόριζων συμπεριλαμβάνονται αυτήν την περίοδο για τις εξής γλώσσες δανικά, ολλανδικά, αγγλικά, φιλανδικά, γαλλικά, γερμανικά, ουγγρικά, ιταλικά, νορβηγικά, πορτογαλικά, ρουμάνικα, ρωσικά, ισπανικά, σουηδικά, και τούρκικα.
- Υποστηρίζεται η αναζήτηση μπαλαντέρ (π.χ. "xap *").
- Τα συνώνυμα υποστηρίζονται.
- Υποστηρίζει αρχεία βάσης δεδομένων > 2GB, κάτι το οποίο είναι ουσιαστικό για την κλιμάκωση στις μεγάλες συλλογές εγγράφων.
- Ανεξάρτητες πλατφόρμες για σχήματα δεδομένων, δηλαδή μπορεί να στηθεί μια βάση δεδομένων σε μια μηχανή και να γίνει η αναζήτηση σε άλλη.
- Επιτρέπει την ταυτόχρονη αναπροσαρμογή και την έρευνα. Τα νέα έγγραφα γίνονται εξερευνήσιμα από μακριά.

(http://www.xapian.org/)

1.11.10 Zebra

Το Zebra είναι ένα ελεύθερο, γρήγορο, φιλικό σύστημα διαχείρισης πληροφοριών. Μπορεί να συντάξει ευρετήρια σε αρχεία όπως XML/SGML, MARC, αρχεία ηλεκτρονικού ταχυδρομείου και πολλά άλλα σχήματα, και να τα βρίσκει γρήγορα χρησιμοποιώντας έναν συνδυασμό της boolean έρευνας και ταξινόμησης σχετικότητας. Οι εφαρμογές αναζήτησης και ανάκτησης μπορούν να γραφτούν χρησιμοποιώντας APIs σε μια ευρεία ποικιλία των γλωσσών, επικοινωνώντας με το κεντρικό υπολογιστή της Zebra χρησιμοποιώντας πρωτόκολλα βιομηχανικής ανάκτησης, δηλαδή τυποποιημένη πληροφορία, ή υπηρεσίες Ιστού.

Το Zebra είναι ανοικτού κώδικα, και μπορεί να επεκταθεί από τον καθένα για οποιοδήποτε σκοπό χωρίς παροχή αδειών. Ο κώδικας της C είναι ανοικτός σε οποιονδήποτε για να διαβαστεί και να αλλαχθεί με άδεια από τη GPL.

Το Zebra είναι ένα συστατικό δικτύου που ενεργεί ως αξιόπιστος κεντρικός υπολογιστής, ο Z39.50, και για να καταγράφει και να τεκμηριώνει την αναζήτηση, παρουσιάζει, εισάγει, ενημερώνει και διαγράφει τις διαδικασίες.

Το Zebra είναι διαθέσιμο σε κεντρικό υπολογιστή MS Windows 2003 (32bit) καθώς επίσης και σε GNU/Debian Linux (32bit και 64bit). Έχει επεκταθεί επιτυχώς σε άλλα συστήματα Unix, συμπεριλαμβανομένου των Sun Sparc, HP Unix, και πολλές παραλλαγές Linux και συστήματα βασισμένα σε BSD.

Το Zebra είναι μια υψηλής απόδοσης, γενικής χρήσης δομημένη μηχανή δημιουργίας ευρετηρίων για τα κείμενα και ανάκτησης. Διαβάζει εγγραφές σε ποικίλα σχήματα εισαγωγής (π.χ. ηλεκτρονικό ταχυδρομείο, XML, MARC) και παρέχει την πρόσβαση σε αυτούς μέσω ενός ισχυρού συνδυασμού των Boolean εκφράσεων αναζήτησης και των σχετικών-ταξινομημένων ελεύθερων ερωτήσεων. Το Zebra, επίσης, υποστηρίζει μεγάλες βάσεις δεδομένων.

Οι βιβλιοθήκες εφαρμογής είναι διαθέσιμες για να επιτρέψουν στους χρήστες να γράψουν σε Perl, C, C++, Java, Tcl, Visual Basic, Python, PHP και σε ακόμα περισσότερες.

Τα κυριότερα χαρακτηριστικά του συστήματος αυτού είναι τα εξής:

- Σύνθετα ημι-δομημένα έγγραφα (όπως XML και έγγραφα GRS-1): Και τα XML και τα έγγραφα GRS-1 εκθέτουν τα DOM όπως στην εσωτερική

αντιπροσώπευση που επιτρέπει τους σύνθετους κανόνες δημιουργίας ευρετηρίων.

- Σχήματα εγγράφων εισαγωγής (όπως τα XML, SGML, κείμενο, ISO2709 (MARC)): Ένα σύστημα των φίλτρων εισαγωγής που οδηγείται από τις κανονικές εκφράσεις επιτρέπει στα περισσότερα σχήματα δεδομένων που βασίζονται στον ASCII να υποβληθούν σε επεξεργασία εύκολα. Το SGML, XML, ISO2709 (MARC), και το ακατέργαστο κείμενο υποστηρίζονται επίσης.
- Αποθήκευση εγγράφων (όπως μόνο με ευρετήρια, βασική αποθήκευση, αποθήκευση εγγράφων): Τα δεδομένα μπορούν, και έτσι γίνεται συνήθως, να εισαχθούν στην μνήμη του Zebra, αλλά το Zebra μπορεί επίσης να αναφερθεί σε εξωτερικά αρχεία.
- Γλώσσες διατύπωσης ερωτήσεων (CQL ΚΑΙ RPN/PQF)
- Σύνθετο δέντρο ερώτησης Boolean (CQL ΚΑΙ RPN/PQF): Και η CQL και η RPN/PQF επιτρέπουν στα ατομικά μέρη ερώτησης (APT) να συνδυαστούν στα σύνθετα δέντρα ερώτησης Boolean.
- Αναζήτηση τομέων (καθορίζεται από τον χρήστη): Τα ατομικά μέρη ερώτησης (APT) είναι είτε γενικά, είτε κατευθυνόμενα στους καθορισμένους ως προς τον χρήστη τομείς εγγράφων.
- Κανονικοποίηση δεδομένων (καθορίζεται από τον χρήστη): Η κανονικοποίηση των δεδομένων, η δειγματοληψία των κειμένων και οι χαρτογραφήσεις χαρακτήρων μπορούν να εφαρμοστούν κατά τη διάρκεια της σύνταξης των ευρετηρίων και της έρευνας
- Προκαθορισμένοι τύποι τομέων (καθορίζεται από τον χρήστη): Οι τομείς στοιχείων μπορούν να συνταχθούν ως φράση, λέξη στο κείμενο, αριθμητικές τιμές, URL, ημερομηνίες, και ακατέργαστα δυαδικά δεδομένα.
- Κανονικό ταίριασμα έκφρασης: Το πλήρες κανονικό ταίριασμα έκφρασης και το ‘κατά προσέγγιση ταίριασμα’ (π.χ. διορθώσεις ορθογραφικού λάθους) υποστηρίζονται.
- Αποκοπή όρου (αριστερό, δεξί, αριστερό-και-δεξί): Η ιδιότητα αποκοπής διευκρινίζει εάν οι παραλλαγές ενός ή περισσότερων χαρακτήρων επιτρέπονται μεταξύ των όρων που αναζητούνται και των όρων που ταιριάζουν, ή όχι. Η χρησιμοποίηση των ιδιοτήτων αποκοπής μη-

προεπιλογής θα διευρύνει το σύνολο των εγγράφων που ταιριάζουν με μια ερώτηση αναζήτησης.

- Συγκεχυμένες αναζητήσεις (διόρθωση ορθογραφίας): Επιπλέον, οι συγκεχυμένες αναζητήσεις εφαρμόζονται, όπου ένα ορθογραφικό λάθος στους όρους αναζήτησης αντιστοιχεί.

Το Zebra έχει επεκταθεί σε πολυάριθμες εφαρμογές, και στον ακαδημαϊκό και στον εμπορικό κόσμο. (http://www.indexdata.dk/zebra/)

1.11.11 LingPipe

Το LingPipe είναι μια συλλογή από βιβλιοθήκες Java για τη γλωσσική ανάλυση της ανθρώπινης γλώσσας.

Οι πληροφορίες που μπορούν να εξαχθούν από το LingPipe και τα εργαλεία εξόρυξης δεδομένων είναι τα εξής:

- αναφορές διαδρομής των οντοτήτων (π.χ. άνθρωποι ή πρωτεΐνες)
- αναφορές συνδέσεων οντοτήτων στις καταχωρήσεις βάσεων δεδομένων
- αποκαλύπτει τις σχέσεις μεταξύ των οντοτήτων και των ενεργειών
- ταξινομεί τις μεταβάσεις κειμένων σύμφωνα με τη γλώσσα, την κωδικοποίηση χαρακτήρα, το ύφος, το θέμα, ή το συναίσθημα
- διορθώνει την ορθογραφία όσον αφορά μια συλλογή κειμένων
- συγκεντρώνει τα έγγραφα από το υπονοούμενο θέμα και ανακαλύπτει τις σημαντικές τάσεις κατά τη διάρκεια του χρόνου και
- παρέχει την επικόλληση μέρους του λόγου και την ομαδική κατάτμηση φράσης.

Η αρχιτεκτονική του LingPipe έχει ως σκοπό να είναι αποδοτική, εξελικτική, επαναχρησιμοποιήσιμη, και δυνατή. Τα κυριότερα σημεία περιλαμβάνουν:

- Java API με τον κώδικα πηγής και τον έλεγχο των μονάδων
- πρότυπα πολυγλωσσικά, πολύ-περιοχών, πολύ-ύφους
- εκπαίδευση με τα νέα στοιχεία για τους νέους στόχους
- n-καλύτερη παραγωγή με τις στατιστικές εκτιμήσεις εμπιστοσύνης
- on-line εκπαίδευση

- τα πρότυπα ασφαλών νημάτων και οι αποκωδικοποιητές για τον συγχρονισμό της παράλληλης-ανάγνωσης αποκλειστικής-γραφής (concurrent-read exclusive-write (CREW)) και
- χαρακτήρας κωδικοποίησης I/O.

(http://alias-i.com/lingpipe/)

1.11.12 Terrier

Το Terrier, TERabyte RetrIEveR, είναι μια υψηλής απόδοσης και εξελικτική μηχανή αναζήτησης που επιτρέπει τη γρήγορη ανάπτυξη των εφαρμογών μεγάλης κλίμακας ανάκτησης, με την παροχή μιας περιεκτικής, εύκαμπτης, δυνατής και διαφανούς πλατφόρμας για την έρευνα και τον πειραματισμό στο IR. Ένα μικρό μέρος της πλατφόρμας ανάκτησης terrier παρέχεται περιοδικά ως ανοικτή πηγή λογισμικού. Το Terrier ξεκίνησε για να διευκολύνει την έρευνα στην αναζήτηση Ιστού, αλλά έχει επεκταθεί από τότε ώστε να περιλαμβάνει άλλες εφαρμογές, όπως ο υπολογιστής γραφείου και η εταιρική αναζήτηση Ιστού (αναζήτηση ενδοδικτύου).

Το Terrier εφαρμόζει τις λειτουργίες σύνταξης ευρετηρίων και ανάκτησης κατάστασης προόδου. Αυτό επίσης παρέχει μια ιδανική πλατφόρμα για τη γρήγορη ανάπτυξη των μεγάλης κλίμακας εφαρμογών ανάκτησης.

Η ανοικτή έκδοση πηγής Terrier παρέχει μια περιεκτική, εύκαμπτη, γερή, και διαφανή πλατφόρμα για την έρευνα και τον πειραματισμό στην ανάκτηση κειμένων. Η έρευνα που τίθεται στο Terrier επεκτείνεται συνεχώς προς τους νέους κλάδους του ευρύτερου τομέα ανάκτησης πληροφοριών, που κάνει το Terrier ιδανική, ισχυρή, αποτελούμενη από υπομονάδες πλατφόρμα κατάστασης προόδου για την ανάπτυξη και την αξιολόγηση των νέων εννοιών και ιδεών.

Το Terrier γράφεται σε Java. Αναπτύσσεται στο τμήμα υπολογισμού της επιστήμης, στο Πανεπιστήμιο της Γλασκώβης.

Παρακάτω, παρατίθεται ένας σύντομος κατάλογος χαρακτηριστικών γνωρισμάτων που προσφέρονται από το Terrier:

Γενικά

- Συντάσσει ευρετήριο υποστήριξης για τα κοινά σχήματα αρχείων υπολογιστών γραφείου, και για τις συνήθως χρησιμοποιημένες

ερευνητικές TREC συλλογές (π.χ. TREC CDs 1-5, WT2G, WT10G, gov, GOV2, Blogs06).

- Πολλοί τεκμηριώνουν τη στάθμιση των προτύπων, όπως πολλές παράμετρες ελεύθερης απόκλισης από τα σταθμισμένα τυχαία πρότυπα, Okapi BM25 και τη γλωσσική διαμόρφωση.
- Υποστηρίζετε η συμβατική γλώσσα διατύπωσης ερωτήσεων, συμπεριλαμβανομένων των φράσεων, και όροι που εμφανίζονται στις ετικέτες.
- Χειρισμός σύνταξης ευρετηρίων σε ολόκληρα κείμενα συλλογών εγγράφων μεγάλης κλίμακας, σε μια αρχιτεκτονική με τουλάχιστον 25 εκατομμύρια έγγραφα.
- Έρευνα ενεργής ανάκτησης πληροφοριών που τροφοδοτείται στην ανοικτή πλατφόρμα κώδικα.
- Ανοικτή πηγή (δημόσια άδεια Mozilla).
- Γράφετε σε Java, εργασίες για τα Windows, MAC OS X, Linux και Unix.

Δημιουργία Ευρετηρίου

- Η εκτός του πλαισίου του συστήματος δημιουργία ευρετηρίου των συλλογών εγγράφων, όπως οι συλλογές δοκιμής TREC.
- Η εκτός του πλαισίου του συστήματος δημιουργία ευρετηρίου για έγγραφα των διάφορων σχημάτων, όπως το HTML, PDF, ή Microsoft Word, Excel και τα αρχεία PowerPoint.
- Δημιουργία ευρετηρίων των πληροφοριών τομέων, όπως ο τίτλος, H1, πληροφορίες ετικετών HTML
- Δεικτοδότηση των πληροφοριών θέσης για μια λέξη, ή ένα επίπεδο φραγμών (π.χ. ένα παράθυρο των όρων μέσα σε μια απόσταση).
- Υποστηρίζει τις διάφορες κωδικοποιήσεις των εγγράφων (UTF), για να διευκολύνει την πολύγλωσση ανάκτηση.
- Υποστηρίζει την προσκόμιση των αρχείων στο ευρετήριο από το HTTP, που επιτρέπει στο ενδοδίκτυο την εύκολη αναζήτηση.
- Συμπιεσμένες δομές δεδομένων δίσκων ευρετηρίων.
- Εναλλακτική γρηγορότερη μονή δημιουργία ευρετηρίου.
- Διάφορες προερχόμενος τεχνικές που υποστηρίζονται, συμπεριλαμβανομένης τα ομόριζα για τις ευρωπαϊκές γλώσσες.

Ανάκτηση

- Παρέχει τα πρότυπα ερωτήσεων, καθώς επίσης και την επέκταση ερώτησης
- Μπορεί να εφαρμοστεί στις αλληλεπιδραστικές εφαρμογές.
- Παρέχει πολλά πρότυπα τυποποιημένων σταθμισμένων έγγραφων,
- Προηγμένη γλώσσα διατύπωσης ερωτήσεων που υποστηρίζει τους Boolean χειριστές, +/- χειριστές, αναζήτηση φράσης και εγγύτητας.
- Παρέχει έναν αριθμό από παραμέτρους ελεύθερων ορών DFR σταθμίζοντας τα πρότυπα για την αυτόματη επέκταση ερώτησης, εκτός από την επέκταση ερώτησης Rocchio.
- Εύκαμπτη επεξεργασία των όρων, όπως η απαλοιφή των κοινών λέξεων και των ομόριζων.

Πειραματισμός

- Χειρίζεται όλες τις διαθέσιμες σήμερα συλλογές δοκιμής TREC.
- Εύκολα αλφάβητο για να αξιολογήσει πολλές τοποθετήσεις παραμέτρων.
- Ενσωματωμένα εργαλεία αξιολόγησης για τη χρήση με TREC για συγκεκριμένο σκοπό και τα αποτελέσματα ανάκτησης αναζήτησης γνωστών δεδομένων, για να παραγάγουν τα διάφορα μέτρα ακρίβειας και ανάκλησης.

(http://ir.dcs.gla.ac.uk/terrier)

1.11.13 Σύγκριση

Τα παραπάνω προϊόντα μπορούν να ομαδοποιηθούν σε δύο μεγάλες κατηγορίες:

- Βιβλιοθήκες Ανάκτησης Πληροφοριών (IR libraries)
- Εφαρμογές σύνταξης ευρετηρίων και έρευνας (Indexing and searching applications).

Η πρώτη ομάδα περιλαμβάνει βιβλιοθήκες πλήρους-κειμένου δεικτοδότησης και αναζήτησης, και είναι αυτές που είναι παρόμοιες με το Lucene (το σύστημα που θα χρησιμοποιηθεί για την περάτωση της εργασίας). Τα προϊόντα αυτά επιτρέπουν την ενσωμάτωση τους σε εφαρμογές.

Η δεύτερη ομάδα αποτελείται από λογισμικά δεικτοδότησης και αναζήτησης, τα οποία είναι έτοιμα για χρήση. Αυτού του είδους τα λογισμικά έχουν σχεδιαστεί,

ουσιαστικά, για να δεικτοδοτούν και να αναζητούν ένα συγκεκριμένο τύπο δεδομένων, όπως είναι οι ιστοσελίδες, και είναι λιγότερο εύκαμπτα από αυτά της πρώτης ομάδας. Εντούτοις, μερικά από αυτά τα προϊόντα εκθέτουν επίσης το χαμηλό-επίπεδό API (lower-level API), έτσι μερικές φορές μπορούν να χρησιμοποιηθούν ως βιβλιοθήκες IR.

Η ταξινόμηση των προαναφερόμενων λογισμικό σ' αυτές τις δυο κατηγορίες παρουσιάζεται στο παρακάτω πίνακα (Πίνακας 1.2).

Πίνακας 1.2 Κατηγοριοποίηση των λογισμικών στην κατηγορία που ανήκει το καθένα

Λογισμικά	**Βιβλιοθήκες IR**	**Εφαρμογές δεικτοδότησης και αναζήτησης**
Egothor	√	
Ht:\\Dig		√
Lemur	√	
Sphinx		√
MG	(√)	
DataparkSearch		√
Wumpus		√
Zettair		√
Xapian	√	
Zebra		√
LingPipe	√	
Terrier		√

Επίσης παρατίθεται ένας συγκριτικός συνοπτικός πίνακας που αναφέρει τα κυριότερα χαρακτηριστικά για κάθε σύστημα. (Πίνακας 1.3)

Πίνακας 1.3 Συκγριτικός πίνακας όλων των λογισμικών.

Χαρακτηριστικά Λογισμικό	Γλώσσα Συγγραφής	Σύστηματα	Τύπος εγγράφων αναζήτησης	Τύπος εγράφων δεικτοδότησης	Boolean	Stop word	Stemmer	Γλώσσα που υποστηρίζει	Υποστήριξη SQL
Egothor	Java		HTML, κοινά αρχεια, εμπλουτισμένο Xdelta	—	Ναι			Πολυγλωσσικό	
Ht:\\Dig	C++	Unix	HTML, κοινά αρχεία	HTML, κοινά αρχεία	Ναι	Ναι	Ναι		
Lemur	C++	Windows XP, Linux, MAC OS/X		—		Ναι	Ναι	Αγγλικά, κινέζικα, αραβικά	
Sphinx	C++	Linux, Windows, FreeBSD, NetBSD, Solaris, MAC OS/X		MySQL, PostgreSQL, XML		Ναι	Ναι	Αγγλικά, ρώσικα	Ναι
MG				—			Ναι		Ναι
DataparkSearch	C		HTML, XML, MP3, GIF	HTML	Ναι	Ναι	Ναι	Πολυγλωσσικό	
Wumpus									
Zettair	C	MAC, Linux, FreeBSD, Solaris, Win32, Cygwin	HTML	HTML	Ναι				
Xapian	C++	Linux, FreeBSD, Mac, Solaris, NetBSD, Windows	HTML, PHP, κοινά αρχεία	—	Ναι		Ναι	Πολυγλωσσικό	
Zebra	C	Windows, Linux, Unix	XML, GRS-1, SGML, txt	XML, txt	Ναι				
LingPipe	Java			—		Ναι	Ναι	Πολυγλωσσικό	
Terrier	Java	Windows, MAC, Linux, Unix	HTML, κοινά αρχεία	HTML, κοινά αρχεία	Ναι	Ναι		Ευρωπαικές	

Κοινά αρχεία: PDF, PS, DOC, XLS

Κεφάλαιο 2

Υλικά & Μέθοδοι

Στο πρώτο κεφάλαιο της εργασίας αυτής έγινε μια μικρή εισαγωγή για τον τομέα της Ανάκτησης Πληροφορίας γενικά. Στόχος, όμως, αυτής της εργασίας είναι να χρησιμοποιηθεί αυτός ο τομέας στην Βιοπληροφορική, γι' αυτό το λόγο θα γίνει μια μικρή εισαγωγή στον τομέα της Βιοπληροφορικής και στο πως αυτή συνδέεται με την Ανάκτηση Πληροφορίας αρχικά, και έπειτα θα προχωρήσουμε με την ανάλυση των εργαλείων που θα χρησιμοποιηθούν για τη κατασκευή της εργασίας.

