ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εργαλείο Συλλογής και Οργάνωσης Γνώσης με Μηχανισμούς Μετα-Αναζήτησης στον Ιστό

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΑΡΓΥΡΗ ΚΟΛΛΙΑ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2009

Στους γονείς μου, στον Παναγιώτη που έφυγε νωρίς και στα παιδιά του

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εργαλείο Συλλογής και Οργάνωσης Γνώσης με Μηχανισμούς Μετα-Αναζήτησης στον Ιστό

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΑΡΓΥΡΗ ΚΟΛΛΙΑ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 30η Μαρτίου 2009.

.....	.....	.....
Σελλής Τιμολέων Καθηγητής Ε.Μ.Π.	Βασιλείου Ιωάννης Καθηγητής Ε.Μ.Π.	Στάμου Γεώργιος Λέκτορας Ε.Μ.Π.

Αθήνα, Μάρτιος 2009

.....

ΑΡΓΥΡΗΣ ΚΟΛΛΙΑΣ

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

«Η προστασία των έργων της διανόησης είναι υποχρέωση όλων μας.»

Περίληψη

Η συγκεκριμένη διπλωματική εργασία αφορά στην ανάπτυξη μίας εφαρμογής συλλογής και οργάνωσης γνώσης με μηχανισμούς μετα-αναζήτησης στον Ιστό, βασισμένη στο λογισμικό ανοικτού κώδικα *FreeMind* (http://freemind.sourceforge.net/) το οποίο ειδικεύεται στην χαρτογράφηση σκέψεων (*mind-mapping*). Κατά την δημιουργία ενός χάρτη σκέψεων, ιδεών ή / και εργασιών που συνδέονται μεταξύ τους (mind-map), ο χρήστης θα είναι πλέον σε θέση να αναζητά πληροφορίες, για ένα συσχετιζόμενο με τον γράφο θέμα, από τον παγκόσμιο ιστό, και να προσθέτει στοιχεία, εμπλουτίζοντας κατά αυτόν τον τρόπο το διάγραμμά του. Η αναζήτηση μπορεί να προσαρμοστεί στις ανάγκες του χρήστη και πιο συγκεκριμένα, μπορεί να γίνει επιλογή του επιθυμητού τύπου των αποτελεσμάτων (π.χ. ιστοσελίδες ή papers), ενώ ειδική μέριμνα λαμβάνεται για τις επιστημονικές δημοσιεύσεις, χάριν των οποίων έχει συσταθεί μία Βάση Δεδομένων – πιστό αντίγραφο εκείνης του *DBLP*, προκειμένου η τελική πληροφόρηση να είναι ακόμα πιο έγκυρη και πληρέστερη. Πρέπει να σημειωθεί ότι η εφαρμογή μας λαμβάνει αποτελέσματα από διάφορες δημοφιλείς μηχανές αναζήτησης, καθώς και από εξειδικευμένες μηχανές, ενώ στο τέλος παρουσιάζει τα αποτελέσματα συγχωνευμένα και ταξινομημένα σύμφωνα με τον «δημοκρατικό» αλγόριθμο *weighted Borda-Fuse*. Επιπροσθέτως, έχει ενσωματωθεί ένας Mozilla-based web-browser, ο οποίος πέρα από τις συνήθεις υπηρεσίες πλοήγησης στο Διαδίκτυο, αξιοποιεί την τεχνική του *screen-scraping*, προκειμένου σε συνδυασμό με ένα κατάλληλο .xml αρχείο (οι προδιαγραφές του οποίου αναφέρονται σε ένα λιτό .dtd αρχείο), να είναι σε θέση ακόμα και ο αρχάριος χρήστης να επεκτείνει το σύστημα, προσθέτοντας κι άλλες μηχανές αναζήτησης. Τέλος, η διαδικασία του *wrapping* μίας ιστοσελίδας αποτελεσμάτων της εκάστοτε νεοεγκαθιστάμενης μηχανής αναζήτησης συνοψίζεται στην πρόγνωση και σύνθεση του *full search URL*, την επισήμανση από μέρους του χρήστη των δομικών τμημάτων ενός αποτελέσματος (τίτλος – διαδικτυακός σύνδεσμος – περίληψη / περιγραφή) και την μηχανική εκμάθηση ανάγνωσης (βάσει της προηγούμενης υπόδειξης) των τριών μερών καθενός αποτελέσματος.