2.1 Εισαγωγή Στην Βιοπληροφορική

Η γρήγορη πρόοδος στην έρευνα της γενετικής και της βιοχημείας έχει ως άμεσο αποτέλεσμα και την μεγάλη αύξηση του όγκου των δεδομένων στις βάσεις βιολογικών δεδομένων. Αυτή η κατάσταση με τη σειρά της οδήγησε σε μεγάλη αύξηση του ενδιαφέροντος για την Βιοπληροφορική και την Υπολογιστική Βιολογία με σκοπό την ανάλυση και κατανόηση των ογκώδεων δεδομένων.

Κατά συνέπεια η Βιοπληροφορική μπορεί να οριστεί ως ο τομέας που παρέχει τα απαραίτητα υπολογιστικά εργαλεία, τις βάσεις δεδομένων, και τις μεθόδους για την υποστήριξη της γωνιδιωματικής (genomic) και μετά-γωνιδιωματικής (postgenomic) έρευνας. Περιλαμβάνει τη μελέτη της δομής και της λειτουργίας του DNA και των πρωτεϊνικών ακολουθιών, την στοίχιση των ακολουθιών του DNA και των πρωτεϊνών με στόχο την σύγκριση αυτών, και την δημιουργία και απεικόνιση τρισδιάστατων μοντέλων για τις πρωτεϊνικές δομές. Με δύο λόγια η Βιοπληροφορική είναι ο κλάδος της βιολογικής επιστήμης που εξετάζει τη μελέτη της εφαρμογής της τεχνολογίας πληροφοριών στον τομέα της μοριακής βιολογίας.

Ο ορισμός που δίνεται για την Βιοπληροφορική από το National Center for Biotechnology Information (NCBI 2001) είναι ο εξής: «Η Βιοπληροφορική είναι ο τομέας της επιστήμης στον οποίο η βιολογία, η πληροφορική, και η τεχνολογία πληροφοριών συγχωνεύονται σε μια ενιαία τάξη. Υπάρχουν τρεις σημαντικές υπό-τάξεις μέσα στη βιοπληροφορική: η ανάπτυξη των νέων αλγορίθμων και των στατιστικών για την αξιολόγηση της σχέσης μεταξύ των μελών των μεγάλων

συνόλων δεδομένων· η ανάλυση και η ερμηνεία των διάφορων τύπων δεδομένων συμπεριλαμβανομένων των ακολουθιών νουκλεοτιδίων και αμινοξέων, των πρωτεϊνικών περιοχών, και των πρωτεϊνικών δομών· και η ανάπτυξη και εφαρμογή των εργαλείων που επιτρέπουν την αποδοτική πρόσβαση και τη διαχείριση των διαφορετικών τύπων πληροφοριών». (http://www.geocities.com/bioinformaticsweb/definition.html)

Ένας πολλά υποσχόμενος τρόπος για την ανάπτυξη της τεχνολογίας πληροφορικής για τα βιολογικά συστήματα, βρίσκεται στην εφαρμογή της σύγχρονης *ανάκτησης πληροφοριών* (IR) και *των διοικητικών* αλγορίθμων *γνώσης* (Knowledge Management - KM) στις βάσεις δεδομένων που περιέχουν βιοϊατρικές δημοσιεύσεις και δεδομένα. Οι σύγχρονες πηγές πληροφοριών μπορούν να θεωρηθούν ως δίκτυα των εγγράφων. Το πρωταρχικό παράδειγμα ενός δικτύου εγγράφων είναι ο Παγκόσμιος Ιστός, αλλά υπάρχουν και άλλοι τύποι τέτοιων δικτύων όπως είναι οι βιβλιογραφικές βάσεις δεδομένων που περιέχουν τις επιστημονικές δημοσιεύσεις (π.χ. MEDLINE: http://www.nlm.nih.gov) και οι βάσεις δεδομένων των συνολικών δεδομένων που χρησιμοποιούνται στις επιστημονικές προσπάθειες (π.χ. GenBank: http://www.ncbi.nlm.nih.gov/Genbank/ και PROSITE: http://www.expasy.org/prosite/). Κάθε μια από αυτές τις βάσεις δεδομένων κατέχει διάφορες ευδιάκριτες σχέσεις μεταξύ των εγγράφων και μεταξύ των εγγράφων και των σημασιολογικών ετικετών ή των δεικτών που ταξινομούν τα έγγραφα. Παραδείγματος χάριν, τα έγγραφα στο Διαδίκτυο συσχετίζονται μέσω ενός δικτύου συνδέσμων υπερκειμένου, ενώ τα έγγραφα στις βιβλιογραφικές βάσεις δεδομένων συνδέονται με τα δίκτυα παραπομπής.(Newman, 2000)

Αυτός θα αποτελεί και τον τρόπο που θα αναπτυχθεί η εφαρμογή της παρούσας εργασίας. Για την πραγματοποίηση θα χρειαστούμε κάποια εργαλεία, και το πρώτο θα είναι ένα σύστημα ανάκτησης πληροφοριών, συγκεκριμένα το Lucene.

2.2 Το Lucene

Ένας από τους παράγοντες κλειδιά που βρίσκονται πίσω από την δημοτικότητα και την επιτυχία του Lucene είναι η απλότητά του. Το Lucene είναι μια ελεύθερη/ανοικτή κώδικα βιβλιοθήκη ανάκτησης πληροφοριών, που αρχικά δημιουργήθηκε σε Java από τον Doug Cutting. Υποστηρίζεται από το ίδρυμα

λογισμικού Apache (Apache Software Foundation) και εκδίδεται με άδεια λογισμικού Apache (Apache Software License). Η προσεκτική έκθεση του API ταξινόμησης και αναζήτησης είναι ένα σημάδι του καλού σχεδιασμού του λογισμικού. Συνεπώς, δεν χρειάζεται κάποιος να γνωρίζει σε βάθος πώς οι πληροφορίες του Lucene ταξινομούνται και ανακτώνται για να αρχίσει να το χρησιμοποιεί. Επιπλέον, τα ακριβή API του Lucene απαιτούν να γνωρίζει ο χρήστης πώς να χρησιμοποιεί μόνο κάποιες από τις κλάσεις.

2.2.1 Η εξέλιξη του οργανισμού πληροφορίας και της προσπέλασης

Για να γίνει κατανοητή η διακριτή πολυπλοκότητα στον κόσμο, οι άνθρωποι επινόησαν διάφορους τύπους ιεραρχικών σχημάτων κατηγοριοποίησης όπως οι κατηγοριοποιήσεις, ταξινομήσεις, ταξινόμηση σύμφωνα με το γένος, τα είδη, και άλλα. Το δεκαδικό σύστημα κατηγοριοποίησης αντικειμένων του Dewey σε μια συλλογή βιβλιοθηκών είναι ένα κλασσικό παράδειγμα ιεραρχικού σχήματος κατηγοριοποίησης. Η έκρηξη του Διαδικτύου και των θησαυρών γνώσεως ηλεκτρονικών δεδομένων παρουσιάζει μεγάλο σύνολο πληροφοριών. Κάποιες εταιρίες, όπως η Yahoo!, δημιούργησαν την οργάνωση και ταξινόμηση των διαθέσιμων ηλεκτρονικά (online) δεδομένων των επιχειρήσεών τους. Με τον καιρό, ωστόσο, το σύνολο των διαθέσιμων δεδομένων έχει γίνει τόσο τεράστιο ώστε χρειάζονται εναλλασσόμενοι και πιο δυναμικοί τρόποι για την εύρεση πληροφοριών. Αν και μπορούν να ταξινομηθούν τα δεδομένα, ψαρεύονται από εκατοντάδες ή χιλιάδες κατηγορίες και υποκατηγορίες δεδομένων δεν είναι μακρύτερα μια αποδοτική μέθοδο για την εύρεση πληροφοριών.

Η ανάγκη για γρήγορη εύρεση πληροφοριών σε μια ‘θάλασσα’ δεδομένων δεν περιορίζεται στην κυριαρχία του Διαδικτύου – οι υπολογιστές μπορούν να αποθηκεύσουν όλο και περισσότερα δεδομένα.

Με τις άφθονες πληροφορίες, και με το χρόνο ζωής να αποτελεί ένα από τα πιο πολύτιμα πλεονεκτήματα για τους περισσότερους ανθρώπους, απαιτείται η ικανότητα για δήλωση εύκαμπτων, ελεύθερων-μορφών, συγκεκριμένου σκοπού ερωτήσεων.

Για να γίνει κατανοητό τι ρόλο παίζει το Lucene στην αναζήτηση, θα ξεκινήσουμε με τα βασικά, τι είναι το Lucene και πως μπορεί να βοηθήσει στην αναζήτηση του χρήστη.

2.2.2 Γνωρίζοντας το Lucene

Διάφοροι άνθρωποι αντιμετωπίζουν το πρόβλημα της υπερφόρτωση πληροφοριών χρησιμοποιώντας διαφορετικές προσεγγίσεις. Κάποιοι έχουν δουλέψει σε νέα περιβάλλοντα αλληλεπίδρασης χρήστη, κάποιοι με έξυπνους πράκτορες, και άλλοι με αναπτυσσόμενα σύνθετα εργαλεία αναζήτησης όπως το Lucene.

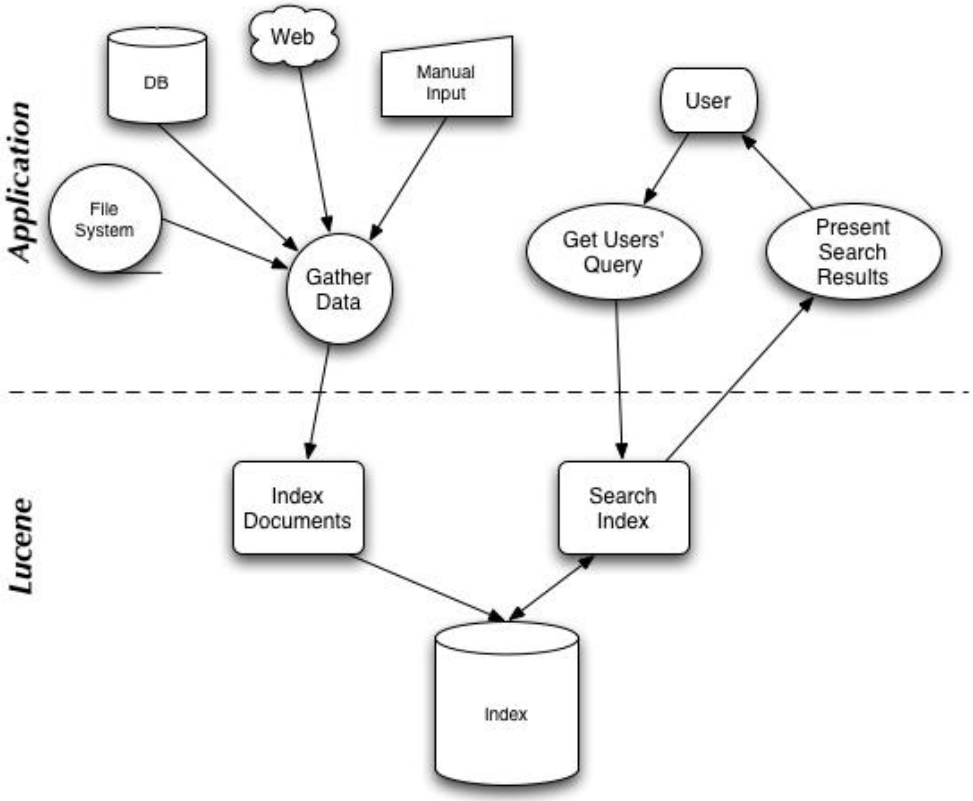
Το Lucene είναι μια υψηλής απόδοσης, κλιμακωτή βιβλιοθήκη Ανάκτησης Πληροφορίας (IR). Επιτρέπει στους χρήστες του να προσθέτουν δυνατότητες δημιουργίας ευρετηρίων και αναζήτησης στις εφαρμογές τους. Το Lucene πρόκειται για ένα πλήρως αναπτυγμένο, ελεύθερο, ανοικτού κώδικα έργο που υλοποιείται σε Java· είναι μέλος της δημοφιλούς οικογένειας εφαρμογών Apache Jakarta, έχει άδεια από το Apache Software License. Το Lucene αποτελεί επί του παρόντος, και θα αποτελεί ακόμη για μερικά χρόνια, τη πιο δημοφιλή ελεύθερη βιβλιοθήκη IR γραμμένη σε Java.

Το Lucene παρέχει ένα απλό ισχυρό πυρήνα API που απαιτεί την ελάχιστη κατανόηση της σύνταξης ευρετηρίων και αναζήτησης σε ένα πλήρες κείμενο. Μπορεί κάποιος να μάθει μόνο ένα μέρος των κλάσεών του για να ξεκινήσει μια εφαρμογή εξολοκλήρου με το Lucene. Επειδή το Lucene είναι μια βιβλιοθήκη σε Java δεν κάνει υποθέσεις για το τι δεικτοδοτείτε και τι αναζητάτε, κάτι το οποίο του δίνει ένα πλεονέκτημα ενάντια σε άλλες εφαρμογές αναζήτησης.

Οι άνθρωποι που είναι νέοι στην χρήση του συχνά κάνουν το λάθος και πιστεύουν ότι το Lucene είναι μια εφαρμογή διαθέσιμη για χρήση όπως ένα πρόγραμμα για την αναζήτηση αρχείων, ένα ερπετό ιστού (web crawler), ή μια ιστοσελίδα μηχανής αναζήτησης. Αντίθετα το Lucene είναι ένα λογισμικό βιβλιοθήκης, μια σειρά εργαλείων ίσως, όχι μια πλήρως χαρακτηριζόμενη εφαρμογή αναζήτησης. Αυτό που το απασχολεί είναι η σύνταξη ευρετηρίων και αναζήτηση κειμένων, και τα κάνει πολύ καλά και τα δύο. Μπορεί κανείς να σκεφτεί το Lucene ως ένα επίπεδο όπου οι εφαρμογές κάθονται στην κορυφή, όπως απεικονίζεται στην Εικόνα 2.1.

Ένα πλήθος από πλήρως χαρακτηρισμένες εφαρμογές αναζήτησης έχουν χτιστεί στην κορυφή του Lucene. Αν κάποιος ενδιαφέρεται για κάτι προ-χτισμένο ή για ένα πλαίσιο εργασίας για σύρσιμο (crawling), χειρισμού εγγράφων, και αναζήτησης, ας συμβουλευτεί την σελίδα Lucene Wiki

(http://wiki.apache.org/jakarta-lucene/PoweredBy) όπου παρέχονται πολλές επιλογές, τα Zilverline, SearchBlox, Nutch, LARM, και jSearch, είναι κάποιες από αυτές.



Εικόνα 2.1. Μια τυπική εφαρμογή ενοποίησης με το Lucene[i]

Το Lucene επιτρέπει στον χρήστη να προσθέσει δυνατότητες δεικτοδότηση και αναζήτησης στις εφαρμογές του. Μπορεί να δεικτοδοτήσει και να κάνει ανιχνεύσιμο κάθε δεδομένο το οποίο μπορεί να μετατραπεί σε μορφή κειμένου. Όπως φαίνεται και στην Εικόνα 2.1 το Lucene δεν ενδιαφέρεται για την πηγή των δεδομένων, για τη μορφή τους, ή ακόμη και για την γλώσσα, εφόσον μπορεί να μετατραπεί σε κείμενο. Αυτό σημαίνει ότι μπορεί να χρησιμοποιηθεί για δεικτοδότηση και αναζήτηση δεδομένων που είναι αποθηκευμένα σε αρχεία: ιστοσελίδες απομακρυσμένες από τους εξυπηρετητές δικτύου, έγγραφα αποθηκευμένα σε τοπικά αρχεία του συστήματος, απλά αρχεία κειμένων, έγγραφα τύπου Microsoft Word, HTML ή αρχεία PDF, ή σε οποιαδήποτε άλλη μορφή από όπου μπορούν να εξαχθούν πληροφορίες βασισμένες σε κείμενο. Ομοίως, με την βοήθεια του Lucene μπορεί κανείς να δεικτοδοτήσει δεδομένα αποθηκευμένα σε μια

[i] Πηγή εικόνας από το κεφάλαιο 1 του βιβλίου “Lucene in Action” των Gospodnetic O. και Hatcer E (2005).

βάση δεδομένων, παρέχοντας δυνατότητες αναζήτησης πλήρους κειμένου, που πολλές βάσεις δεν εξασφαλίζουν. Ακόμη, επιτρέπει την δεικτοδότηση και αναζήτηση μηνυμάτων ηλεκτρονικού ταχυδρομείου, αρχείων λίστας σταλθέντων μηνυμάτων, και άλλα.

Το Lucene γράφτηκε από τον Doug Cutting· ήταν αρχικά διαθέσιμο για κατέβασμα από την αρχική του ιστοσελίδα SourceForge. Συμμετείχε στην οικογένεια Jakarta του Apache Software Foundation για την υψηλής ποιότητας προϊόντος ανοιχτού κώδικα Java το Σεπτέμβριο του 2001. Με αυτή την ανακοίνωση και από τότε, η ζήτηση του προϊόντος αυξάνεται συνεχώς, ελκύοντας όλο και περισσότερους χρήστες και ερευνητές. Η έκδοση που θα χρησιμοποιηθεί για την εφαρμογή είναι η 3.2.1, αλλά συνεχώς υπάρχουν καινούργιες εκδόσεις με επιδιορθώσεις. Για την εύρεση των τελευταίων εκδόσεων μπορεί κανείς να επισκεφτεί το δικτυακό τόπο του Lucene, http://lucene.apache.org/java/docs/.

Ο Doug Cutting παραμένει η κύρια δύναμη πίσω από το Lucene, αλλά πιο έξυπνες ιδέες συμμετέχουν στο προϊόν από τότε που το Lucene μετακινήθηκε κάτω από την προστασία του Apache Jakarta. Εκτός από τους επίσημους υπεύθυνους για την ανάπτυξη του προγράμματος, το Lucene έχει μια αρκετά μεγάλη και ενεργό τεχνική κοινότητα χρηστών που συμβάλλει συχνά στις επιδιορθώσεις, και τα νέα χαρακτηριστικά γνωρίσματα.

Στον δικτυακό τόπο του επίσης αναφέρονται ένας αριθμός από μεγάλες, καλά αναγνωρισμένες, πολυεθνικές εταιρίες που χρησιμοποιούν το Lucene. Παρέχει ικανότητες αναζήτησης για το Eclipse IDE, το CD-ROM/DVD της Εγκυκλοπαίδειας Britannica, FedEX, Mayo Clinic, Hewlett-Packard, το περιοδικό New Scientist, Epiphany, MIT’s OpenCourseware και DSpace, και τα λοιπά.

Ένας τρόπος για να υπολογιστεί η επιτυχία των λογισμικών ανοικτού κώδικα είναι να υπολογιστεί ο αριθμός των άλλων γλωσσών προγραμματισμού, εκτός από την αρχική τους γλώσσα, που έχουν γραφτεί. Χρησιμοποιώντας αυτή την μετρική, το Lucene είναι μια απόλυτη επιτυχία! Αν και το αρχικό Lucene είναι γραμμένο σε Java, επίσης μπορούν να χρησιμοποιηθούν οι Perl, Python, C++, και .NET, και Ruby. Αυτά είναι υπέροχα νέα για τους κατασκευαστές που χρειάζονται να προσπελάσουν τα ευρετήρια του Lucene από εφαρμογές που γράφονται σε διαφορετικές γλώσσες.

2.2.3 Δεικτοδότηση και Αναζήτηση

Στην καρδιά όλων των μηχανών αναζήτησης είναι η αρχή της δεικτοδότησης: επεξεργασία των αρχικών δεδομένων σε μια υψηλού απόδοσης διαδικασία αναζήτησης διαναφοράς για να διευκολύνει την γρήγορη αναζήτηση.

Ας υποθέσουμε ότι κάποιος χρειάζεται να ψάξει σε ένα μεγάλο αριθμό αρχείων, και απαιτεί να βρίσκει αρχεία τα οποία περιέχουν μια σταθερή λέξη ή μια φράση. Πως μπορεί να αναλάβει το γράψιμο ενός προγράμματος για αυτό; Μια απλοϊκή προσέγγιση είναι να σαρωθεί διαδοχικά κάθε αρχείο για τη δοσμένη λέξη ή φράση. Αυτή η προσέγγιση έχει ένα αριθμό από ελαττώματα, το πιο φανερό από αυτά είναι ότι δεν μπορεί να εφαρμοστεί σε μεγαλύτερο σύνολο αρχείων ή γεγονότα. Εδώ είναι όπου η *δεικτοδότηση* αναλαμβάνει δράση: Για την αναζήτηση μεγάλου ποσού κειμένου γρήγορα, πρέπει πρώτα ο χρήστης να δεικτοδοτήσει αυτό το κείμενο και να το μετασχηματιστεί σε μια μορφή που θα επιτρέπει στο χρήστη να αναζητήσει το κείμενο γρήγορα, εξαλείφοντας τις αργές διαδοχικές διαδικασίες σάρωσης. Αυτή η διαδικασία μετατροπής καλείται δεικτοδότηση, και η έξοδος της καλείται ευρετήριο.

Μπορεί κανείς να φανταστεί ένα ευρετήριο ως μια δομή δεδομένων που επιτρέπει την γρήγορη τυχαία προσπέλαση των λέξεων που αποθηκεύονται μέσα σ' αυτό. Η σκέψη πίσω από αυτό βρίσκεται στα ευρετήρια που υπάρχουν στο τέλος των βιβλίων, που επιτρέπουν στον αναγνώστη τον γρήγορο εντοπισμό σελίδων που συζητούν διάφορα θέματα. Στην περίπτωση του Lucene, ένα ευρετήριο είναι μια ειδικά σχεδιασμένη δομή δεδομένων, που τυπικά αποθηκεύει τα αρχεία του συστήματος ως ένα σύνολο από δεικτοδοτούμενα αρχεία. Η δεικτοδότηση του Lucene, στην ουσία είναι, ένα εργαλείο που επιτρέπει την γρήγορη εύρεση μιας λέξης.

Αναζήτηση είναι η διαδικασία αναζήτησης λέξεων σε ένα ευρετήριο για την εύρεση εγγράφων όπου αυτές εμφανίζονται. Η ιδιότητα μιας αναζήτησης είναι τυπικά περιγραφόμενη χρησιμοποιώντας τις μετρικές της ακρίβειας και της ανάκλησης. Η ανάκληση μετράει πόσο καλά το σύστημα αναζήτησης βρίσκει σχετικά έγγραφα, ενώ η ακρίβεια μετράει πόσο καλά το σύστημα φιλτράρει τα άσχετα έγγραφα. Όμως, ο χρήστης πρέπει να μελετήσει ένα αριθμό από άλλους παράγοντες που έχουν να κάνουν με την αναζήτηση, όπως η ταχύτητα και η ικανότητα της γρήγορης αναζήτησης μεγάλου ποσού κειμένου. Η υποστήριξη για απλές και πολλαπλών όρων ερωτήσεις, ερωτήσεις φράσεων, wildcards, σπουδαιότερου αποτελέσματος,

αποτελούν επίσης σημαντικά χαρακτηριστικά για μια αναζήτηση, καθώς είναι συντάξεις φιλικές προς το χρήστη για την εισαγωγή αυτών των ερωτήσεων. Η εισαγωγή των ερωτήσεων συνδέεται με τον τρόπο που θα επεξεργαστεί το Lucene την ερώτηση, δηλαδή ο τρόπος επεξεργασίας που θα επιλέξει ο κατασκευαστής έχει άμεσο αντίκτυπο στον τρόπο δήλωσης των ερωτήσεων από τον χρήστη. Το Lucene υποστηρίζει διάφορους τρόπους για την επεξεργασία της ερώτησης του χρήστη, άλλοι πιο απλοί και εύχρηστοι και άλλοι πιο πολύπλοκοι. Στην συνέχεια περιγράφονται εν συντομία αυτοί οι τύποι.

- TermQuery: Πρόκειται για ένα στοιχειώδη τρόπο για την αναζήτηση, η οποία χρησιμοποιεί έναν ειδικό όρο. Ένας όρος είναι το μικρότερο δεικτοδοτούμενο κομμάτι, που αποτελείται από ένα ζεύγος ονόματος πεδίο και μια τιμή κειμένου. Για παράδειγμα, έστω ότι από μια συλλογή κειμένων θέλουμε τα κείμενα που περιέχουν τη λέξη 'java' στο πεδίο 'contents' (περιεχόμενα), τότε ο κώδικας θα ήταν ο εξής:

```
Term t = new Term("contents", "java");
Query query = new TermQuery(t);
```

- RangeQuery: Αυτή η μέθοδος εκτελεί μια αναζήτηση από έναν αρχικό όρο ως ένα τελικό. Ο αρχικός και ο τελικός όρος πρέπει να συμπεριλαμβάνονται ή να αποκλείονται και οι δυο. Ο παρακάτω κώδικας κάνει μια αναζήτηση για την εύρεσης κειμένων τα οποία να έχουν δημοσιευτεί μεταξύ του Μαΐου του 1988 και του Οκτωβρίου του 1988.

```
Term begin = new Term("pubmonth","198805");
Term end = new Term("pubmonth","198810");
…
RangeQuery query = new RangeQuery(begin, end, true);
```

Το τρίτο όρισμα της `RangeQuery,` δηλαδή το `true`, δηλώνει ότι και οι δυο όροι θα πρέπει να συμπεριλαμβάνονται στην αναζήτηση.

- PrefixQuery: Η αναζήτηση με αυτή την μέθοδο ψάχνει για έγγραφα τα οποία περιέχουν όρους που αρχίζουν με μια συγκεκριμένη αλληλουχία. Στο παρακάτω παράδειγμα ψάχνουμε για προγραμματιστικά βιβλία, συμπεριλαμβάνοντας και υποκατηγορίες:

```
Term term = new Term
        ("category","/technology/computers/programming");
PrefixQuery query = new PrefixQuery(term);
```

- BooleanQuery: Οι διάφοροι τρόποι που τέθηκαν παραπάνω μπορούν να συνδυαστούν με την χρήση αυτής της μεθόδου. Η BooleanQuery περιέχει τις προτάσεις Boolean. Μια πρόταση είναι μια υπό-ερώτηση η οποία μπορεί να είναι προαιρετική, απαιτούμενη, ή απαγορευμένη. Αυτά τα χαρακτηριστικά επιτρέπουν την χρήση των λογικών AND, OR, και NOT, ή ακόμη και συνδυασμούς αυτών.
- PhraseQuery: Αυτός ο τρόπος χρησιμοποιεί τις πληροφορίες που έχει ένα ευρετήριο (το οποίο περιέχει πληροφορίες για την θέση των όρων) για να εντοπίσει έγγραφα τα οποία οι όροι έχουν μια συγκεκριμένη απόσταση.
- WildcardQuery: Αυτή η μέθοδος μπορεί να δεχτεί και να αναζητήσει όρους με υπολειπόμενους χαρακτήρες. Δυο τυποποιημένη χαρακτήρες που χρησιμοποιεί είναι οι εξής: * για μηδέν ή περισσότερους υπολειπόμενους χαρακτήρες, και ? για μηδέν ή έναν ακόμη χαρακτήρα. Ένα παράδειγμα χρήσης του είναι: `Query query = new WildcardQuery(new Term(“contents”, “?ild*”));`
- FuzzyQuery: Αυτή η μέθοδος ταιριάζει όρους που είναι παρόμοιοι με ένα συγκεκριμένο όρο.
- QueryParser: Όλοι οι προηγούμενοι τρόποι είναι γραμμένοι σε Java, αλλά αυτή η τεχνική χρησιμοποιεί ερωτήσεις σε κατανοήσιμη μορφή από τον άνθρωπο. Κατασκευάζει μια από τις προηγούμενες υποκατηγορίες ερωτήσεων.

Η βιβλιοθήκη ισχυρού λογισμικού Lucene προσφέρει ένα αριθμό από μελλοντικά χαρακτηριστικά. Κάποια από τα χαρακτηριστικά του Lucene απεικονίζονται στον Πίνακα 2.1.

Πίνακας 2.1. Χαρακτηριστικά του Lucene

Δεικτοδότηση επέκτασης και υψηλής απόδοσης	**Ισχυροί, ακριβείς και αποτελεσματικοί αλγόριθμοι αναζήτησης**	**Λύση cross-πλατφόρμας**
Πάνω από 20MB/min σε Pentium M 1.5GHz	Ταξινομημένη αναζήτηση, το καλύτερο αποτέλεσμα επιστρέφεται πρώτο	Διατίθεται ως Open Source λογισμικό υπό την Apache License που σας επιτρέπει να χρησιμοποιήσετε το Lucene τόσο σε εμπορικά όσο και σε Open Source προγράμματα
Μικρές απαιτήσεις RAM	Πολλά είδη ισχυρών ερωτημάτων: οι ερωτήσεις φράσεων, ερωτήσεις μπαλαντέρ, τα ερωτήματα εγγύτητας και άλλα	100% καθαρή Java
Σταδιακή δεικτοδότηση	Αναζήτηση τοποθετημένη σε πεδία (π.χ. τίτλος, συντάκτης)	Υλοποιήσεις σε άλλες γλώσσες προγραμματισμού που είναι συμβατές με ευρετήριο
Ευρετήριο μεγέθους περίπου 20-30% του μεγέθους του κειμένου στο ευρετήριο	Αναζήτηση ακτίνας ημερομηνίας	
	Διαλογή από κάθε τομέα	
	Πολλαπλά ευρετήρια αναζήτησης με ενοποιημένα αποτελέσματα	
	Επιτρέπει την ταυτόχρονη ενημέρωση και αναζήτηση	

2.2.4. Εγκατάσταση του Lucene

Το πρώτο πράγμα που θα χρειαστεί να κάνει κάποιος για την εγκατάσταση είναι να επισκεφτεί τον ιστοχώρο του Lucene και να κατεβάσει από εκεί την τελευταία έκδοση του Lucene. Η έκδοση που θα χρησιμοποιηθεί στην εφαρμογή είναι η 2.3.1. Υπάρχουν δύο μορφές που είναι διαθέσιμες για να κατεβάσει κανείς: είτε αρχείο τύπου .zip είτε αρχείο τύπου .tar.gz. Ο καθένας επιλέγει αυτή που είναι πιο κατάλληλη για το περιβάλλον του συστήματος του.

Στην συνέχεια εξάγεται το δυαδικό αρχείο στην διεύθυνση επιλογής στο σύστημα. Το αρχείο περιλαμβάνει μια διεύθυνση υψηλού επιπέδου που ονομάζεται lucene-2.3.1, έτσι είναι ασφαλής να γίνει η εξαγωγή στον δίσκο C:\ στα Windows ή στην αρχική διεύθυνση του καθενός στα UNIX. Στα Windows, εάν κάποιος διαθέτει

το WinZip, το χρησιμοποιεί για να ανοίξει το αρχείο .zip και να εξάγει το περιεχόμενο του στον c:\. Εάν διαθέτει UNIX ή χρησιμοποιεί cygwin στα Windows, κάνει unzip στο φάκελο και στην συνέχεια untar (tar zxvf lucene-2.3.1.tar.gz) το αρχείο στην αρχική του διεύθυνση.

Μετά από την διαδικασία αυτή, μπορεί κανείς να βρει το lucene-2.3.1.jar. Αυτό είναι το μοναδικό αρχείο που απαιτείται για να εισαχθεί το Lucene στις εφαρμογές κάποιου. Υπάρχουν διάφοροι τρόποι όπου μπορεί κάποιος να χρησιμοποιήσει για να ενσωματώσει το αρχείο Jar του Lucene στις εφαρμογές εξαρτώντας από το περιβάλλον. Από τους κατασκευαστές του Lucene προτείνετε να χρησιμοποιηθεί το Ant για το χτίσιμο του κώδικα των εφαρμογών. Σε αυτή την περίπτωση πρέπει ο χρήστης να σιγουρέψει ότι ο κώδικας του έχει μεταγλωττιστεί έναντι του Jar του Lucene χρησιμοποιώντας την επιλογή μονοπατιού της τάξης από τη ομάδα <javac>.

Τέλος, σε κάθε διανομή εφαρμογής θα πρέπει να συμπεριληφθεί το αρχείο JAR του Lucene κατάλληλα. Για παράδειγμα, σε εφαρμογές της γραμμής Εντολών θα πρέπει να επιβεβαιωθεί ότι το Lucene βρίσκεται στο μονοπάτι κλάσεων (classpath) κατά την προώθηση του JVM.

Η εγκατάσταση του Lucene είναι εύκολη αν τα βήματα ακολουθηθούν σωστά. Λίγο περισσότερο προσοχή πρέπει να δείξει κανείς όσο αναφορά στο τελευταίο σκέλος, δηλαδή την σωστή τοποθέτηση του αρχείου jar ανάλογα με την εφαρμογή. (Gospodnetic & Hatcher, 2005)

2.3 Το LingPipe

Ο σκοπός όμως της εργασίας είναι αναπτυχθεί μια ακόμη εφαρμογή, που στόχος της θα είναι ο χαρακτηρισμός των οντοτήτων που υπάρχουν σε βιοϊατρικά κείμενα. Για την περάτωση αυτής της εργασίας θα χρησιμοποιηθεί ως βάση ένα άλλο σύστημα, το LingPipe.Το LingPipe είναι μια συλλογή από βιβλιοθήκες της Java για την γλωσσική ανάλυση της ανθρώπινης γλώσσας. Τα εργαλεία εξαγωγής πληροφορίας και εξόρυξης δεδομένων του LingPipe είναι:

- αναφορά των ιχνών των οντοτήτων (π.χ. άνθρωποι ή πρωτεΐνες)
- αναφορά συνδέσμων οντοτήτων σε εισαγωγές των βάσεων δεδομένων
- μη καλυπτόμενες σχέσεις μεταξύ των οντοτήτων και των ενεργειών (action)

- ταξινόμηση των εδάφιων των κειμένων σύμφωνα με την γλώσσα, το χαρακτήρα, την κωδικοποίηση, το είδος (ύφος), το θέμα ή την γνώμη
- διόρθωση της ορθογραφίας σε σχέση με μια συλλογή κειμένων
- ομαδοποίηση των εγγράφων ανάλογα με τα συνεπαγόμενα θέματα και την ανακάλυψη της εκφραστικής τάσης, και
- ορισμός ετικετών μέρους του λόγου και κομματιών φράσεων.

2.3.1. Η δομή του LingPipe

Η αρχιτεκτονική του LingPipe σχεδιάστηκε να είναι αποτελεσματική, κλιμακωτή, επαναχρησιμοποιήσιμη, και γερή. Στα πιο σημαντικά χαρακτηριστικά του περιλαμβάνονται και τα εξής:

- το Java API με το κώδικα πηγής και ομάδες ελέγχου (τεστ),
- πολυγλωσσικά, πολλών πεδίων και πολλών ειδών μοντέλα,
- εκπαίδευση με νέα δεδομένα από νέες εργασίες,
- οι ν-καλύτεροι έξοδοι με στατιστική εκτίμηση της εμπιστοσύνης,
- συνεχόμενη εκπαίδευση (on - line) (learn-a-little, tag-a-little),
- μοντέλα ασφαλών νημάτων και αποκωδικοποιητές για συγχρονισμό της παράλληλης–ανάγνωσης και αποκλειστικής–εγγραφής (concurrent-read exclusive-write (CREW)), και
- ευαίσθητη κωδικοποίηση χαρακτήρων Εισόδων /Εξόδων (I/O).

Η τελευταία έκδοση που υπάρχει αυτή την στιγμή και είναι διαθέσιμη και στο Διαδίκτυο στην ιστοσελίδα του οργανισμού είναι η LingPipe 3.5.1. Για την ενημέρωση σχετικά με τις εκδόσεις του προγράμματος θα ήταν χρήσιμο να επισκεφτεί κάποιος τον δικτυακό τόπο http://alias-i.com/lingpipe/.

2.3.2. Εφαρμογές του LingPipe

Το LingPipe παρέχει κάποιες έτοιμες εφαρμογές API για την κατανόηση των δυνατοτήτων του. Η κάθε εφαρμογή έχει τα δεδομένα των δειγμάτων, τα αρχεία τύπου jar που χρειάζονται, καθώς και κάποιο παράδειγμα. Οι εφαρμογές αυτές είναι οι παρακάτω:

Topic Classification: Κατηγοριοποίηση των νέων άρθρων σύμφωνα με το είδος χρησιμοποιώντας μοντέλα γλωσσικών χαρακτήρων.

Named Entity Recognition: Περιγράφει πώς να τρέξει κανείς τα συστήματα αναγνώρισης ονομάτων οντοτήτων με μοντέλα τα πρώτα καλύτερα (first best), ν – καλύτερα (n-best) και εμπιστοσύνης ανά οντότητα (per-entity confidence). Επίσης γίνεται μια περιγραφή του πως κάποιος μπορεί να εκπαιδεύσει και να αξιολογήσει τα συστήματα αναγνώρισης ονομάτων οντοτήτων. Υπάρχουν παραδείγματα με το newswire στα ισπανικά και με το genomics στα αγγλικά.

Clustering: Μια απεικόνιση των ιεραρχικών ομαδοποιήσεων σε ενιαίους-συνδέσμους και πλήρης-συνδέσμους, συμπεριλαμβανομένων μια ποικιλία των τεχνικών συστάδων αξιολόγησης. Υπάρχει ένα παράδειγμα χρήσης ομαδοποίησης για την συν-αναφορά (co reference) των διαγώνιων-εγγράφων. Υπάρχει επίσης ένα εκτενές σεμινάριο στη λανθάνουσα κατανομή Dirichlet (LDA).

Part-of-Speech Tagging: Αναλύει την διαδικασία εκπαίδευσης των ετικετών του μέρους του λόγου (POS) από τα σώματα που χρησιμοποιούν τους κατατμητές και τους χειριστές ετικετών, και ακόμη δίνει μια φόρμα για το πώς θα γίνει η σύνταξη των πρότυπων στο δίσκο και πως θα διαβαστούν, και τέλος πώς θα εκτελεστούν και θα αξιολογηθούν οι ετικέτες με τα μοντέλα πρώτη-καλύτερη (first best), των ν-καλύτερων (n-best) και σύμφωνα με το σκορ εμπιστοσύνης (confidence-scored).

Sentence Detection: Περιγράφει τη διαδικασία εκτέλεσης της ανίχνευσης πρότασης που χρησιμοποιεί τη διεπαφή ομαδικής κατάτμησης, και την διαδικασία της αξιολόγησης της απόδοσης ενός προτύπου πρότασης ενάντια σε ένα εφ' όλης της ύλης σώμα που χρησιμοποιεί τους κατατμητές και τους χειριστές κατάτμησης προτάσεων, και τέλος περιγράφει τη διαδικασία συντονισμού ενός πρότυπου για ένα ιδιαίτερο σώμα.

Spelling Correction: 'Did you mean' – μοντέλο ανίχνευσης προφοράς των μηχανών αναζήτησης. Αυτή η εφαρμογή δηλώνει το πως θα γίνει η εκπαίδευση και ο συντονισμός ενός πρότυπου.

String Comparison: Γίνεται μια επίδειξη της χρήσης της απόστασης και των μέτρων εγγύτητας πέρα από τις σειρές αλφαριθμητικών, συμπεριλαμβανομένου την σταθμισμένη απόσταση διόρθωσης κειμένων, την απόσταση TF/IDF, την απόσταση Jaccard, την απόσταση Jaro-Winkler κλπ.

Interesting Phrase Detection: Εξαγωγή των στατιστικά σημαντικών λέξεων φράσεων σε ένα σώμα και των σχετικά σημαντικών όρων σε ένα σώμα συγκριτικά με κάποιο άλλο.

Character Language Modeling: Η εκπαίδευση και ο συντονισμός των μοντέλων γλωσσικών χαρακτήρων εκτείνεται στο com.aliasi.util.AbstractCommand και χρησιμοποιεί τις com.aliasi.corpus.TextHandler και com.aliasi.corpus.Parser διασυνδέσεις.

MEDLINE Download, Parsing and Indexing: Η χρήση της διεπαφής κατατμητών (αναλυτών λέξεων) της MEDLINE για να εξαγάγει τις παραπομπές MEDLINE όπως αντικείμενα δομημένης Java με μια εφαρμογή στο κείμενο γίνεται η αναζήτηση με την χρήση της μηχανής αναζήτησης Lucene της Apache. Περιέχει επίσης τις πληροφορίες για αυτοματοποιημένο κατέβασμα της MEDLINE, ενημέρωσης και προγράμματα αθροίσματος ελέγχου της μνήμης του Η/Υ.

Database Text Mining: Το πρώτο μέρος 'κατοικεί' σε μια βάση δεδομένων MySQL με τις παραπομπές MEDLINE χρησιμοποιώντας JDBC. Το δεύτερο μέρος προσπελαύνει τη βάση δεδομένων των εγγράφων για να δημιουργήσει τους πίνακες των προτάσεων και των οντοτήτων. Το τρίτο μέρος επιδεικνύει πώς γίνεται η εξόρυξη των δεδομένων κειμένων μέσω των ερωτήσεων στις βάσεις δεδομένων. Γι' αυτή την εφαρμογή απαιτείται η άδεια GNU για την MySQL.