Λέξεις Κλειδιά: χαρτογράφηση σκέψεων, μετα-αναζήτηση, εξόρυξη δεδομένων, απόξεση οθόνης, πρόγνωση συνδέσμου μηχανής αναζήτησης, αναζήτηση επιστημονικών δημοσιεύσεων, αναζήτηση εγγραφών στο DBLP

Abstract

The scope of this diploma thesis is the development of a knowledge collection and organization application, equipped with an advanced web meta-searching mechanism. This thesis is based upon the open source tool *FreeMind* (http://freemind.sourceforge.net/), which specializes in *mind-mapping*. During the creation of a map of thoughts, ideas or / and tasks, linked all together (mindmap), the user will now be able to search for information (as far as a topic of this graph is concerned) in the World Wide Web, and furthermore add elements, enriching in this way the diagram. The searching process can be adapted to the user's needs, and particularly, a choice of the desired result data type (for example webpages or papers) can be made, while special concern is paid for scientific publications, in favor of which a Data Base – exact copy of the one *DBLP* uses has been acquainted, in order for the final information to be even more approved and fuller. It must also be recorded, that our system receives results from various popular search engines, even from ad hoc ones, while in the end the results are being presented merged and sorted, according to the "democratic" algorithm *weighted Borda-Fuse*. What is more, a Mozilla-based web-browser has been integrated, which beyond usual navigation services in the Internet, it also takes advantage of the *screen-scraping* technique, in order to allow even beginner-level users to expand the system by adding more search engines. This is being accomplished by combining an appropriate .xml file, which specifications are described in a frugal .dtd file. Finally, the *wrapping process* of a result webpage, which origins from a newly being installed search engine, is summarized in the prediction and composition of the *full search URL*, the result structural segments labeling on behalf of the user (title – web link – summary / description) and the machine learning of the reading (based on the former indication) regarding the three parts of every one result.

Keywords: mind-mapping, meta-searching, data mining, screen-scraping, search URL prediction, paper searching, DBLP record searching

Ευχαριστίες

Θα ήθελα να ευχαριστήσω όλους τους ανθρώπους, οι οποίοι με άμεσο ή έμμεσο τρόπο συνέβαλαν στην εκπόνηση αυτής της διπλωματικής εργασίας.
Ευχαριστώ θερμά τον επιβλέποντα Καθηγητή κ. Τιμολέοντα Σελλή, ο οποίος, τόσο με την εμβέλεια της προσωπικότητάς του όσο και με την έκταση της επιστημονικής του κατάρτισης, αποτέλεσε ισχυρό κίνητρο για εμένα, προκειμένου να ενασχοληθώ με την περιοχή των Βάσεων Δεδομένων και του Διαδικτύου ως γνωστικό αντικείμενο.
Επίσης, επιθυμώ να εκφράσω τις ιδιαίτερες ευχαριστίες μου στον συνεπιβλέποντα ερευνητή κ. Θεόδωρο Δαλαμάγκα, ο οποίος, μέσω της ακαδημαϊκής καθοδήγησης και της ακατάπαυστης ψυχολογικής ενθάρρυνσής του, αποτέλεσε θεμελιώδη πυλώνα για την επιτυχή έκβαση του όλου εγχειρήματος. Επιπλέον, σε καμία περίπτωση δεν θα μπορούσα να λησμονήσω τον υποψήφιο διδάκτορα του Ε.Μ.Π. κ. Γεώργιο Γιαννόπουλο, που με την αμέριστη συμπαράστασή του, συνετέλεσε στον υπερκερασμό προκυψασών δυσκολιών.
Τέλος, θέλω από τα βάθη της ψυχής μου να ευχαριστήσω τους γονείς και τα αδέρφια μου Ισίδωρο και Βαρβάρα, για την ουσιαστική υποστήριξη που μου παρείχαν όλα αυτά τα χρόνια των σπουδών μου.