Chinese Word Segmentation: Επιδεικνύει πώς γίνεται η κατάτμηση μιας ροής των κινεζικών χαρακτήρων στις ευδιάκριτες λέξεις. Η επίδειξη χρησιμοποιεί τον τυποποιημένο διορθωτή ορθογραφίας LingPipe με την απόσταση διόρθωσης που συντονίζεται για την κατάτμηση της λέξης. Επιδεικνύει επίσης το πώς γίνεται η εκπαίδευση και η αξιολόγηση χρησιμοποιώντας τα διαθέσιμα δημόσια σώματα κατάρτισης από την πρώτη και δεύτερη διεθνή κινεζική κατάτμηση λέξεων Bakeoffs. Αυτή η εφαρμογή απαιτεί το 'κατέβασμα' των στοιχείων SigHan.

Sentiment Analysis: Χρησιμοποιεί γλωσσικά πρότυπα ταξινόμησης για να κάνει την ανάλυση συναισθήματος πέρα από τις αναθεωρήσεις ταινιών. Ολόκληρες οι αναθεωρήσεις ταινιών ταξινομούνται από την πολικότητα, και οι ενιαίες προτάσεις είναι ταξινομημένες όσον αφορά την υποκειμενικότητα (υποκειμενική/άποψη ή στόχος/γεγονός). Εκτελούνται δια μέσου των μοντέλων μεταγλώττισης και διαβάζοντας τις εκτεταμένες εξόδους που παράγονται από τους εκτιμητές ταξινομητών. Επίσης εξηγεί την ιεραρχική ταξινόμηση που συσσωρεύει τον ταξινομητή πολικότητας πάνω από τον ταξινομητή υποκειμενικότητας για τη βελτιωμένη απόδοση. Συζητά τα διώνυμα διαστήματα εμπιστοσύνης και τον κίνδυνο της a *posteriori* ρύθμισης παραμέτρου. Η εφαρμογή αυτή απαιτεί το 'κατέβασμα' των δεδομένων γνώμεων/συναισθημάτων.

Language Identification: Σε αυτή την εφαρμογή παρουσιάζεται ο γλωσσικός προσδιορισμός ως πρόβλημα ταξινόμησης και παρουσιάζονται τρόποι για το πως θα γίνει η εκπαίδευση και η αξιολόγηση των γλωσσικών προσδιοριστικών, με τα παραδείγματα από το σώμα της Leipzig 15 γλωσσών.

Singular Value Decomposition: Χρησιμοποιεί την μοναδικής αξίας ανάλυσης στις μήτρες συνιστωσών. Εξηγεί πώς θα διανεμηθεί τον άγνωστο αξίας καταλογισμό, τη συστηματοποίηση και θέτει τις παραμέτρους συντονισμού.

Logistic Regression: Πώς να υπολογίσει τα συστηματοποιημένα multinomial λογιστικά πρότυπα οπισθοδρόμησης για την discriminitive ταξινόμηση.

Word Sense Disambiguation: Η αποσαφήνιση έννοιας της λέξης είναι η διαδικασία που του καθορισμού ποια από τις πιθανές έννοιες μιας λέξης προορίζεται από μια ιδιαίτερη περίπτωση της λέξης. Η αποσαφήνιση της έννοιας της λέξης έχει εφαρμογές για την ταξινόμηση, την αναζήτηση, τη συγκέντρωση, και λοιπά.

2.3.3. Εγκατάσταση του LingPipe

Το πρώτο πράγμα το οποίο θα πρέπει να γίνει για την εγκατάσταση του LingPipe είναι να κατεβάσει κανείς το αρχείο του LingPipe από την ιστοσελίδα του οργανισμού στο Διαδίκτυο (http://alias-i.com/lingpipe/web/download.html). Στην συνέχεια θα πρέπει να γίνει εξαγωγή του αρχείου από τη μορφή gzipped tar και να αποθηκευτεί σε μια νέα διεύθυνση στο σύστημα. Η όλη διαδικασία δεν έχει κάποια δυσκολία. Όμως πρέπει κανείς να είναι προσεχτικός σε ότι αφορά τη Java. Το LingPipe για να τρέξει απαιτεί την εγκατάσταση της Java 1.4 ή ενός μεγαλύτερου JVM και το JDK για να μεταγλωττίσει. (http://alias-i.com/lingpipe/)

2.4. Η συλλογή των κειμένων

Εκτός από τα παραπάνω εργαλεία θα χρειαστούν και κάποιες συλλογές κειμένων για την δημιουργία των εφαρμογών. Για το πρώτο κομμάτι της εφαρμογής μας, και το οποίο στηρίζεται στο Lucene, θα χρησιμοποιήσουμε την συλλογή κειμένων Medlars Medical. Αυτή αποτελείται από 1033 αποσπάσματα κειμένων. Μαζί με τα αρχεία κειμένων παρέχεται και ένα αρχείο txt με 30 ερωτήσεις πάνω σε αυτά τα κείμενα για τον έλεγχο του συστήματος.

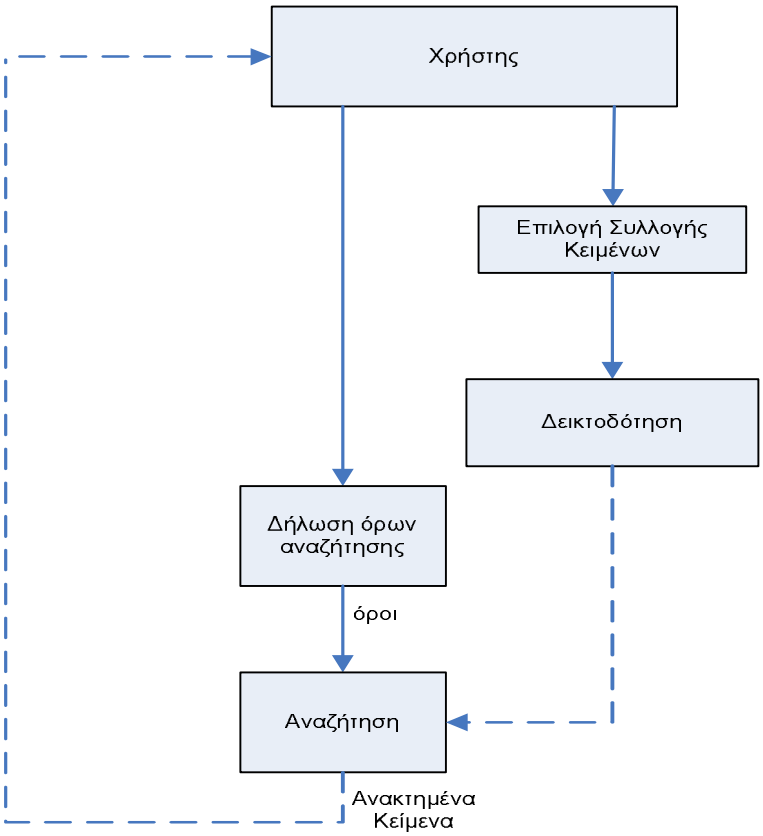
Όσο αναφορά το δεύτερο σκέλος της εφαρμογής μας θα χρησιμοποιήσουμε 91 αποσπάσματα κειμένων από την βάση δεδομένων της Medline του 2008. Η Medline είναι η μεγαλύτερη βιοϊατρική βάση δεδομένων αγγλικής γλώσσας.

Κεφάλαιο 3

Αποτελέσματα

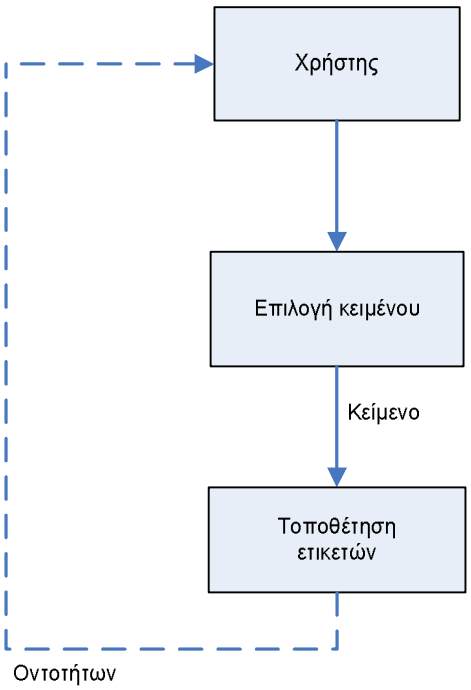
Στόχος της εργασίας αυτής είναι να κατασκευαστούν και να παρουσιαστούν δύο εφαρμογές. Η πρώτη αναφέρεται στην δεικτοδότηση και αναζήτηση βιοϊατρικών κειμένων και η δεύτερη, στην εξαγωγή συγκεκριμένων πληροφοριών από ιατρικά κείμενα. Η δεύτερη εφαρμογή έχει ως στόχο της τον καθορισμό των οντοτήτων που υπάρχουν σε ένα κείμενο.

Η πρώτη είναι η 'Αναζήτηση Βιοϊατρικών Κειμένων'. Στόχος της εφαρμογής αυτής είναι η αναζήτηση και η εμφάνιση κάποιων κειμένων. Για παράδειγμα, έστω ότι υπάρχει κάποιος (ο χρήστης της εφαρμογής), ο οποίος θέλει να βρει κάποια βιοϊατρικά κείμενα τα οποία θα περιέχουν συγκεκριμένες λέξεις, όπως για παράδειγμα τις λέξεις 'human protein'. Σ' αυτή την περίπτωση θα μπορούσε να χρησιμοποιήσει την εφαρμογή αυτή. Η χρήση της είναι αρκετά απλή. Αρχικά, ο χρηστής θα δηλώσει σε ποια συλλογή κειμένων θέλει να γίνει η αναζήτηση για τους όρους αυτούς. Αφού πραγματοποιηθεί αυτό το βήμα, στην συνέχεια η συλλογή θα υποστεί την διαδικασία της 'δεικτοδότησης'. Μετά την διαδικασία αυτή ο χρήστης μπορεί να θέσει τους όρους για την αναζήτηση, δηλαδή να δηλώσει τις λέξεις τις οποίες επιθυμεί να υπάρχουν στα κείμενα. Έπειτα πραγματοποιείται η αναζήτηση, η οποία είναι το τελευταίο στάδιο της εφαρμογής, και παρουσιάζονται τα αποτελέσματα στο χρήστη. Η διαδικασία αυτή απεικονίζεται και στο παρακάτω διάγραμμα (Εικόνα 3.1).



Εικόνα 3.1. Διαγραμμα της εφαρμογής 'Αναζήτηση Βιοϊατρικών Κειμένων'.

Η επόμενη εφαρμογή ονομάζεται 'Εντοπισμός των Οντοτήτων'. Αυτή η εφαρμογή δημιουργήθηκε για να αναγνωρίζει και να εντοπίζει λέξεις με συγκεκριμένο νόημα στην βιολογία, δηλαδή *όρους* τους συγκεκριμένου τομέα επιστήμης, και να τις επιστρέφει στο χρήστη. Η σκέψη ήταν η εξής: Ο χρήστης έχει ένα κείμενο το οποίο είναι αρκετά μεγάλο και δε μπορεί να αναζητήσει σ' αυτό βιολογικούς όρους που ενδεχομένως να τον ενδιαφέρουν. Χρησιμοποιώντας αυτή την εφαρμογή μπορεί να έχει όλους τους όρους που έχουν κάποια σχέση με την βιολογία και περιέχονται στο κείμενο αυτό συγκεντρωμένους, οπότε η ανακάλυψη του περιεχομένου του κειμένου γίνεται πιο εύκολη. Η διαδικασία είναι η παρακάτω: Αρχικά ο χρήστης δίνει σαν είσοδο το κείμενο στο οποίο θα γίνει ο εντοπισμός των οντοτήτων, εκτελείται η διαδικασία της τοποθέτησης ετικετών και τέλος επιστρέφονται τα αποτελέσματα στο χρήστη. Η διαδικασία αυτή απεικονίζεται και στο διάγραμμα της Εικόνας 3.2.



Εικόνα 3.2. Το διάγραμμα της εφαρμογής 'Χαρακτηρισμός Οντοτήτων'.

3.1. Αναζήτηση Βιοϊατρικών Κειμένων

Όπως αναφέρθηκε και παραπάνω η πρώτη εφαρμογή που θα κατασκευαστεί θα έχει ως στόχο την παρουσίαση των βιοϊατρικών κειμένων τα οποία θα ταιριάζουν στα κριτήρια αναζήτησης που θέτει ο χρήστης. Για να κατασκευαστεί αυτή η εφαρμογή θα χρειαστούμε το πακέτο λογισμικού Lucene (βλέπε κεφάλαιο 2.2) καθώς και μια συλλογή βιοϊατρικών κειμένων (βλέπε κεφάλαιο 2.4). Αφού γίνει η εγκατάσταση του Lucene (Παράγραφος 2.2.4) και η αποθήκευση των κειμένων στο δίσκο, μπορεί να αρχίσει η δημιουργία της εφαρμογής.

Το πρόγραμμα της εφαρμογής θα είναι γραμμένο σε Java, και για να γίνει η συγγραφή του προγράμματος πιο εύκολη θα χρησιμοποιηθεί ένα πρόγραμμα συγγραφής προγραμμάτων Java, το NetBeans.

Η εφαρμογή αυτή θα αποτελείται από δυο προγράμματα: το πρώτο σκέλος είναι η δεικτοδότηση των κειμένων και το δεύτερο η αναζήτηση των ανάλογων κειμένων σύμφωνα με τις λέξεις κλειδιά.

3.1.1. Περιγραφή της Εφαρμογής

Η εφαρμογή ‘Αναζήτηση Βιοϊατρικών Κειμένων’ έχει ως σκοπό της, όπως δηλώνει και το όνομα της, την αναζήτηση κειμένων. Αυτό που απαιτείται αρχικά είναι μια συλλογή από κείμενα, που μπορούν να είναι αποθηκευμένα σε ένα αρχείο στο σύστημα. Στην συγκεκριμένη εφαρμογή τα κείμενα έχουν αποθηκευτεί στο αρχείο C:\Medlars_Medical. Στόχος του χρήστη είναι η ανάκτηση των κειμένων από αυτή την συλλογή που τον ενδιαφέρουν, δηλαδή η ανάκτηση των κειμένων που περιέχουν όλες ή κάποιες από τις λέξεις που θέτει ο χρήστης ως κριτήρια αναζήτησης.

Για να μπορέσει να γίνει η ανάκτηση των κατάλληλων κειμένων πρέπει να πραγματοποιηθούν δυο διαδικασίες. Πρώτα θα πρέπει τα κείμενα της συλλογής να δεικτοδοτηθούν. Αυτό σημαίνει ότι το κάθε κείμενο θα μετατραπεί σε μια άλλη δομή που καλείται ευρετήριο. Αυτό γίνεται για να επιτευχθεί γρήγορη προσπέλαση των λέξεων που βρίσκονται στα κείμενα.

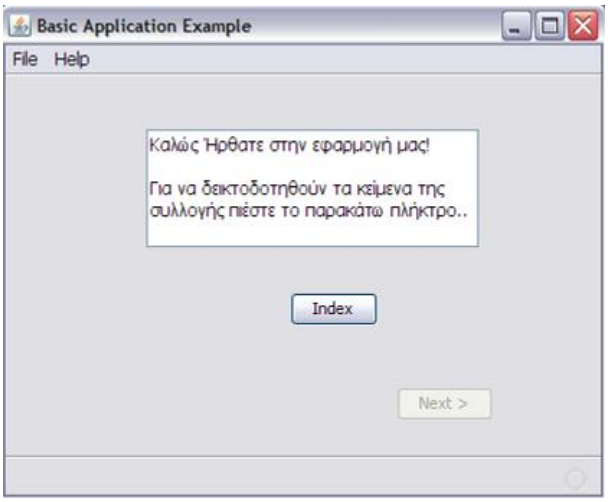
Στην συνέχεια θα πραγματοποιηθεί η διαδικασία της αναζήτησης. Για να γίνει η αναζήτηση θα πρέπει να υπάρχουν τα κριτήρια της αναζήτησης πρώτα. Αυτά είναι στην ουσία λέξεις ή φράσεις που ενδιαφέρουν το χρήστη να υπάρχουν στα κείμενα που θα ανακτηθούν. Αφού δηλωθούν οι λέξεις κλειδιά πραγματοποιείται η αναζήτηση των λέξεων αυτών στα ευρετήρια, και στην συνέχεια η εμφάνιση των αποτελεσμάτων.

Παρακάτω θα αναλυθεί η εφαρμογή βήμα-βήμα.

3.1.2. Η Δημιουργία του Ευρετηρίου

Η δημιουργία του ευρετηρίου αποτελεί το πιο σημαντικό κομμάτι της εφαρμογής. Στην ουσία πάνω σ’ αυτό το κομμάτι στηρίζεται η καλή λειτουργία του προγράμματος, διότι όπως έχει αναφερθεί και πιο πάνω η δημιουργία ευρετηρίων με βάση τα κείμενα κάνει την αναζήτηση πιο εύκολη συγκριτικά με την ανάγνωση ολόκληρων των κειμένων για την εύρεση των λέξεων κλειδιών μέσα σ’ αυτά.

Το πρώτο παράθυρο της εφαρμογής δεν είναι απλά ένα παράθυρο καλωσορίσματος στην εφαρμογή, αλλά το παράθυρο που θα γίνει η επιλογή του αρχείου με τα κείμενα που θα χρησιμοποιηθούν για την αναζήτηση. (Εικόνα 3.3)

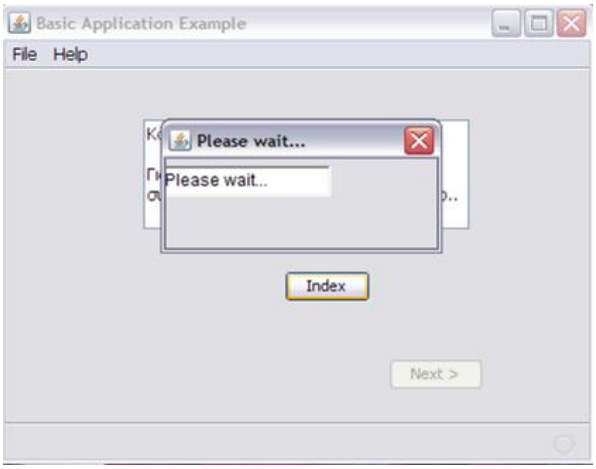


Εικόνα 3.3. Το αρχικό παράθυρο της εφαρμογής

Το παράθυρο αυτό (όπως φαίνεται και από την παραπάνω εικόνα) έχει μια περιοχή κειμένου και δύο κουμπιά. Στην περιοχή κειμένου αναγράφεται ένα μήνυμα καλωσορίσματος αλλά και ένα μικρό κείμενο με οδηγίες, το οποίο πληροφορεί το χρήστη ότι για να συνεχίσει στην εφαρμογή θα πρέπει να πιέσει το κουμπί που βρίσκεται ακριβώς κάτω από την περιοχή του κειμένου. Επίσης δηλώνεται ότι μέσω του κουμπιού αυτού θα γίνει η δεικτοδότηση των κειμένων.

Παρατηρώντας κανείς την Εικόνα 3.3, βλέπει ότι το δεύτερο πλήκτρο, το πλήκτρο με το όνομα “Next >”, είναι ανενεργό. Αυτό σημαίνει δεν υπάρχει η δυνατότητα επιλογής του την παρούσα στιγμή, και επομένως υπάρχουν δύο εναλλακτικές, οι οποίες είναι είτε να συνεχιστεί η εφαρμογή πιέζοντας το πλήκτρο “Index” είτε να τερματιστεί από το κουμπί “X” του παραθύρου.

Εάν υπάρξει μια κανονική ροή στο πρόγραμμα, τότε θα επιλεχθεί το κουμπί “Index”. Σ’ αυτή την περίπτωση θα ανοίξει ένα παράθυρο ‘Επιλογής Αρχείου’. Από αυτό το παράθυρο μπορεί ο χρήστης της εφαρμογής να επιλέξει το αρχείο με τα κείμενα, στα οποία θέλει να κάνει την αναζήτηση. Αφού γίνει η επιλογή του αρχείου τότε στην οθόνη εμφανίζεται ένα παράθυρο αναμονής, όπως φαίνεται στην Εικόνα 3.4.



Εικόνα 3.4. Το παράθυρο αναμονής μετά την επιλογή του αρχείου για δεικτοδότηση.

Όπως μπορεί κανείς να παρατηρήσει το παράθυρο αναμονής δεν έχει κάποια ενεργή επιλογή, και αυτό γιατί θα κλείσει μόνο του όταν τελειώσει η διαδικασία που εκτελείται παρασκηνιακά. Επομένως ο σκοπός του παραθύρου είναι να πληροφορήσει το χρήστη ότι επιτελείται μια εργασία και θα πρέπει να περιμένει εωσότου τελειώσει αυτή.

Η διαδικασία που εκτελείται παρασκηνιακά είναι η δημιουργία του ευρετηρίου. Υπάρχουν τα εξής βασικά βήματα που θα πρέπει κάποιος να ακολουθήσει στην εφαρμογή αυτή:

- Άνοιξε το `IndexWriter`
- Πρόσθεσε όλα τα κείμενα (`Documents`) που χρειάζεσαι
- Κλείσε το `IndexWriter`.

Παρακάτω επισυνάπτεται ένα κομμάτι από τον κώδικα:

```
final File INDEX_DIR = new File("index");

final File docDir = new File(spath);
      .
      .
      .
IndexWriter writer = new IndexWriter(INDEX_DIR, new
       StandardAnalyzer(), true);

indexDocs(writer, docDir);
writer.close();
```

Στην πρώτη σειρά του κώδικα δημιουργείται το ευρετήριο στη διεύθυνση `index`, ενώ στην δεύτερη γραμμή δηλώνεται που θα αποθηκευτούν τα κείμενα αφού υποστούν την διαδικασίας της δημιουργίας ευρετηρίων.

Η κλάση `IndexWriter` αποτελεί το κεντρικό συστατικό για την διαδικασία της δημιουργίας ευρετηρίων. Αυτό που στην ουσία κάνει αυτή η κλάση είναι να δημιουργεί ένα νέο ευρετήριο και να προσθέτει έγγραφα σ' έναν ήδη υπάρχων ευρετήριο (`INDEX_DIR`). Η `IndexWriter` μπορεί κανείς να την παρομοιάσει ως ένα αντικείμενο που του δίνει την δυνατότητα να γράψει σ' ένα ευρετήριο, αλλά δεν του επιτρέπει είτε να διαβάσει είτε να ψάξει γι' αυτό. Ο κατασκευαστής αυτής της κλάσης είναι ο εξής:

```
public IndexWriter(Directory d,
                   Analyzer a,
                   boolean create)
```

Όπως κανείς μπορεί να παρατηρήσει ο κατασκευαστή παίρνει τρία ορίσματα: Το d, το οποίο είναι η διεύθυνση του ευρετήριο· το a, που είναι ο 'αναλυτής' που θα χρησιμοποιηθεί και· τέλος ένα λογικό όρισμα, create, το οποίο όταν παίρνει την τιμή true (αληθής) μπορεί να δημιουργηθεί το ευρετήριο ή να γραφτεί το καινούργιο ευρετήριο πάνω από έναν ήδη υπάρχων, ενώ όταν έχει την τιμή false (ψευδής) προσαρτήσει το ευρετήριο σε έναν υπάρχων.

Στον κώδικα της εφαρμογής μας για τον `IndexWriter` υπάρχει η εξής αρχικοποίηση για τα ορίσματα του. Το d παίρνει την τιμή `INDEX_DIR`, το a παίρνει την τιμή `StandardAnalyzer()`, και το create ισούται με `true`.

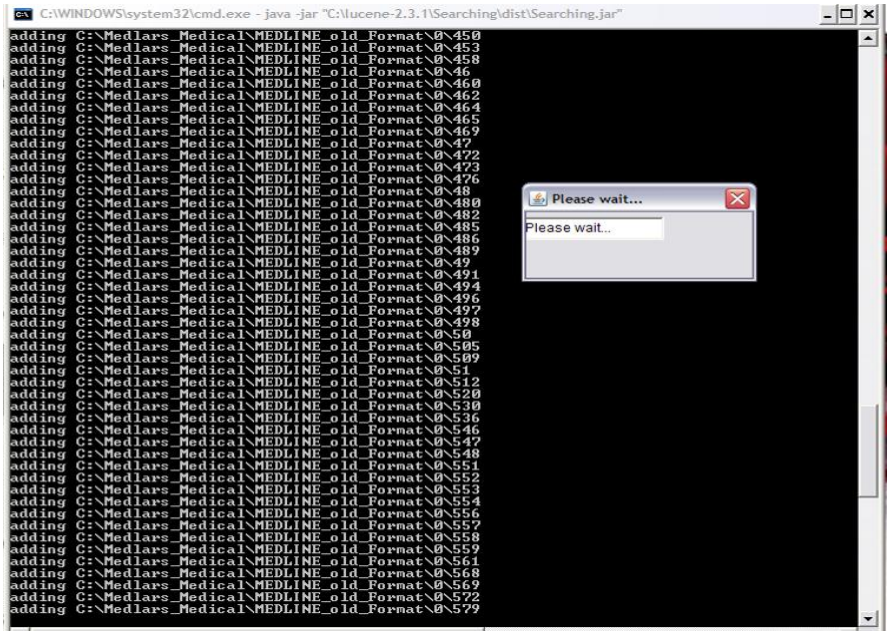
Η `StandardAnalyzer` είναι ευρέως αναγνωρίσιμη ως η γενικότερα χρήσιμη ενσωματωμένη συσκευή ανάλυσης. Μια γραμματική βασισμένη σε JavaCC κρύβεται κάτω από αυτή, που περιέχει τους ακόλουθους λεξικολογικούς τύπους: αλφαριθμητικά, αρκτικόλεξα, ονόματα επιχειρήσεων, διευθύνσεις ηλεκτρονικού ταχυδρομείου, ονόματα οικοδεσποτών υπολογιστών, αριθμούς, λέξεις με μια εσωτερική απόστροφο, αύξοντες αριθμούς, διευθύνσεις IP, και χαρακτήρες CJK (Chinese Japanese Korean - Κινεζικούς Ιαπωνικούς Κορεάτικους). Το `StandardAnalyzer` επιτελεί επίσης την διαδικασία της αφαίρεση των κοινών λέξεων (stop word) που περιλαμβάνονται στην ερώτηση του χρήστη, χρησιμοποιώντας έναν μηχανισμό που περιέχει μια λίστα με όμοιες προεπιλεγμένες Αγγλικές λέξεις, και ένα προαιρετικό κατασκευαστή `String[]` για την υπερκάλυψη λέξεων που μπορεί να προσθέσει ο δημιουργός της εφαρμογής στην λίστα. Η λίστα με τις κοινές αγγλικές λέξεις είναι οι εξής: "a, an, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, s, such, t, that, the, their, then, there, these, they, this, to, was, will, with". Αυτές οι λέξεις ακόμη και εάν υπάρχουν στα κριτήρια

αναζήτησης του χρήστη θα αφαιρεθούν από το πρόγραμμα και δε θα αποτελούν πλέον κριτήρια. Τέλος έχει την δυνατότητα να αναγνωρίζει τους πεζούς χαρακτήρες

Κάτι το οποίο αξίζει να αναφερθεί σ' αυτό το σημείο είναι ότι η εφαρμογή αυτή θα μπορούσε να αναπτυχθεί και για ελληνικά αρχεία. Για αυτό το σκοπό χρειάζονται, αρχικά, μια συλλογή ελληνικών κειμένων, όπως και επίσης μια λίστα με κοινές ελληνικές λέξεις όπως άρθρα, σύνδεσμοι και άλλες λέξεις που συναντιόνται συχνά μέσα σε κείμενα. Σ' αυτή την περίπτωση βέβαια θα αλλάξουν και τα κριτήρια αναζήτησης, όπως είναι λογικό, και θα είναι ελληνικά.

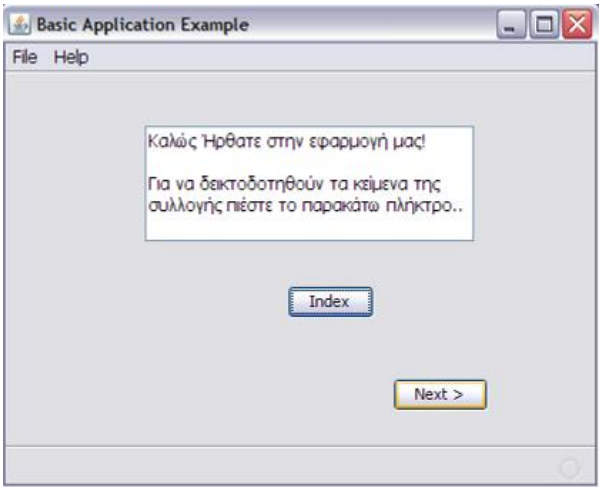
Το `indexDocs(writer, docDir)` είναι μια μέθοδος που δημιουργήθηκε με σκοπό, όπως δηλώνει και το όνομα της, να επιτελέσει την απλή από εδώ κι πέρα λειτουργία της ευρετηρίασης των εγγράφων και της αποθήκευσης αυτών στην διεύθυνση `docDir`.

Στην Εικόνα 3.5 παρουσιάζονται τα αποτελέσματα της διαδικασίας της δημιουργίας ευρετηρίων όπως αυτή φαίνεται από την γραμμή εντολών.

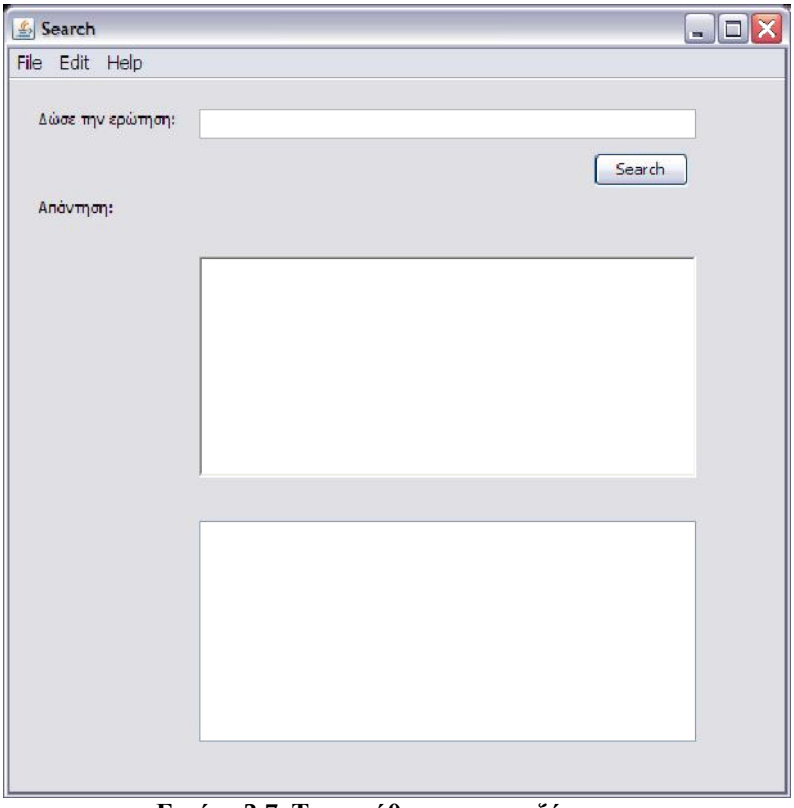


Εικόνα 3.5. Ένα στιγμιότυπο της δεικτοδότησης από την γραμμή εντολών.

Μετά την διαδικασία αυτή υπάρχει μια διαφορά στο αρχικό παράθυρο της εφαρμογής (Εικόνα 3.6.). Πλέον το κουμπί "Next >" είναι ενεργό. Αυτό σημαίνει ότι υπάρχει η δυνατότητα να συνεχιστεί η εφαρμογή παρακάτω. Αν επιλεχθεί το κουμπί αυτό τότε εμφανίζεται στην οθόνη το επόμενο παράθυρο της εφαρμογής (Εικόνα 3.7).



Εικόνα 3.6. Το παράθυρο της εφαρμογής μετά την διαδικασία της δεικτοδότησης.



Εικόνα 3.7. Το παράθυρο της αναζήτησης.

3.1.3. Η αναζήτηση

Ο στόχος της εργασίας αυτής είναι η αναζήτηση λέξεων κλειδιών σε βιοϊατρικά κείμενα. Η ευρετηρίαση αποτελεί ένα προπαρασκευαστικό τμήμα για την διαδικασία της αναζήτησης. Η αναζήτηση που παρέχει το API του Lucene είναι αρκετά απλή. Βέβαια αυτή μπορεί να αναπτυχθεί με διάφορους τρόπους αλλάζοντας διάφορα χαρακτηριστικά της. Για παράδειγμα, τα έγγραφα τα οποία έχουν το

μεγαλύτερο σκορ, όσο αναφορά το ταίριασμα τους με τους όρους της αναζήτησης, να εμφανίζονται πρώτα στην λίστα των αποτελεσμάτων· αυτό είναι κάτι το οποίο ενσωματώθηκε στην παρούσα εργασία. Σ' αυτό το σημείο θα ήταν χρήσιμο να αναφερθεί ο τρόπος που το Lucene υπολογίζει το σκορ. Ο υπολογισμός του σκορ γίνεται τυπικά από το μέτρο του συνημίτονου. Όπως δηλώνει και το όνομά του, το μέτρο υπολογίζει την ομοιότητα ενός κειμένου με ένα ερώτημα ως το συνημίτονο της γωνίας μεταξύ δύο διανυσμάτων. Τα διανύσματα αυτά αντιστοιχούν στην ερώτηση και το κείμενο και ορίζονται σε ένα διανυσματικό χώρο με τόσες διαστάσεις όσες οι διαφορετικές λέξεις σε κείμενο και ερώτηση. Αυτό είναι, ένα προϊόν ομαλοποιημένου μήκους σημείων των διανυσμάτων ερωτήσεων και εγγραφών. Ο τύπος είναι ο παρακάτω:

$$score(\vec{doc}, \vec{query}) = \cos ine(\vec{doc}, \vec{query}) = (\vec{doc} * \vec{query}) / (length(\vec{doc}) * lenght(\vec{query}))$$

όπου τα διανύσματα εγγράφων και ερωτήσεων παράγονται από τους εξής τύπους:

$$\vec{doc}[term] = tf(term, doc)$$

$$\vec{query}[term] = tf(term, query) * idf(term, numDocs)$$

με το *numDocs* να είναι ο συνολικός αριθμός των εγγράφων που υπάρχουν στο σώμα της συλλογής. Από τους παραπάνω τύπους μπορεί κανείς να παρατηρήσει ότι η κανονικοποιημένη IDF εφαρμόζεται μόνο στο διάνυσμα ερώτησης. Ο λόγος που συμβαίνει αυτό είναι γιατί εάν οι συχνότητες εγγράφου και ο αριθμός των εγγράφων συνεχώς άλλαζαν, τότε θα έπρεπε το μήκος ενός διανύσματος εγγράφων να αλλάζει εάν είχε ομαλοποιήθηκε με τη IDF. Σε αυτήν την περίπτωση, κάθε έγγραφο στο σύνολο αποτελεσμάτων θα έπρεπε να υποστεί ξανά την διαδικασία της κανονικοποίησης για κάθε ερώτηση, κάτι το οποίο θα απαιτούσε να γίνουν πολλαπλασιασμοί και προσθήκες για κάθε όρο στο έγγραφο. Αλλά αυτή είναι μια πολυέξοδη διαδικασία. (Για περισσότερες πληροφορίες όσο αναφορά στους όρους TF και IDF, μπορεί κανείς να ανατρέξει στο κεφάλαιο 1.)

Το παράθυρο της αναζήτησης (Εικόνα 3.7) έχει ένα πεδίο στο οποίο μπορεί ο χρήστης της εφαρμογής να γράψει την έκφραση για την οποία θέλει να κάνει την αναζήτηση, και κάτω από αυτό υπάρχει ένα κουμπί με το όνομα "Search", το οποίο είναι και το μοναδικό κουμπί που υπάρχει στην φόρμα του παραθύρου αυτού. Επίσης υπάρχει ένα πεδίο στο οποίο θα εμφανιστούν τα αποτελέσματα της αναζήτησης σε μορφή λίστας και ένα ακόμη κάτω από αυτό στο οποίο εμφανίζεται το κείμενο ενός εγγράφου, από τα αποτελέσματα της αναζήτησης, που επιλέγει ο χρήστης. Εάν κάποιος επιθυμεί να εξέρθει από την εφαρμογή θα πρέπει να κλείσει το αρχικό

παράθυρο (Εικόνα 3.6), καθώς εάν κλείσει αυτό το παράθυρο (Εικόνα 3.7) επιστρέφει στο προηγούμενο και η εφαρμογή δεν τερματίζεται.

Κάτω από το κουμπί “Search” ‘κρύβεται’ η διαδικασία της αναζήτησης. Πρόκειται για ένα κώδικα γραμμένο σε Java, ο οποίος εκτελεί την αναζήτηση της έκφρασης που τοποθετείται στο πεδίο ‘Δώσε την ερώτηση’ στα κείμενα που έχουν υποστεί την διαδικασία της δεικτοδότησης. Τα αποτελέσματα αυτής της ανίχνευσης εμφανίζονται στο επόμενα πεδίο της φόρμας. Παρακάτω υπάρχει ένα απόσπασμα του κώδικα που έχει χρησιμοποιηθεί για την αναζήτηση.

```
IndexReader reader = IndexReader.open(index);
Searcher searcher = new IndexSearcher(reader);
Analyzer analyzer = new StandardAnalyzer();

QueryParser parser = new QueryParser(field, analyzer);

String line = jTextField1.getText();

line = line.trim();

org.apache.lucene.search.Query query = parser.parse(line);
Hits hits = searcher.search(query);
resultsLabel.setText(hits.length() + " total matching documents");
resultsLabel.setVisible(true);
final int HITS_PER_PAGE = 10;
for (int start = 0; start < hits.length(); start += HITS_PER_PAGE)
{
   int end = Math.min(hits.length(), start + HITS_PER_PAGE);
   for (int i = start; i < end; i++) {
         org.apache.lucene.document.Document doc = hits.doc(i);
         String path = doc.get("path");
         if (path != null) {
                  list1.add(path);
         } else {
                  list1.add("No path for this document");
         } //end if
   } //end for
} //end for
reader.close();
```

Η κλάση `IndexReader` είναι μια αφηρημένη κλάση, που παρέχει μια διεπαφή για την πρόσβαση ενός ευρετηρίου. Η αναζήτηση στο ευρετήριο γίνεται εξ’ ολοκλήρου μέσω αυτής της αφηρημένης διεπαφής, έτσι ώστε οποιαδήποτε υποκλάση που την υλοποιεί να προσφέρει αυτή τη δυνατότητα.

Στην δεύτερη γραμμή του κώδικα που παρουσιάζεται εδώ, δημιουργείται ένα αντικείμενο τύπου `Searcher`. Η κλάση `Searcher` είναι μια αφηρημένη κατηγορία βάσεων για τις εφαρμογές αναζήτησης και εφαρμόζει τις κύριες μεθόδους αναζήτησης. Εδώ το αντικείμενο αρχικοποιείται ως αντικείμενο της

`IndexSearcher`, η οποία είναι μια υποκλάση της `Searcher`. Η `IndexSearcher` στην ουσία δημιουργεί έναν ανιχνευτή για την έρευνα του ορίσματός της, δηλαδή του ευρετηρίου.

Έπειτα δημιουργείται ένα αντικείμενο τύπου `QueryParser`. Αυτό που στην ουσία πραγματοποιεί η `QueryParser` είναι η ανάλυση μιας έκφρασης ερώτησης, δηλαδή η μετατροπή της ερώτησης του χρήστη σε ένα κατάλληλο προδιαγεγραμμένο αντικείμενο τύπου `Query`. Η `QueryParser` απαιτεί έναν 'αναλυτή' για να σπάσει σε όρους τα κομμάτια της ερώτησης. Στον κώδικα αυτό, χρησιμοποιείται η `StandardAnalyzer`. (Για τα γνωρίσματα της έχουμε αναφερθεί στην προηγούμενη ενότητα, 3.1.2).

Στην γραμμή του κώδικα `org.apache.lucene.search.Query query = parser.parse(line)` γίνεται η ανάλυση για τους όρους της αναζήτησης. Για παράδειγμα, έστω ότι ο χρήστης δίνει τη φράση "`electron microscopy of lung or bronchi`". Η ερώτηση αυτή αποθηκεύεται στην μεταβλητή '`line`'. Για να γίνει ένα αντικείμενο της κλάσης '`Object`' θα πρέπει να εκτελεστεί η παραπάνω εντολή. Δηλαδή θα εφαρμοστεί πάνω στην φράση αυτή η `QueryParser`. Μετά την διαδικασία αυτή η αναζήτηση θα γίνει για κείμενα που θα περιέχουν τις λέξεις: "`electron microscopy lung bronchi`", ή εάν εκφραστεί με Boolean σύνταξη θα είναι '`electron`' `or` '`microscopy`' `or` '`lung`' `or` '`bronchi`'. Δηλαδή θα αναζητηθούν κείμενα τα οποία θα περιέχουν είτε και τις τέσσερεις λέξεις, είτε τις τρεις (με όλους τους δυνατούς συνδυασμούς), είτε τις δύο, ή ακόμη και μια από τις τέσσερεις λέξεις.