Πίνακας περιεχομένων

1

Εισαγωγή

1.1 Εργαλεία υποστήριξης της δημιουργικότητας

(Creativity support tools)

Οι μηχανές αναζήτησης, όπως το Google, είναι το βασικό εργαλείο αναζήτησης πληροφορίας στο Web. Η δημοφιλία στη χρήση τους οφείλεται σε δύο παράγοντες:

1. Απλή μορφή γλώσσας ερώτησης (keyword-based search): ο χρήστης δεν χρειάζεται να γνωρίζει κάποια γλώσσα ερωτήσεων (π.χ. SQL) με σύνταξη και σημασιολογία για να διατυπώνει την ερώτησή του. Απλά και μόνο πληκτρολογεί ένα σύνολο από λέξεις-κλειδιά (keywords) που θεωρεί ότι περιγράφουν καλύτερα το θέμα προς αναζήτηση. Στη συνέχεια, η μηχανή επιστρέφει ιστοσελίδες με περιεχόμενο σχετικό ως προς αυτό το θέμα. Μάλιστα οι ιστοσελίδες ταξινομούνται (relevance ranking) ως προς το βαθμό ομοιότητάς (similarity value) τους με τις λέξεις-κλειδιά.
2. Ώριμη τεχνολογία αναζήτησης κειμένου (text information retrieval): οι τεχνολογίες αναζήτησης κειμένων με περιεχόμενο σχετικό ως προς κάποιες λέξεις-κλειδιά έχουν ήδη συμπληρώσει πάνω από 25 χρόνια ζωής[1]. Η προσθήκη μηχανισμών που εκμεταλλεύονται την ύπαρξη συνδέσμων μεταξύ κειμένων στον Ιστό για να επιβεβαιώσουν την ομοιότητά τους και τη σχέση τους, και να αναπροσαρμόσουν την

[1] http://www.cs.mu.oz.au/mg/

ταξινόμηση των αποτελεσμάτων (το γνωστό PageRank[2] του Google), έχει βελτιώσει σημαντικά την ποιότητα των αποτελεσμάτων της αναζήτησης.

Το απλό μοντέλο ερώτησης είναι σημαντικό πλεονέκτημα, τουλάχιστον για αναζητήσεις σε θέμα καλά ορισμένο εκ των προτέρων. Αν για παράδειγμα θέλετε να βρείτε reviews για συσκευές mp3, τότε πληκτρολογώντας απλά τις λέξεις-κλειδιά "reviews mp3 player" όλα τα πρώτα αποτελέσματα θα ικανοποιούν πλήρως τις ανάγκες σας.

Το πρόβλημα…

Συχνά όμως οι ανάγκες αναζήτησης πληροφορίας είναι πιο σύνθετες. Σκεφτείτε ένα μεταπτυχιακό φοιτητή που ψάχνει πληροφορίες για τις τρέχουσες τεχνολογίες, τις δημοσιεύσεις, τις ερευνητικές ομάδες, κλπ για μια ερευνητική θεματική περιοχή. Ο φοιτητής έχει στο μυαλό του μια αφαιρετική περιγραφή, δηλαδή κάποιες έννοιες ή θέματα (θα τα λέμε απλά έννοιες – concepts – από εδώ και στο εξής) που περιγράφουν το γενικότερο πεδίο γνώσης που θέλει να εξερευνήσει και να αναζητήσει πληροφορία. Για κάθε μια τέτοια έννοια, μπορεί να αναζητά διαφορετικά πράγματα: π.χ. κείμενα σχετικά με την έννοια Α, blogs για την έννοια Β, κλπ. Επίσης, οι έννοιες μπορεί να σχετίζονται μεταξύ τους.

Για παράδειγμα, ο φοιτητής γνωρίζει ότι τον ενδιαφέρει το θέμα *κατευθυνόμενοι γράφοι* σε σχέση με το θέμα *βάσεις δεδομένων*. Η πληροφορία που αναζητά είναι δημοσιεύσεις σε συνέδρια (papers). Εκτός από αυτό, τον ενδιαφέρουν οι αλγόριθμοι *αποτίμησης ερωτήσεων σε γράφους*, και πιο συγκεκριμένα οι ερωτήσεις τύπου reachability που απαντούν αν δύο κόμβοι είναι στο ίδιο μονοπάτι. Γνωρίζει ότι οι αλγόριθμοι αυτοί είναι δύο κατηγοριών: αυτοί που χρησιμοποιούν κάποια μορφή αριθμητικής κωδικοποίησης (labelling scheme) για τους κόμβους και αυτοί που δεν τη χρησιμοποιούν. Και για τις δύο περιοχές θα ήθελε να βρει δημοσιεύσεις σε επιστημονικά περιοδικά και συνέδρια.

Οι αφαιρετικές περιγραφές του πεδίου γνώσης είναι δημοφιλές χαρακτηριστικό των εργαλείων διαχείρισης της σκέψης (mind manager tool). Τέτοια εργαλεία είναι γνωστά στο χώρο της εκπαιδευτικής κοινότητας (δείτε σχετικά http://en.wikipedia.org/wiki/Mind_map). Χρησιμοποιούν συνήθως διαγράμματα αναπαράστασης ιδεών και συσχετίσεων μεταξύ τους, ώστε να βοηθήσουν τον εκπαιδευόμενο να κατανοήσει τις βασικές ιδέες τις οποίες αργότερα θέλει να αναλύσει και να εξειδικεύσει. Τα διαγράμματα αυτά είναι σημαντικό βοήθημα για οργάνωση μελέτης, επίλυση προβλημάτων, επιλογή απόφασης, συγγραφή κειμένων, κλπ.

Το τρέχον μοντέλο λειτουργικότητας των μηχανών αναζήτησης αδυνατεί να ικανοποιήσει τις ανάγκες χρηστών όπως ο παραπάνω φοιτητής. Συγκεκριμένα:

[2] http://www.webworkshop.net/pagerank.html, http://infolab.stanford.edu/~backrub/google.html

1. Το μοντέλο ερώτησης δεν μπορεί να εκφράσει τις παραπάνω ανάγκες σε μια ενιαία διαδικασία αναζήτησης.
2. Η αφαιρετική περιγραφή του πεδίου γνώσης παρέχει επίσης πληροφορία με την οποία μπορεί κάποιος να συντονίσει την αναζήτηση, κατευθύνοντας τις ερωτήσεις σε συγκεκριμένες μηχανές (π.χ. Technorati για blogs, Google Scholar για δημοσιεύσεις), πάλι έχοντας ως στόχο τη βελτίωση των αποτελεσμάτων αναζήτησης. Και πάλι ο μηχανισμός αποτίμησης ερωτήσεων στο τρέχον μοντέλο λειτουργικότητας των μηχανών αναζήτησης αδυνατεί να πραγματοποιήσει το συντονισμό αυτό.

Ο στόχος…

Η διπλωματική εργασία θα αναπτύξει μια εφαρμογή συλλογής και οργάνωσης γνώσης με μηχανισμούς μετα-αναζήτησης στον Ιστό. Η εφαρμογή θα χρησιμοποιεί αφαιρετικές περιγραφές και συσχετίσεις των εννοιών προς αναζήτηση. Το μοντέλο αυτό έχει κοινά σημεία με αυτό των εργαλείων διαχείρισης σκέψης (mindmaps).