Η εφαρμογή θα μπορούσε να είναι γραμμένη, βέβαια, έτσι ώστε να υποστηρίξει και κάποιον άλλο τύπο αναζήτησης. Παραδείγματος χάριν θα μπορούσε η αναζήτηση να γίνει αποκλειστικά για κείμενα που περιέχουν και τις τέσσερεις λέξεις, για το προηγούμενο παράδειγμα. Σ' αυτή την περίπτωση θα άλλαζε ο κώδικας και συγκεκριμένα θα γίνονταν ο εξής:

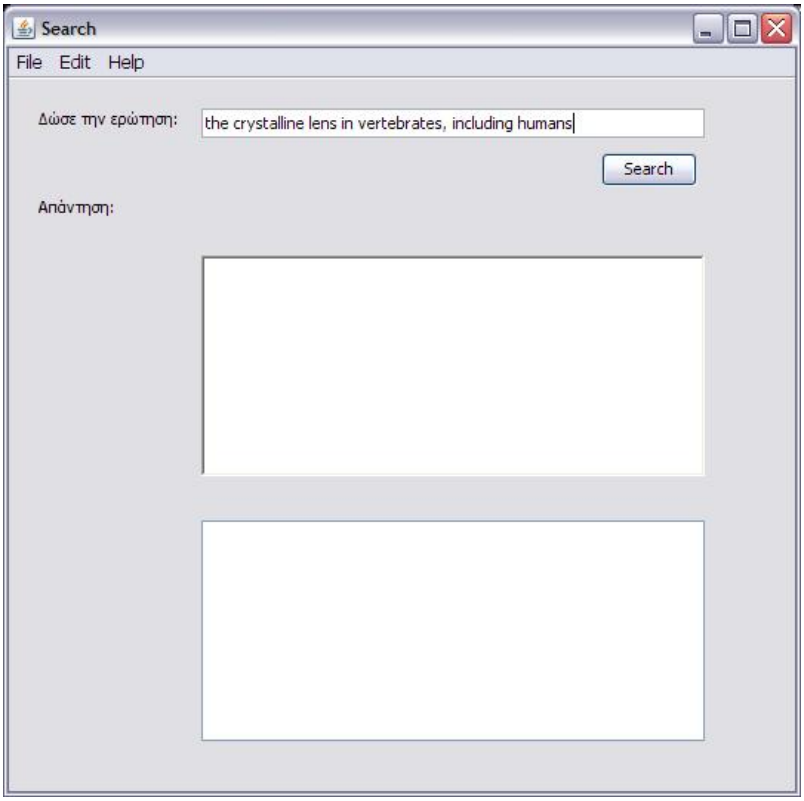
```
QueryParser parser = new QueryParser(field, analyzer);
Parser.setOperator(QueryParser.DEFAULT_OPERATOR_AND);
```

Βέβαια υπάρχουν και άλλοι τρόποι για τον ορισμό της αναζήτησης, όπως επίσης μπορεί να υπάρξει και συνδυασμός κάποιον από αυτών των μεθόδων, για πιο πολύπλοκες αναζητήσεις.

Η κλάση `Hits` αποτελεί μια ταξινομημένη λίστα εγγράφων, που χρησιμοποιείται για την προσωρινή αποθήκευση των αποτελεσμάτων της αναζήτησης.

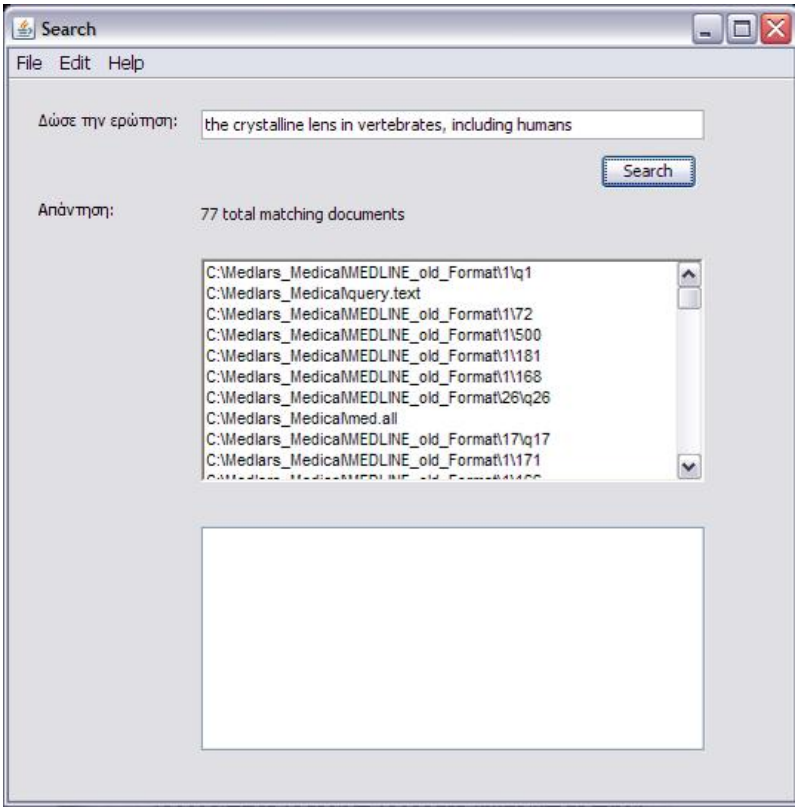
Η κλάση `Document` είναι η μονάδα της ευρετηρίασης και αναζήτησης. Ένα `Document` είναι ένα σύνολο τομέων. Οι τομείς αυτοί απεικονίζουν το έγγραφο ή τα μεταδεδομένα που έχουν κάποια σχέση με το έγγραφο. Στο αντικείμενο της κλάσης αυτής αποθηκεύετε το έγγραφο που ταιριάζει με τα κριτήρια αναζήτησης και στην συνέχεια ο κώδικας που ακολουθεί εμφανίζει τα αποτελέσματα αυτά στην λίστα.

Παρακάτω παρατίθεται ένα παράδειγμα για την εκτέλεση αυτής της εφαρμογής. Αρχικά ο χρήστης γράφει στο κατάλληλο πεδίο την έκφραση για την οποία θα πραγματοποιηθεί η αναζήτηση, σ' αυτή την περίπτωση η έκφραση είναι η "`the crystalline lens in vertebrates, including humans`" (Εικόνα 3.8). Κατόπιν ο χρήστης πατάει το κουμπί "Search" και πραγματοποιείται η διαδικασία της αναζήτησης όπως περιγράφηκε παραπάνω. Για να οριστεί το αντικείμενο της `QueryParser`, όπως ειπώθηκε και προηγουμένως, χρειάζεται να οριστεί ως όρισμα αυτής και η `StandardAnalyzer`. Μια από τις λειτουργίες της `StandardAnalyzer` είναι η αφαίρεση των κοινών λέξεων (όπως άρθρα), οπότε μετά την εφαρμογή της η έκφραση για την οποία πραγματοποιείται η αναζήτηση είναι η "`crystalline lens vertebrates including humans`".



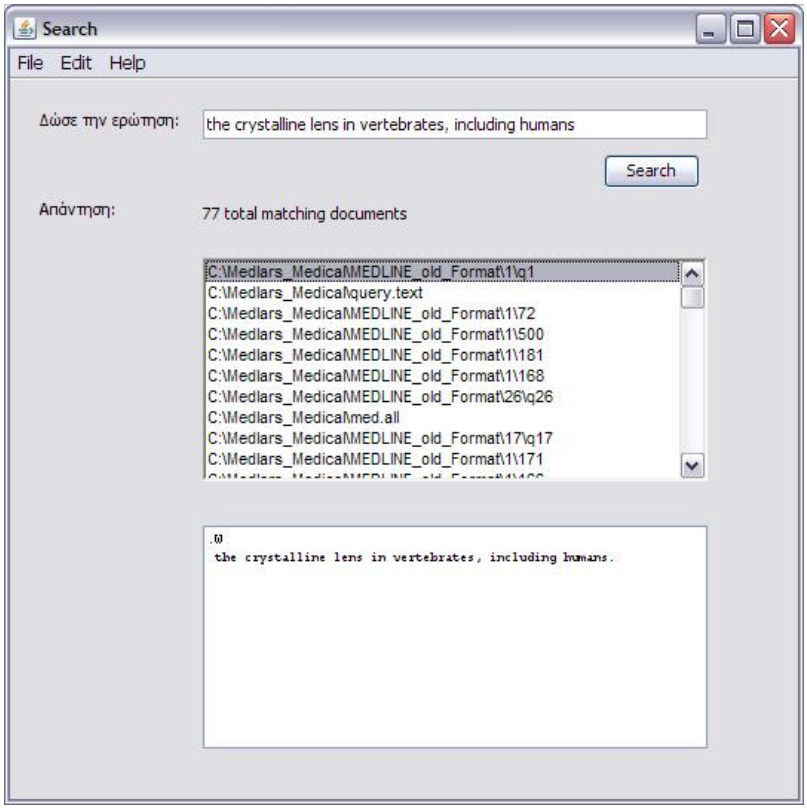
Εικόνα 3.8. Η αναζήτηση για την έκφραση "the crystalline lens in vertebrates, including humans."

Αφού πραγματοποιηθεί η αναζήτηση εμφανίζονται τα αποτελέσματα της σε μια λίστα (Εικόνα 3.9). Παρατηρείται όμως και η εμφάνιση ενός πεδίου πάνω από τη περιοχή της λίστα, το οποίο δεν υπήρχε πριν. Σ' αυτό το πεδίο αναγράφεται το σύνολο των εγγράφων, τα οποία βρέθηκαν να ταιριάζουν με τα κριτήρια αναζήτησης.



Εικόνα 3.9. Τα αποτελέσματα της αναζήτησης για την παραπάνω έκφραση.

Σ' αυτό το σημείο θα ήταν το ιδανικό για το χρήστη να μπορεί να διαβάσει τα κείμενα που του επιστρέφει η αναζήτηση για να κρίνει και μόνος του τα αποτελέσματα. Αυτό μπορεί να επιτευχθεί πατώντας διπλό κλικ με το ποντίκι του πάνω στο μονοπάτι που αντιστοιχεί σε κάθε έγγραφο. Τότε το κείμενο του εγγράφου παρουσιάζεται στο πεδίο που βρίσκεται κάτω από τη λίστα (Εικόνα 3.10).



Εικόνα 3.10. Η εμφάνιση ενός κειμένου απο τα αποτελέσματα της αναζήτησης (Συγκεκριμένα του πρώτου κειμένου που επιστρέφεται) .

Εκτελώντας ξανά την ίδια διαδικασία, δηλαδή διπλοκλικάροντας κάποιο άλλο μονοπάτι, μπορεί να δει και τα υπόλοιπα κείμενα που έχουν επιστραφεί ως απάντηση στην ερώτηση.

Η εφαρμογή σ' αυτό το σημείο έχει τελειώσει. Εάν ο χρήστης επιθυμεί να κάνει μια νέα αναζήτηση, χωρίς να αλλάξει την συλλογή των κειμένων, πολύ απλά γράφει την νέα του ερώτηση στο αντίστοιχο πεδίο και πατάει το πλήκτρο "Search".

3.2 Εντοπισμός Οντοτήτων σε Βιοϊατρικά Κείμενα

Ο εντοπισμός συγκεκριμένων οντοτήτων σε βιοϊατρικά κείμενα είναι η δεύτερη εφαρμογή που θα υλοποιήσουμε σ' αυτή την εργασία. Ο στόχος του προγράμματος αυτού είναι η τοποθέτηση ετικετών στις οντότητες του κειμένου, δηλαδή η εύρεση μέσα στο κείμενο λέξεων που αναφέρονται σε βιολογικούς όρους. Η δομή αυτής της εργασίας στηρίζεται στο LingPipe, το οποίο αναλύθηκε στην παράγραφο 2.3. Για να εκτελεστεί όμως χρειάζεται και ένα βιοϊατρικό κείμενο, το

οποίο θα είναι κάποιο από τα αποσπάσματα κειμένων της βάσης δεομένων της Medline του 2008 (παράγραφος 2.4).

Η πρώτη ενέργεια η οποία πρέπει να γίνει είναι η εγκατάσταση του LingPipe, όπως αυτή αναφέρεται στην παράγραφο 2.3.3, και η αποθήκευση των κειμένων σε κάποιο φάκελο.

Όπως και στην προηγούμενη εφαρμογή, έτσι και εδώ το πρόγραμμα θα είναι γραμμένο σε Java.

3.2.1 Περιγραφή της εφαρμογής

Πίσω από τον όρο του εντοπισμού οντοτήτων μπορούν να κρύβονται διάφορες εργασίες. Μία από αυτές είναι και ο εντοπισμός και χαρακτηρισμός οντοτήτων που περιέχονται σε βιοϊατρικά κείμενα, όπως γονίδια, πρωτεΐνες, και άλλα. Αυτός είναι και ο στόχος της εργασίας αυτής. Συγκεκριμένα αυτό που θα κάνει η εφαρμογή αυτή είναι αφού θα πάρει σαν όρισμα ένα βιοϊατρικό απόσπασμα κειμένου, να εντοπίσει μέσα σ' αυτό τις λέξεις που αναφέρονται σε βιοϊατρικούς όρους όπως είναι τα ονόματα γονιδίων, οργανισμούς και άλλα. Για να μπορέσει να γίνει η αναγνώριση αυτή το σύστημα θα πρέπει να εκπαιδευτεί. Δηλαδή θα πρέπει να περαστούν με κάποιο τρόπο οι λέξεις και η σχέση τους με τις βιοϊατρικές οντότητες σε κάθε περίπτωση (αυτό θα το αναλύσουμε παρακάτω, για κάθε υπό-εφαρμογή). Η εφαρμογή αυτή έχει δύο υπό-εφαρμογές, που αφορούν αντίστοιχα δύο διαφορετικές συλλογές κειμένων:

- Gene Tag
- Genia.

Και οι δυο αναγνωρίζουν και χαρακτηρίζουν τις οντότητες σε ένα βιοϊατρικό κείμενο. Η διαφορά που έχουν είναι ότι στηρίζονται σε διαφορετικά πρότυπα αναζήτησης. Συγκεκριμένα το Gene Tag στηρίζεται στο πρότυπο CharLmHmmChunker και το Genia στο TokenShapeChunker. Στόχος και των δύο είναι να εκπαιδεύσουν αρχικά τον ανιχνευτή οντοτήτων με προσημασμένα κείμενα και κανόνες για τον εντοπισμό πιθανών οντοτήτων και στη συνέχεια, να εφαρμόσουν το εκπαιδευμένο μοντέλο, δηλαδή τον αντίστοιχο Chunker, σε μια άγνωστη συλλογή χωρίς σήμανση. Σαν αποτέλεσμα προκύπτει η πρόσθεση στα κείμενα της συλλογής ετικέτες που επισημαίνουν τις οντότητες που εντοπίστηκαν και σε ορισμένες

περιπτώσεις προσφέρονται και πληροφορίες για την πιθανή κατηγοριοποίηση της κάθε οντότητας.

Το πρότυπο CharLmHmmChunker βασίζεται σε μια κωδικοποίηση μπλοκ ως ένα πρόβλημα τοποθέτησης ετικετών. Για την τοποθέτηση των ετικετών χρησιμοποιείται ένα γλωσσικό πρότυπο χαρακτήρων HMM[j], το οποίο χρησιμοποιεί τα πρότυπα για κάθε ετικέτα (κατάσταση) στο HMM, και ένα πρότυπο μέγιστης πιθανότητας μετάβασης σε μεγάλη γραμματοσειρά. Αυτό το πρότυπο είναι αρκετά απλό, αλλά συνάμα και μη ακριβής. Επίσης, είναι αρκετά γρήγορο εξαιτίας της χρήσης της εναποθήκευσης. Τέλος, διαθέτει καλή ανάκληση για την έξοδο εμπιστοσύνης και αρκετά αξιόπιστες εκτιμήσεις εμπιστοσύνης.

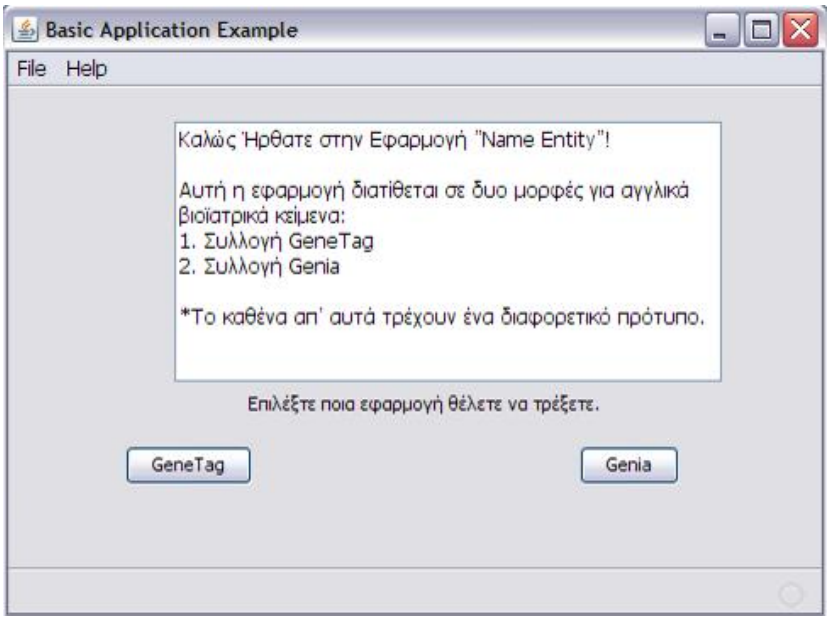
Αντίθετα, το πρότυπο TokenShapeChunker χρησιμοποιεί ένα παραγωγικό πρότυπο, που προβλέπει το επόμενο σύμβολο και ετικέτα βασισμένο στα δύο προηγούμενα σύμβολα και ετικέτες. Τυχών άγνωστες λέξεις αντικαθίστανται με χαρακτηριστικά σχήματα, που βασίζονται σε ένα βήμα του TokenCategorizer. Για την εκπαίδευση αυτού του προτύπου χρησιμοποιείται η κλάση TrainTokenShapeChunker. Πρόκειται για ένα πολύ γρήγορο πρότυπο, και η απόδοσή του είναι συνήθως η μισή απόσταση του προηγούμενου προτύπου. Στον Πίνακα 3.1 αναφέρονται συνοπτικά τα χαρακτηριστικά των δυο προτύπων.

Πίνακας 3.3. Συγκριτικός πίνακας των δύο προτύπων.

Πρότυπο	**Μέγεθος**	**Έξοδος**	
		Ταχύτητα	**Ακρίβεια**
CharLmHmmChunker	Μεσαίο	Μεγάλη	Χαμηλή
TokenShapeChunker	Μικρό	Μεγάλη	Μέτρια

Στην Εικόνα 3.11 απεικονίζεται το παράθυρο της εφαρμογής “Named Entity”. Σ’ αυτό το σημείο δίνεται η δυνατότητα επιλογής της υπό-εφαρμογής που θέλει κάποιος να τρέξει, για το χαρακτηρισμό των οντοτήτων. Παρακάτω θα αναλυθούν το καθένα ξεχωριστά.

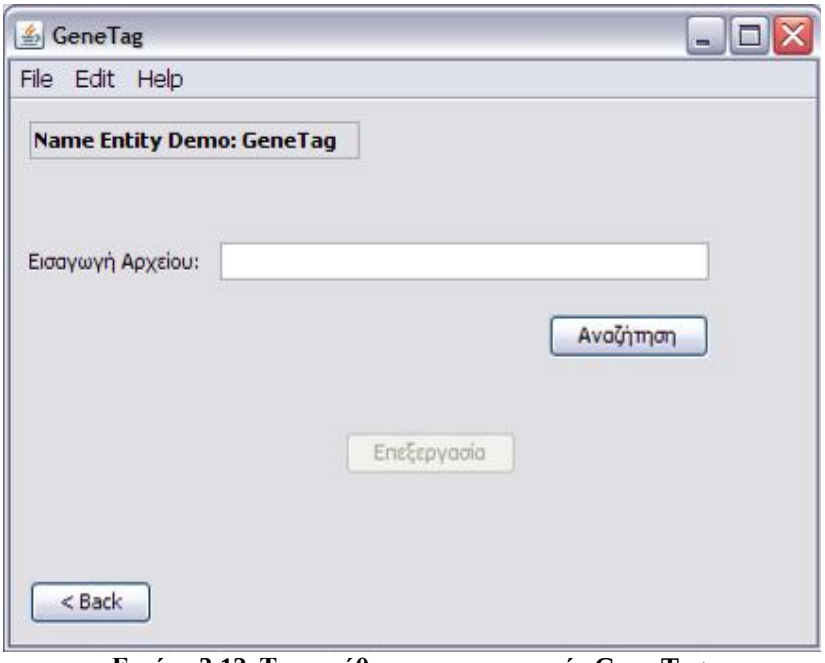
[j] Το HMM είναι συντομογραφία των λέξεων Hidden Markov Model. Ένα Hidden Markov Model είναι ένα στατιστικό μοντέλο που το σύστημα μοντελοποιείται για να αναλάβει μια Μαρκοβιανή διαδικασία με άγνωστες παραμέτρους. Η πρόκληση του μοντέλου αυτού είναι ο καθορισμός των κρυφών παραμέτρων από τα δεδομένα που παρατηρούνται. Ένα HMM μπορεί να θεωρηθεί ως ένα απλό δυναμικό δίκτυο Bayssian.