Στόχος της εφαρμογής είναι να έχει το ρόλο ενός εργαλείου υποστήριξης της δημιουργικότητας (creativity support tool). Κυρίαρχη πρόκληση όσον αφορά τα εργαλεία αυτά, είναι η προώθηση της δημιουργικότητας των χρηστών, με απώτερο σκοπό την επίσπευση των ενδεχόμενων νέων ανακαλύψεων και καινοτομιών [Shn07]. Έτσι, το βασικό ερώτημα που τίθεται είναι το πώς μπορούν οι σχεδιαστές των διαφόρων διαδραστικών εφαρμογών (και των αντιστοίχων περιβαλλόντων με τις σχετικές διαπροσωπείες), να επιτρέψουν στους ανθρώπους να είναι πιο δημιουργικοί και μάλιστα πιο συχνά. Καθώς οι πρωτοπορίες των επιστημόνων και οι πρωτοτυπίες των μηχανικών παράγουν οφέλη σε ευρεία κλίμακα, βελτιωμένα εργαλεία τα οποία προάγουν την ατομική, ομαδική και συλλογική δημιουργικότητα αποτελούν σημαντικές συνεισφορές. Η παρούσα και οι επόμενες γενιές προγραμματισμού, προσομοίωσης, αναπαράστασης της πληροφορίας και άλλες εφαρμογές εφοδιάζουν και εμπνέουν τους μηχανικούς και τους επιστήμονες, όπως η γραφική με υπολογιστές τους παραγωγούς ταινιών και τα εργαλεία σύνθεσης ήχων τους μουσικούς. Μέσω της αξιοποίησης τέτοιου είδους εργαλείων, και σε συνδυασμό πάντα με άλλα ήδη υπάρχοντα, οι αρχάριοι χρήστες μπορούν να δρουν ως έμπειροι, ενώ οι προχωρημένοι δύνανται να έχουν εκπληκτικά αποτελέσματα. Αν και τα τηλεσκόπια ή τα μικροσκόπια είναι ισχυρές συσκευές, οι οποίες επιτρέπουν την πραγματοποίηση ανακαλύψεων και καινοτομιών, δεν παύουν να συνιστούν εργαλεία και μόνο, ενώ η πράξη της δημιουργίας υλοποιείται από το ίδιο το άτομο. Ο διαρκώς επιταχυνόμενος ρυθμός της ακαδημαϊκής έρευνας, της καινοτομίας της Μηχανικής και της σχεδίασης εμπορικών προϊόντων αποτυπώνονται στις επιστημονικές δημοσιεύσεις, τις πατέντες και τις καταναλωτικές αγορές, αναδεικνύοντας το τι ο συγκερασμός της προσπάθειας του ανθρώπου με τις δυνατότητες της μηχανής μπορούν να επιτύχουν.

Στην προαναφερθείσα κατεύθυνση κινείται και η συγκεκριμένη διπλωματική εργασία, προσθέτοντας ωφέλιμα στοιχεία διαδικτυακών υπηρεσιών σε ένα δημοφιλές εργαλείο ανάπτυξης χαρτών σκέψεων. Η μέθοδος της συστηματικής χαρτογράφησης σκέψεων – ιδεών – εννοιών (που καλείται *mindmapping*) ενισχύει την δημιουργικότητα των χρηστών, επιτρέποντας την οργάνωση των συνειρμών τους και την προαγωγή της φαντασίας τους. Οι υπάρχουσες εφαρμογές στον χώρο αυτό είναι πολλές, όμως καμία εξ αυτών δεν μεριμνά για την σύνδεση των παρεχόμενων πληροφοριών από το Διαδίκτυο με τους αντίστοιχους mindmaps. Το κενό αυτό φιλοδοξεί να καλύψει η παρούσα διπλωματική, προσφέροντας ως τελικό προϊόν ένα εργαλείο σύνθεσης και ανάλυσης χαρτών σκέψεων, το οποίο θα εκμεταλλεύεται όμως και την τεράστια δυναμική του Internet. Το κοινό στο οποίο απευθύνεται είναι ιδιαίτερα ευρύ, καθώς θεωρούμε ότι μπορεί να χρησιμοποιηθεί τόσο από έναν απλό χρήστη για θέματα της καθημερινότητάς του, όσο και από έναν εξειδικευμένο επιστήμονα, ο οποίος αποζητά λύσεις σε προβλήματα του αντικειμένου μελέτης του.

1.2 Το αντικείμενο της διπλωματικής εργασίας

Το αξιοποιηθέν εργαλείο κατασκευής mindmaps είναι το *FreeMind*, το οποίο συνιστά μία εφαρμογή ανοικτού κώδικα. Ως mindmap εννοούμε ένα διάγραμμα, το οποίο χρησιμοποιείται για την αναπαράσταση λέξεων, ιδεών, εργασιών ή άλλων στοιχείων, που συνδέονται και τοποθετούνται κυκλικά γύρω από μία κεντρική λέξη-κλειδί ή ιδέα, ενώ οι κύριες χρήσεις του είναι η παραγωγή, η οπτικοποίηση, η σύνθεση και η κατηγοριοποίηση ιδεών, ως βοήθημα για την μελέτη, την οργάνωση, την επίλυση προβλημάτων και την λήψη αποφάσεων. Για την δημιουργία ή την επέκταση τέτοιου είδους χαρτών, παρέχουμε στον χρήστη δυνατότητες αξιοποίησης του Internet, και πιο συγκεκριμένα επιτρέπουμε τα ακόλουθα:

- Πραγματοποίηση προηγμένων διαδικτυακών αναζητήσεων βάσει συγκεκριμένων συνόλων λέξεων-κλειδιών
    - Αναζήτηση ιστοσελίδων γενικού περιεχομένου (general web-pages)
    - Αναζήτηση επιστημονικών δημοσιεύσεων (papers)
- Ενσωμάτωση καινούριων μηχανών αναζήτησης από τον ίδιο τον χρήστη, μέσω της εισαγωγής κάποιων στοιχείων και της τεχνικής του screen-scraping
- Διασταύρωση των αποτελεσμάτων για papers με την έγκριτη βάση δεδομένων του DBLP και εμπλουτισμός των ευρεθέντων εγγραφών με περαιτέρω στοιχεία
- Διαλογή μεταξύ των επιστρεφόμενων αποτελεσμάτων και μεταφορά των επιλεγμένων στοιχείων στον χάρτη μας

- Πλοήγηση στο Διαδίκτυο με την βοήθεια ενσωματωμένου φυλλομετρητή ιστοσελίδων

Άξιο λόγου είναι το γεγονός ότι ο εν λόγω εσωτερικός browser αποτελεί έναν κλώνο του πασίγνωστου Mozilla Firefox. Επίσης σημειώνουμε, ότι η διαδικασία του searching πραγματοποιείται στηριζόμενη σε έναν μηχανισμό μετα-αναζήτησης, δηλαδή λαμβάνονται αποτελέσματα από διάφορες μηχανές και στην συνέχεια ταξινομούνται σε μία ενιαία λίστα, βάσει της προκαθορισμένης βαθμολογίας αξιολόγησης της εκάστοτε μηχανής αναζήτησης και της σημαντικότητας (η οποία προδηλώνεται από την κατάταξη) του κάθε αποτελέσματος. Επιπλέον, πρέπει να αναφερθεί πως η βάση δεδομένων DBLP του Deutsche Trier Universitaet αφορά επιστημονικές δημοσιεύσεις στον τομέα της Επιστήμης των Υπολογιστών και είναι μία από τις πιο διάσημες σε διεθνές επίπεδο. Εκτός αυτού, σημαντικό είναι το γεγονός, πως οι μηχανές στο σύστημά μας διακρίνονται σε δύο είδη:

1. API-embedded search engines, οι οποίες έχουν προεγκατασταθεί κατά την φάση της ανάπτυξης της εφαρμογής, και
2. Wrapped search engines, οι οποίες μπορούν να εισαχθούν ως πρόσθετες κατά την φάση της χρήσης. Τέλος, όσον αφορά το screen-scraping, πρόκειται για μία μέθοδο όπου ένα πρόγραμμα υπολογιστή εξάγει πληροφορία από την έξοδο επί της οθόνης ενός άλλου προγράμματος.

1.2.1 Συνεισφορά

Κεντρικός στόχος της παρούσας διπλωματικής εργασίας είναι η διασύνδεση της ανάπτυξης χαρτών τύπου mindmap με χρήσιμες σχετικές πληροφορίες από το Internet. Πιο συγκεκριμένα, αυτή αποσκοπεί στην καλύτερη οργάνωση και τον ωφέλιμο εμπλουτισμό των εκάστοτε μελετούμενων θεμάτων, στην παροχή υπηρεσιών στοχευμένης διαδικτυακής αναζήτησης (μέσω του αντίστοιχου ranking-sorting algorithm), καθώς και εξειδικευμένης αναζήτησης που επικεντρώνεται σε papers.

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Μελετήσαμε συστήματα χαρτογράφησης σκέψεων, τα αξιολογήσαμε και τα κατηγοριοποιήσαμε ανάλογα με την άδειά τους (open source / proprietary) και τον τρόπο εκτέλεσής τους (desktop / web – application).
2. Καταστήσαμε λειτουργικές, αξιοποιώντας τα παρεχόμενα APIs, τις εξής μηχανές αναζήτησης: α) Gigablast Web Search, β) Google Web Search, γ) Google Scholar Paper Search, δ) MS Live Web Search, ε) Technorati Blog Search, στ) Yahoo Web Search & ζ) YouTube Video Search.