Εικόνα 3.11. Παράθυρο επιλογής προτύπου.

3.2.2. Η Εφαρμογή 'Gene Tag'

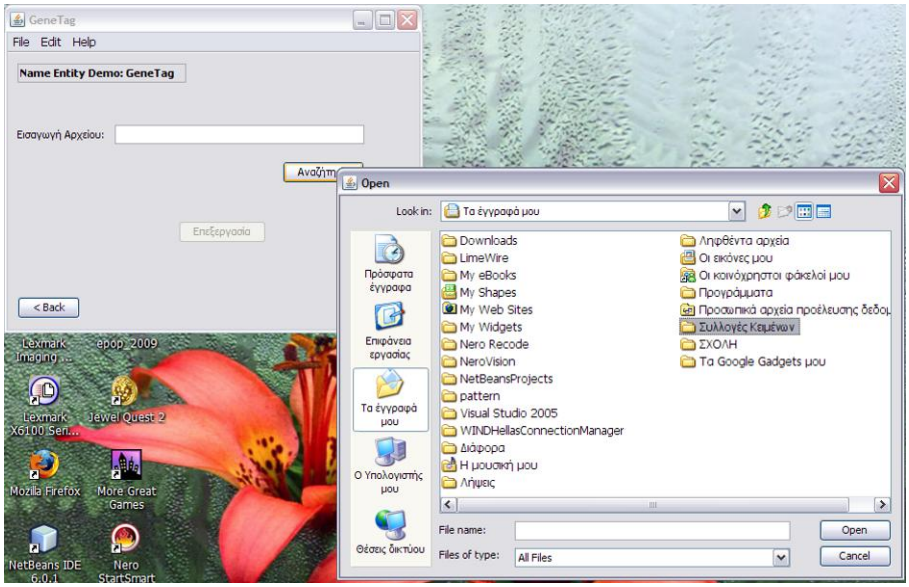
Εάν στην Εικόνα 3.11 επιλεχθεί το αριστερό κουμπί με όνομα 'Gene Tag', τότε θα ανοίξει το παράθυρο που φαίνεται στην παρακάτω εικόνα (Εικόνα 3.12).



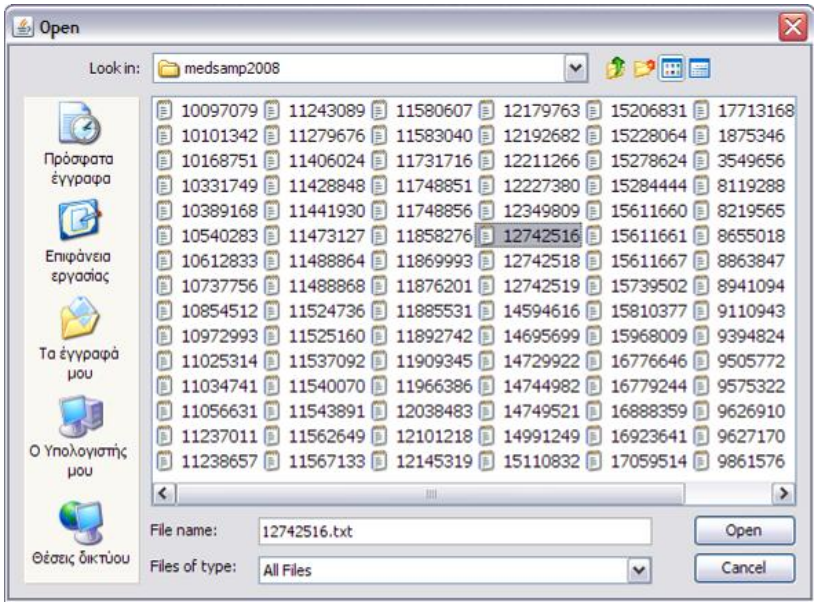
Εικόνα 3.12. Το παράθυρο της εφαρμογής Gene Tag.

Το παράθυρο αυτό είναι ένα απλό παράθυρο με δύο ενεργά κουμπιά, ένα μη ενεργό κουμπί και ένα πεδίο κειμένου. Στο κενό πεδίο κειμένου εισάγεται το μονοπάτι του αρχείου στο οποίο θα εφαρμοστεί η εφαρμογή. Το κουμπί ακριβώς

κάτω από το πεδίο αυτό υπάρχει για να διευκολύνει την εισαγωγή του αρχείου ανοίγοντας ένα παράθυρο επιλογής, όπως φαίνεται στην Εικόνα 3.13. Στην Εικόνα 3.14 φαίνονται τα κείμενα της συλλογής MedSamp2008, τα οποία χρησιμοποιούνται στην εφαρμογή αυτή.

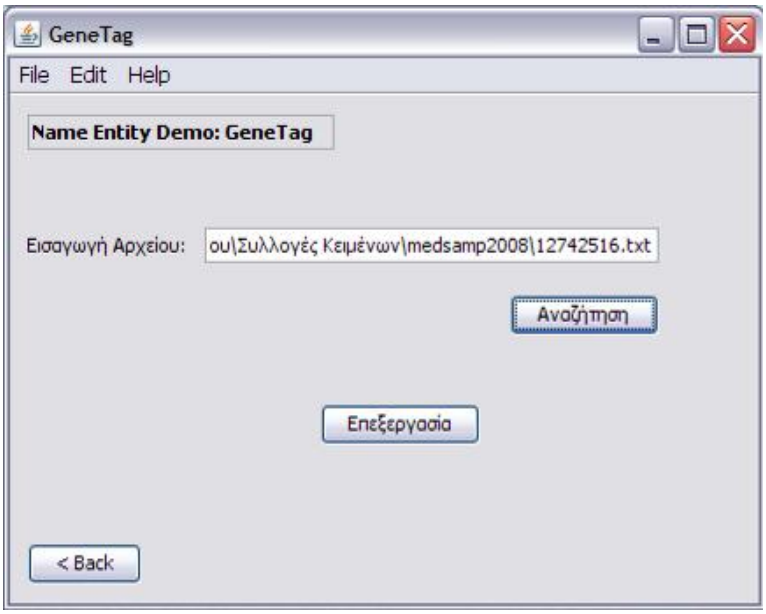


Εικόνα 3.13. Παράθυρο επιλογής αρχείου.



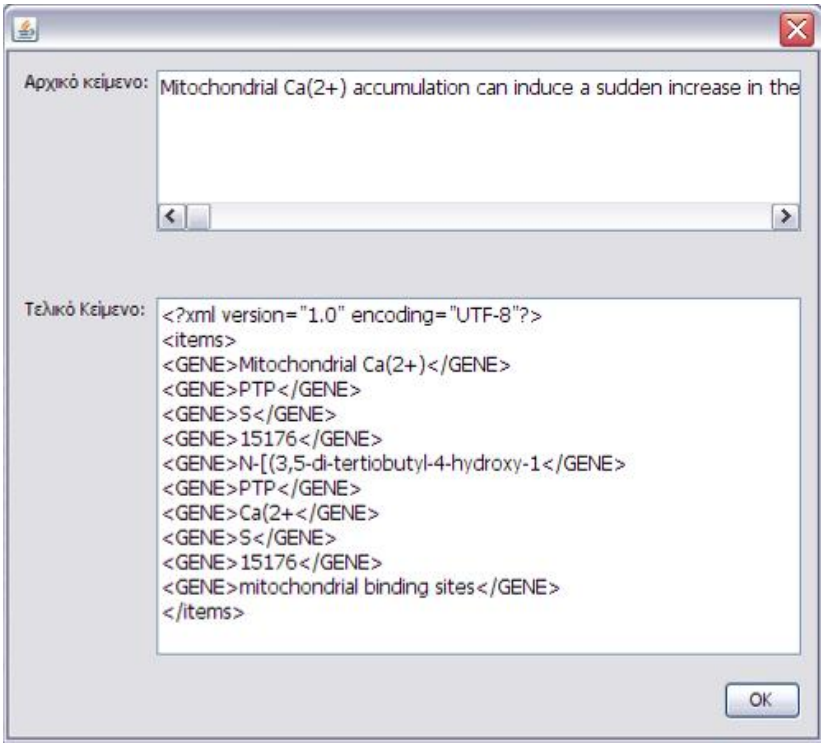
Εικόνα 3.14. Επιλογή κειμένου.

Επιλέγοντας ένα κείμενο από την συλλογή, για παράδειγμα το κείμενο '12742516.txt', και πατώντας το 'Open' της φόρμας αυτής, εμφανίζεται πάλι το παράθυρο της εφαρμογής (Εικόνα 3.12), αλλά με κάποιες διαφορές (Εικόνα 3.15).



Εικόνα 3.15. Το αρχικό παράθυρο της εφαρμογής αλλά με το κείμενο προς επεξεργασία, σαν όρισμα στο πεδίο κειμένου.

Οι διαφορές που υπάρχουν σε σχέση με πριν είναι δύο. Πρώτον το πεδίο κειμένου πλέον έχει τιμή, η οποία είναι η διεύθυνση που είναι αποθηκευμένο το κείμενο που θα γίνει ο χαρακτηρισμός των οντοτήτων. Και δεύτερον, όπως μπορεί κανείς να παρατηρήσει είναι ότι το κουμπί που πριν ήταν ανενεργό (με όνομα ‘Επεξεργασία’), τώρα είναι ενεργό και μπορεί να επιλεχθεί. Στην πραγματικότητα αν κάποιος θέλει να συνεχίσει την εφαρμογή και να δει τα αποτελέσματα θα πρέπει να επιλέξει το κουμπί αυτό. Τότε το αποτέλεσμα θα είναι αυτό που απεικονίζεται στην φόρμα της Εικόνας 3.16.



Εικόνα 3.16. Τα αποτελέσματα της εφαρμογής για το κείμενο '12742516'.

Το αρχικό κείμενο φαίνεται στη πρώτη περιοχή κειμένου, ενώ τα αποτελέσματα φαίνονται στην περιοχή κειμένου που βρίσκεται ακριβώς από κάτω. Τα αποτελέσματα είναι κάποιες λέξεις ή φράσεις του κειμένου, οι οποίες αναγνωρίστηκαν από το σύστημα ως οντότητες της βιολογίας. Το κομμάτι του κώδικα που είναι υπεύθυνο γι' αυτή την αναγνώριση είναι το παρακάτω:

```
Chunker chunker = null;
chunker = (Chunker) AbstractExternalizable.readObject(modelFile);
Chunking chunking = chunker.chunk(doc);

Set<Chunk> allChunks = chunking.chunkSet();
out.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
out.println("<items>");
Iterator it = allChunks.iterator();
while (it.hasNext()) {
        String geneinfo = it.next().toString();
        String[] tokens = geneinfo.split("[-:@]");
        int start = new Integer(tokens[0]).intValue();
        int end = new Integer(tokens[1]).intValue();
        String tag = tokens[2];
        out.println("<"+tag+">" + doc.substring(start,end) +
"</"+tag+">");
}
out.println("</items>");
out.close();
Scanner c = new Scanner(new FileInputStream("gene.xml"));
jTextArea2.setText("");
while (c.hasNextLine()) {
```

```
            jTextArea2.append(c.nextLine() + "\n");
}
```

Στην δεύτερη σειρά γίνεται η ανάγνωση του μοντέλου στην μνήμη, ενώ στην τρίτη ο τεμαχισμός του κειμένου σε τμήματα. Στην συνέχεια επιστρέφονται τα τμήματα των τμημάτων που παράχθηκαν νωρίτερα. Κατόπιν πραγματοποιείται μια επαναληπτική διαδικασία για κάθε κομμάτι και τα αποτελέσματα αυτά αποθηκεύονται σε ένα αρχείο με όνομα “gene.xml”. Τέλος τυπώνονται τα δεδομένα αυτά στην οθόνη, στην περιοχή κειμένου όπως φαίνεται στην παραπάνω εικόνα.

Ένα θέμα που τέθηκε και νωρίτερα και το οποίο είναι πολύ σημαντικό για την πραγματοποίηση της εφαρμογής, είναι η εκπαίδευση. Ο κώδικας για την εκπαίδευσηε είναι ο εξής:

```
File corpusFile = new File("genetag.tag");
File modelFile = new File("ne-en-bio-genetag.HmmChunker");

TokenizerFactory factory = new IndoEuropeanTokenizerFactory();
HmmCharLmEstimator hmmEstimator = new
            HmmCharLmEstimator(MAX_N_GRAM,NUM_CHARS,LM_INTERPOLATION);
CharLmHmmChunker chunkerEstimator = new
            CharLmHmmChunker(factory,hmmEstimator);

GeneTagParser parser = new GeneTagParser();
parser.setHandler(chunkerEstimator);
parser.parse(corpusFile);
File gene = new File("gene.txt");
AbstractExternalizable.compileTo(chunkerEstimator, modelFile);
```

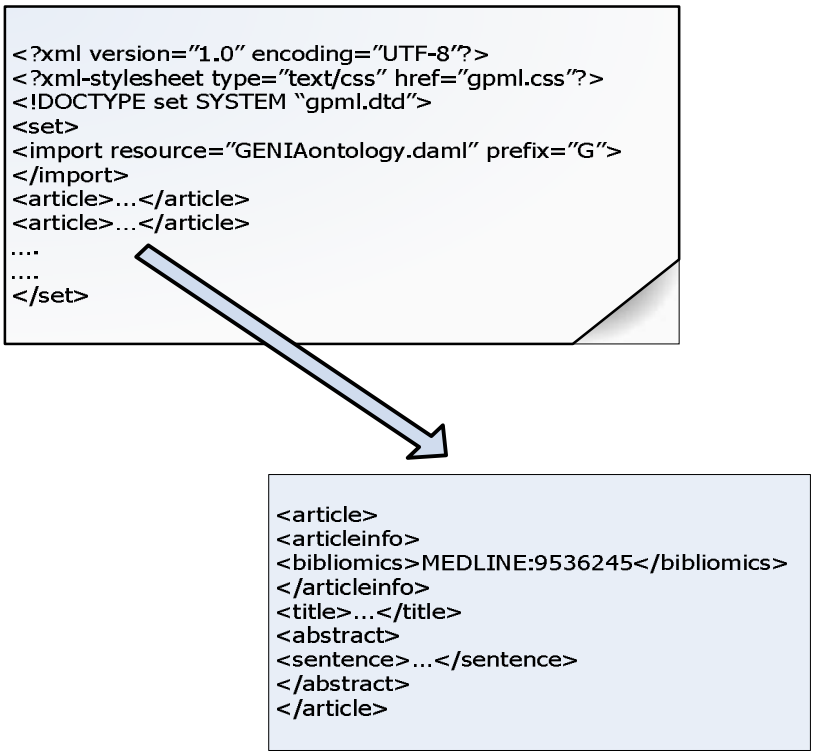
Αρχικά δημιουργούνται δύο αρχεία, το πρώτο χρησιμοποιείται για την εκπαίδευση του μοντέλου, και το δεύτερο χρησιμοποιείται για την εγγραφή του μοντέλου μετά την εκπαίδευση. Ο υπόλοιπος κώδικας είναι η καθαυτού διαδικασία της εκπαίδευσης. Όπως βλέπουμε σ’ αυτό το σημείο δηλώνεται και το πρότυπο της εφαρμογής. Το πρότυπο σ’ αυτή την περίπτωση είναι το CharLmHmmChunker. Ο κατασκευαστής αυτού, ο οποίος είναι ο: `CharLmHmmChunker(TokenizerFactory tokenizerFactory, AbstractHmmEstimator hmmEstimator)` δέχεται δυο ορίσματα: Την κλάση `TokenizerFactory` η οποία δημιουργεί δείγματα (token) από μια υπό-ακολουθία πινάκων χαρακτήρων. Και την κλάση `AbstractHmmEstimator` η οποία χρησιμοποιείται για την εκπαίδευση ενός μοντέλου Hidden Markov. Αυτά τα δύο ορίσματα ορίζονται με την σειρά τους

παραπάνω. Η κλάση GeneTagParser, που υπάρχει μετά, παρέχει έναν αναλυτή ετικετών για την εφαρμογή GeneTag.

3.2.3. Η Εφαρμογή 'Genia'

Σ' αυτό το σημείο πριν συνεχίσουμε με την ανάλυση της εφαρμογής, θα ήταν ίσως καλό να κάνουμε μια μικρή αναφορά στο σώμα Genia. Ο σχολιασμός σωμάτων αποτελεί αυτό το διάστημα ένα βασικό θέμα για όλους τους τομείς της Επεξεργασίας Φυσικής Γλώσσας (Natural Language Processing - NLP) και εξαγωγής πληροφοριών που υιοθετούν την εποπτευόμενη μάθηση. Με την μεγάλη αύξηση των αποτελεσμάτων στην μοριακή βιολογία υπάρχει μια αυξανόμενη ανάγκη για την εξόρυξη των πληροφοριών για την εξαγωγή γνώσης από βάσεις δεδομένων και για την έξυπνη αναζήτηση των πληροφοριών στις online συλλογές περιοδικών. Το σώμα Genia, ένα σημασιολογικά σχολιασμένο σώμα της βιολογικής λογοτεχνίας, συντάσσεται και σχολιάζεται στο στόχο του προγράμματος Genia. Η έκδοση 3.0 του σώματος Genia αποτελείται από 2.000 περιλήψεις που λαμβάνονται από τη βάση δεδομένων MEDLINE, και περιέχει περισσότερες από 400.000 λέξεις και σχεδόν 100.000 σχολιασμούς που έχουν κωδικοποιηθεί με το χέρι για τους βιολογικούς όρους. Οι βασικές περιλήψεις έχουν επιλεγεί από τα αποτελέσματα της αναζήτησης με λέξεις κλειδιά (όρους MeSH) όπως 'Human', 'Blood Cells', και 'Transcription Factors'. (Kim, Ohta, Tateisi & Tsujii, 2003)

Στο σώμα Genia, τα άρθρα κωδικοποιούνται σε ένα σχέδιο που βασίζεται στην XML όπου κάθε άρθρο περιέχει την ταυτότητα MEDLINE του, τον τίτλο και την περίληψη, όπου όλα τα κείμενα των περιλήψεων αποτελούνται από προτάσεις, η μορφή αυτή απεικονίζεται στην Εικόνα 3.17.



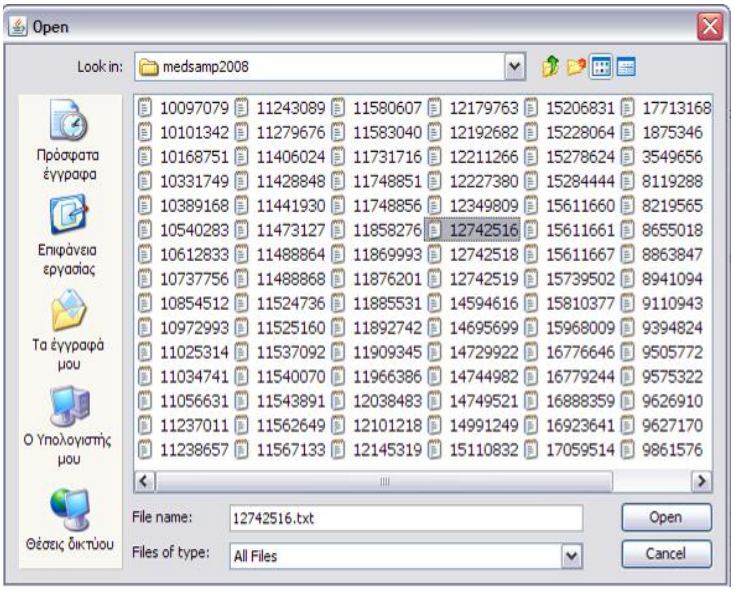
Εικόνα 3.17. Η κωδικοποίηση σε XML του σώματος Genia.

Το σώμα Genia αποτελεί ένα ουσιαστικό πρότυπο για την εξόρυξη κειμένου από μια βιοϊατρική συλλογή. Πολλές ομάδες έχουν χρησιμοποιήσει αυτό το σώμα για να εκπαιδεύσουν και να αξιολογήσουν τις επιδόσεις των συστημάτων τους. Αυτό το σώμα κατέχει μια από τις πλουσιότερες οντολογίες που έχουν χρησιμοποιηθεί για τον άμεσο σχολιασμό των κειμένων, επιπροσθέτως περιλαμβάνει περισσότερα δεδομένα. Για τους παραπάνω λόγους αποφασίστηκε να χρησιμοποιηθεί το σώμα Genia για την τοποθέτηση ετικετών στις οντότητες.

Για να συνεχίσουμε με την εφαρμογή, στην Εικόνα 3.11, αυτή τη φορά, επιλέγουμε το δεξί κουμπί με όνομα ‘Genia’, τότε θα ανοίξει το παράθυρο που φαίνεται στην Εικόνα 3.18.

Εικόνα 3.18. Το αρχικό παράθυρο της εφαρμογής Genia.

Όπως μπορεί να παρατηρήσει κανείς αυτό το παράθυρο είναι πανομοιότυπο με το αρχικό παράθυρο της εφαρμογής Gene Tag (Εικόνα 3.12). Στην φόρμα αυτή υπάρχουν ένα πεδίο κειμένου στο οποίο θα τοποθετηθεί η διεύθυνση του κειμένου, και τρία κουμπιά. Το πλήκτρο '<Back' μας πηγαίνει ένα παράθυρο πίσω, δηλαδή στο παράθυρο της Εικόνας 3.11. Το πλήκτρο 'Αναζήτηση' είναι αυτό που θα κάνει την εισαγωγή της διεύθυνσης του κειμένου πιο εύκολη και γρήγορη, ανοίγοντας ένα παράθυρο επιλογής αρχείου όπως αυτό στην Εικόνα 3.19. Και το τελευταίο πλήκτρο που βρίσκετε στο παράθυρο, το πλήκτρο με το όνομα 'Επεξεργασία', και είναι αυτή τη στιγμή ανενεργό, αλλά μόλις εισαχθεί το αρχείο θα γίνει ενεργό, όπως φαίνεται στην Εικόνα 3.20, αποτελεί το πλήκτρο όπου από πίσω του κρύβεται η διαδικασία της επεξεργασίας του κειμένου.

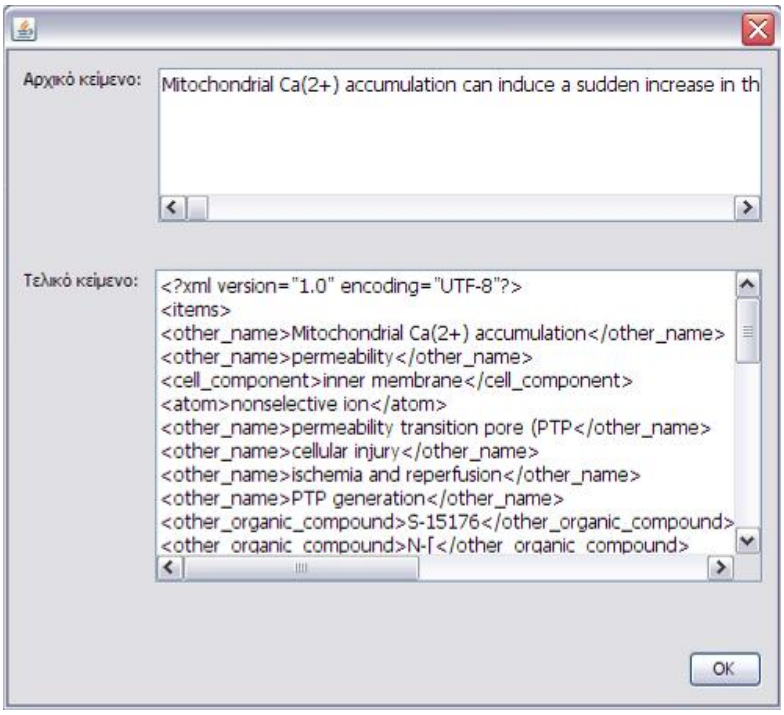


Εικόνα 3.19. Το παράθυρο επιλογής του αρχείου.

Εικόνα 3.20. Το παράθυρο της εφαρμογής μετά την εισαγωγή του αρχείου.

Εάν στο παραπάνω παράθυρο επιλεχθεί το πλήκτρο ‘Επεξεργασία’ τότε θα εκτελεστεί η διαδικασία του χαρακτηρισμού των οντοτήτων και τα αποτελέσματα θα εμφανιστούν στο παράθυρο που φαίνεται παρακάτω.

Στην πρώτη περιοχή κειμένου εμφανίζεται το κείμενο που δίνετε στην αρχή, και στην δεύτερη περιοχή του κειμένου εμφανίζονται τα αποτελέσματα της εφαρμογής.



Εικόνα 3.21. Τα αποτελέσματα για την εφαρμογή 'Genia' για το κείμενο με όνομα '12742516'.

Όπως μπορεί κανείς να παρατηρήσει τα αποτελέσματα που παράγουν οι δύο αυτές υπό-εφαρμογές είναι διαφορετικά. Συγκεκριμένα η εφαρμογή 'Gene Tag' αναγνωρίζει μέσα στο ίδιο κείμενα λιγότερες οντότητες από ότι η 'Genia'. Κάτι ακόμη που είναι φανερό επίσης είναι ότι οι οντότητες στην εφαρμογή 'Genia' είναι σαν να τοποθετούνται σε κατηγορίες, καθώς μπροστά από κάθε φράση που αναγνωρίζεται τοποθετείται το όνομα της κατηγορίας στην οποία αναγνωρίστηκε ότι ανήκει. Για παράδειγμα η πρώτη φράση που δίνει η 'Gene Tag'σαν αποτέλεσμα είναι η `<GENE>Mitochondrial Ca(2+) accumulation</GENE>`, ενώ η 'Genia' επιστρέφει ως πρώτη αναγνωρισμένη οντότητα την `<other_name>Mitochondrial Ca(2+) accumulation</other_name>`.

Ένα κομμάτι του κώδικα που εκτελεί την διαδικασία του χαρακτηρισμού των οντοτήτων για αυτή την εφαρμογή φαίνεται παρακάτω:

```
File modelFile = new File("models\" +
      "ne-en-bio-genia.TokenShapeChunker");
……
TokenizerFactory factory = new IndoEuropeanTokenizerFactory();
TokenCategorizer categorizer = new IndoEuropeanTokenCategorizer();

TrainTokenShapeChunker chunkerEstimator =
       new TrainTokenShapeChunker(categorizer, factory);
……
GeniaEntityChunkParser parser = new GeniaEntityChunkParser();
```

```
parser.setHandler(chunkerEstimator);
……
AbstractExternalizable.compileTo(chunkerEstimator, modelFile);

PrintStream out = new PrintStream(new FileOutputStream("genia.xml"));

Chunker chunker =
      (Chunker)AbstractExternalizable.readObject(modelFile);

Chunking chunking = chunker.chunk(doc);

Set<Chunk> allChunks = chunking.chunkSet();
out.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
out.println("<items>");
Iterator it = allChunks.iterator();

while (it.hasNext()) {
     String geniainfo = it.next().toString();
     String[] tokens = geniainfo.split("[-:@]");
     int start = new Integer(tokens[0]).intValue();
     int end = new Integer(tokens[1]).intValue();
     String tag = tokens[2];
     out.println("<"+tag+">" + doc.substring(start,end) +
           "</"+tag+">");
}
out.println("</items>");
out.close();
Scanner c = new Scanner(new FileInputStream("genia.xml"));
jTextArea2.setText("");
while (c.hasNextLine()) {
      jTextArea2.append(c.nextLine() + "\n");
}
```

Ας αναλύσουμε κάποια από τα σημαντικά τμήματα αυτού του κώδικα. Το αρχείο με όνομα `modelFile` προορίζεται για την εγγραφή του μοντέλου μετά τη διαδικασία της εκπαίδευσης. Η μέθοδος `TrainTokenShapeChunker` χρησιμοποιείται για την εκπαίδευση του μοντέλου. Η `GeniaEntityChunkParser` παρέχει έναν αναλυτή οντοτήτων για το σώμα οντοτήτων Genia, το οποίο είναι σε μορφή XML. Στην συνέχεια ρυθμίζεται ο χειριστής σε ένα ειδικό μπλοκ χειριστή. Ο `AbstractExternalizable` είναι ένας προσαρμοστής ο οποίος παρέχει ένα μηχανισμό για την εγγραφή ενός αντικειμένου σε ένα αντικείμενο εξόδου και το διαβάζει πάλι πίσω. Ο επόμενος κώδικας που ακολουθεί στην ουσία είναι αυτός που εγγράφει το αποτέλεσμα στην κατάλληλη περιοχή κειμένου και είναι πανομοιότυπος με τον κώδικα (για την συγκεκριμένη λειτουργία) της εφαρμογής GeneTag.

Κεφάλαιο 4

Συζήτηση-Συμπεράσματα

4.1 Συμπεράσματα

Ως κύριος στόχος της εργασίας είχε τεθεί η συγγραφή ενός προγράμματος το οποίο θα τροφοδοτείται με μια συλλογή βιοϊατρικών κειμένων, και θα προσφέρει διάφορα εργαλεία ανάκτησης ή εντοπισμού πληροφορίας από αυτά. Στην πραγματικότητα υλοποιήθηκαν δύο εφαρμογές: η πρώτη είχε ως στόχο την ανάκτηση πληροφοριών, και η δεύτερη τον εντοπισμό πληροφορίας σε βιοϊατρικά κείμενα.

Στην πρώτη εφαρμογή πραγματοποιήθηκε δεικτοδότηση των κειμένων με την βοήθεια του συστήματος Lucene και στην συνέχεια δημιουργήθηκε μια μηχανή αναζήτησης που αξιοποιεί το ευρετήριο. Η μηχανή αναζήτησης δέχεται ως είσοδο μια φράση που περιέχει τις λέξεις που ενδιαφέρουν το χρήστη, και οι οποίες θα πρέπει να περιέχονται στα κείμενα, που θα δοθούν ως έξοδος του προγράμματος. Ο στόχος αυτός επιτεύχθηκε και επιπλέον δόθηκε η δυνατότητα στο χρήστη, πέρα από το να βλέπει τα αποτελέσματα βαθμονομημένα ανάλογα με τη σχετικότητά τους ως προς την ερώτηση, να βλέπει και τα περιεχόμενα των κειμένων που επιλέγει μέσα από την εφαρμογή. Η λίστα με τα αποτελέσματα επιστρέφεται ταξινομημένη, με το κείμενο που έχει επιτύχει το μεγαλύτερο σκορ (περιέχει όλους τους όρους αναζήτησης) στην αρχή και τα υπόλοιπα να ακολουθούν.

Ένα σημαντικό θέμα που τίθεται όταν πραγματοποιείται μια εργασία είναι πόσο εύκολο είναι να στηθεί η εφαρμογή της εκάστοτε εργασίας. Για να μπορέσει κανείς να συμπεράνει αυτό θα πρέπει να συζητηθούν οι γνώσεις που απαιτούνται καθώς και η υπολογιστική ισχύς για την περάτωση της εφαρμογής. Για την συγκεκριμένη εφαρμογή απαιτείται η καλή γνώση της γλώσσας προγραμματισμού Java. Η γλώσσα Java είναι μια από τις πιο ευρέως διαδεδομένες γλώσσες προγραμματισμού στον κόσμο των προγραμματιστών. Όσο αναφορά το υπολογιστικό σύστημα που θα χρησιμοποιηθεί δεν υπάρχει κάποιος περιορισμός. Το Lucene μπορεί να εκτελεί την δημιουργία ευρετηρίων τάξεως άνω των 20MB/λεπτό σε ένα σύστημα με Pentium M 1.5GHz. Επίσης για την ίδια διαδικασία απαιτεί χαμηλή μνήμη RAM (μόνο ενός 1MB σωρό). Όμως η χρήση της μνήμης, για την συγκεκριμένη εφαρμογή,

συσχετίζεται άμεσα με το μέγεθος της συλλογή των κειμένων, καθώς η συλλογή βρίσκεται αποθηκευμένη τοπικά στον υπολογιστή.

Η μηχανή αναζήτησης που δημιουργήθηκε σ' αυτή την εργασία δεν είναι μια μηχανή αποκλειστικά για την ανάκτηση βιοϊατρικών πληροφοριών. Η μηχανή μπορεί και λειτουργεί εξίσου καλά για οποιαδήποτε πληροφορία. Μια ενδιαφέρουσα παραλλαγή θα ήταν να μπορεί να τρέξει η εφαρμογή αυτή και για ελληνικά κείμενα. Για να μπορέσουμε να πετύχουμε μια αλλαγή σαν αυτή τότε θα πρέπει να αλλάξουμε κυρίως το κομμάτι της δημιουργίας ευρετηρίων. Δηλαδή πρέπει να μελετηθεί η μέθοδος `IndexWriter` και τα ορίσματα της. Σίγουρα λοιπόν θα χρειαζόμασταν μια συλλογή ελληνικών κειμένων στην συγκεκριμένη περίπτωση. Επιπλέον θα έπρεπε να μετατρέψουμε τον 'αναλυτή' μας έτσι ώστε να είναι σε θέση να αναγνωρίζει τους ελληνικούς χαρακτήρες.

Ένα επακόλουθο της απλότητας της εφαρμογής είναι η εύκολη ανάπτυξη της και σύνδεση και με άλλες εφαρμογές. Για παράδειγμα θα μπορούσε να συνδεθεί η εφαρμογή αυτή με την δεύτερη εφαρμογή που αναπτύχθηκε από την εργασία αυτή. Μ' αυτό τον συνδυασμό ο χρήστης θα μπορούσε να ανακτά και να επεξεργάζεται πληροφορίες άμεσα και γρήγορα.

Βέβαια υπάρχουν και άλλες εξελίξεις που θα μπορούσαν να χρησιμοποιηθούν για την εργασία αυτή. Για παράδειγμα, κάτι που θα είχε, ίσως, ιδιαίτερο ενδιαφέρον ως μελλοντική συνέχιση του προγράμματος είναι δίπλα σε κάθε κείμενο που εμφανίζεται να δίνεται και το ανάλογο σκορ. Ένα από τα σημαντικότερα πλεονεκτήματα της εφαρμογής είναι ότι αξιοποιεί τις δυνατότητες του Lucene ώστε να επιστρέφει αποτελέσματα πολύ γρήγορα ακόμη και για μεγάλες συλλογές κειμένων.

Στη δεύτερη εφαρμογή μελετήθηκε το πρόβλημα του εντοπισμού οντοτήτων πληροφορίας σε κείμενα μετά από εκπαίδευση. Η εκπαίδευση είχε ως στόχο να δημιουργήσει δομικά και συντακτικά πρότυπα από κείμενα στα οποία έχουν προσημανθεί οντότητες και στη συνέχεια να αξιοποιήσει τα πρότυπα αυτά για τον εντοπισμό οντοτήτων σε νέες συλλογές βιοϊατρικών κειμένων. Εδώ έγινε χρήση του εργαλείου LingPipe το οποίο προσαρμόστηκε στις ανάγκες της εφαρμογής. Τα συμπεράσματα από τη χρήση του είναι ότι έχει πολλές δυνατότητες, απαιτεί όμως εκπαίδευση με κατάλληλα κείμενα για να έχει καλή απόδοση. Η καλή γνώση της

γλώσσας προγραμματισμού Java θεωρείται ως απαραίτητη για την πραγματοποίηση της εφαρμογής.

Η εφαρμογή αυτή θα μπορούσε εύκολα να μετατραπεί για την επεξεργασία οντοτήτων σε απλά αγγλικά κείμενα (δηλαδή για παράδειγμα να εντοπίζει διευθύνσεις ηλεκτρονικού ταχυδρομείου). Σε αυτή την περίπτωση θα χρειαζόταν να αλλάξουμε το πρότυπο της εφαρμογής. Ένα καλό πρότυπο που θα μπορούσαμε να χρησιμοποιήσουμε θα ήταν το `CharLmRescoringChunker`. Επομένως θα άλλαζε και η διαδικασία της εκπαίδευσης. Όμως η μετατροπή της εφαρμογής αυτής για την υποστήριξη των ελληνικών κείμενων είναι μια πολύπλοκη διαδικασία, και αυτό κυρίως γιατί θα χρειαζόταν να κατασκευαστούν τα αρχεία για την εκπαίδευση από το μηδέν.

Μελλοντική δουλειά στην κατεύθυνση της εκπαίδευσης μπορεί να περιλαμβάνει τη σχεδίαση και ανάπτυξη εργαλείων για την ανάκτηση γνώσης από συγκεκριμένες συλλογές βιοϊατρικών κειμένων, την όσο το δυνατόν καλύτερη και πληρέστερη σήμανσή τους και εμπλουτισμό τους με μεταδεδομένα και σε επόμενο στάδιο την προσθήκη της ανακτηθείσας πληροφορίας σε ιατρικές οντολογίες.

Μια ακόμη μελλοντική εργασία που θα μπορούσε να κατασκευαστεί πάνω στον τομέα του εντοπισμού οντοτήτων είναι η εύρεση των συν αναφορών μέσα στα βιοϊατρικά κείμενα. Ο σκοπός αυτής της εφαρμογής θα ήταν να αναζητά μέσα σ' ένα κείμενο και να εντοπίζει λέξεις που αναφέρονται σε ίδιες οντότητες, δηλαδή να αναζητά και να εντοπίζει λέξεις που πιθανώς είναι συνώνυμες.

Τέλος, ένα επιπλέον χαρακτηριστικό της εφαρμογής αυτής είναι ότι τα αποτελέσματα της είναι σε XML μορφή. Το κύριο χαρακτηριστικό της XML που την καθιστά ιδιαίτερα σημαντική είναι ότι παρέχει μια βασική σύνταξη με την οποία μπορεί να μοιραστεί πληροφορίες μεταξύ διαφορετικών υπολογιστών, διαφορετικών εφαρμογών, και διαφορετικών οργανισμών. Άρα και η εφαρμογή μας αποκτά αυτά τα χαρακτηριστικά, που την καθιστούν ως μια εφαρμογή που τα αποτελέσματα της μπορούν να αναγνωστούν από οποιαδήποτε υπολογιστικό σύστημα.

Ο τομέας της ανάκτησης πληροφορίας είναι σίγουρα ένας τομέας που έχει μεγάλο ερευνητικό ενδιαφέρον ειδικά για τις εφαρμογές βιοϊατρικής και ιατρικής, καθώς μεγάλος όγκος πληροφορίας έχει γραφτεί και γράφεται σε ηλεκτρονικά κείμενα χωρίς να είναι άμεσα διαθέσιμος σε μορφή που να μπορεί να αξιοποιηθεί από πληροφοριακά συστήματα, όπως είναι για παράδειγμα τα συστήματα γνώσης και στήριξης απόφασης. Η εργασία επέδειξε τις δυνατότητες συγκεκριμένων

προγραμμάτων που μάλιστα μπορούν να επεκταθούν και να προσαρμοστούν και υπέδειξε πιθανές εφαρμογές που μπορούν να αναπτυχθούν.

Βιβλιογραφία

1. Baeza-Yates R., & Ribeiro-Neto B. (1999), *"Modern Information Retrieval"*, AddisonWesley.
2. Clarke S., & Willett P. (1997). *"Estimating the recall performance of search engines"*. ASLIB Proceedings, 49 (7), pp. 184-189.
3. Cleverdon C.W., Mills J., & Keen M. (1966), *"Factors Determining the Performance of Indexing Systems"*, Cranfield, Vol. 1, Design, Vol.II, *Test Results*, ASLIB Cranfield Project.
4. Dunham M. (2004), *"Data Mining: Εισαγωγικά και Προηγμένα Θέματα Εξόρυξη Γνώσης Από Δεδομένα"*, Επιμέλεια Ελληνικής έκδοσης: Βασίλης Βερύκιος και Γιάννης Θεοδωρίδης, ©2004.
5. Gospodnetic O., & Hatcher E. (2005), *"Lucene in Action"*, ©2005 by Manning Publications Co.
6. Hand D., Mannila H., & Smyth P. (2001), *"Principles of Data Mining"*, MIT Press.
7. Kim J.D., Ohta T., Tateisi Y., & Tsujii J. (2003), *"GENIA corpus—a semantically annotated corpus for bio-textmining"*, Vol. 19 Suppl. 1 2003, pp. i180–i182, DOI: 10.1093/bioinformatics/btg1023.
8. Korfhage Robert R. (1997). *"Information Storage and Retrieval"*, Wiley.
9. Newman M.J. (2000), *"The structure of scientific collaboration networks"*, Proc.Nat.Acad.Sci.. No. 98, pp. 404-409.
10. Oppenheiem C., Moris A., Mcknight C., & Lowley S. (2000), *"The evaluation of WWW search engines"*, Journal of documentation, 56 (2), pp. 190-211.
11. Rijsbergen K. (1979), *"Information Retrieval"*, 2nd ed. London: Butterworths, chapter 1, Accessed at http://www.dcs.gla.ac.uk/Keith/Preface.html on 5/12/07
12. Singhal A. (2001), *"Modern Information Retrieval: A Brief Overview"*, Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, *Copyright 2001 IEEE.*

Δικτυακοί τόποι

13. http://alias-i.com/lingpipe/
14. http://ir.dcs.gla.ac.uk/terrier
15. http://sphinxsearch.com/

16. http://www.dataparksearch.org/
17. http://www.egothor.org/index.shtml
18. http://www.geocities.com/bioinformaticsweb/definition.html
19. http://www.htdig.org/main.html
20. http://www.indexdata.dk/zebra/
21. http://www.lemurproject.org/lemur/
22. http://www.nzdl.org/html/mg.html
23. http://www.seg.rmit.edu.au/zettair/
24. http://www.wumpus-search.org/
25. http://www.xapian.org/