3. Οικοδομήσαμε έναν πρωτότυπο και φιλόδοξο μηχανισμό εγκατάστασης πρόσθετων μηχανών αναζήτησης, ο οποίος εκμεταλλεύεται την τεχνική του screen-scraping για την μηχανική εκμάθηση ανάγνωσης των αποτελεσμάτων (που προέρχονται από την εκάστοτε νεοεισαγόμενη μηχανή) και των δομικών τμημάτων αυτών.
4. Επινοήσαμε μέθοδο πρόγνωσης του full search URL μίας καινούριας μηχανής αναζήτησης, η οποία βαίνει προς εγκατάσταση στο σύστημά μας.
5. Αξιοποιήσαμε τον Jericho HTML Parser για την διαχείριση των HTML δενδρικών δομών των ιστοσελίδων, που αξιοποιούνται στην wrapping process.
6. Κάνοντας χρήση του DTD προτύπου, ακολουθήσαμε συγκεκριμένο τρόπο γραφής σε / ανάγνωσης από XML αρχεία, κάθε ένα από τα οποία περιλαμβάνει όλες τις απαραίτητες πληροφορίες για την λειτουργία μίας νέας μηχανής αναζήτησης.
7. Υλοποιήσαμε τον «δημοκρατικό» αλγόριθμο βαθμολόγησης και ταξινόμησης αποτελεσμάτων, προερχόμενων από διαφορετικές πηγές (άλλης σημαντικότητας η κάθε μία), που πρώτοι εμπνεύστηκαν οι Borda – Fuse.
8. Κατασκευάσαμε μία βάση δεδομένων, η οποία συνιστά κλώνο εκείνης του DBLP.
9. Χρησιμοποιήσαμε το Apache Lucene, ως εργαλείο οργάνωσης, ευρετηριοποίησης και πραγματοποίησης αναζητήσεων για την προαναφερθείσα ΒΔ.
10. Ενσωματώσαμε τον φυλλομετρητή ιστοσελίδων WebRenderer.
11. Όλες οι παραπάνω λειτουργικότητες ενσωματώθηκαν σε μία δημοφιλή και ανοικτού κώδικα εφαρμογή, το FreeMind.

1.3 Οργάνωση του κειμένου

Στην συγκεκριμένη ενότητα περιγράφονται συνοπτικά τα κεφάλαια της παρούσας διπλωματικής εργασίας. Έτσι, στο Κεφάλαιο 1 βρίσκεται η εισαγωγή, στην οποία επιχειρείται να οριοτεθεί ο ενδιαφέρων επιστημονικός χώρος, να ενταχθεί στα πλαίσιά του η εφαρμογή μας, να δοθεί μία πρώτη προσέγγιση για τις προκλήσεις που καλούμαστε να αντιμετωπίσουμε, καθώς και να παρουσιαστούν αφαιρετικά οι προτεινόμενες λύσεις. Το Κεφάλαιο 2 πραγματεύεται τα σχετικά θεωρητικά θέματα, επικεντρώνοντας στην έννοια του mindmap και αναδεικνύοντας τα πιο διάσημα ήδη υλοποιηθέντα συστήματα wrapping. Στο Κεφάλαιο 3 το συνολικό σύστημα διαιρείται σε υποσυστήματα, ενώ για το κάθε ένα από αυτά τονίζονται οι απαιτήσεις που το διέπουν και οι εργασίες, τις οποίες είναι επιφορτισμένο να φέρει εις πέρας, ενώ επίσης δίνεται και το E-R διάγραμμα της υπάρχουσας βάσης δεδομένων. Στο Κεφάλαιο 4 αναλύεται η αρχιτεκτονική της εφαρμογής μας, παρατίθενται οι

λεπτομερείς περιγραφές των κλάσεων (καθότι πρόκειται για object-oriented application), συζητείται περαιτέρω η βάση δεδομένων, ενώ επιπροσθέτως γίνεται και αναφορά στην χρησιμοποιούμενη κωδικοποίηση αρχείων (στα πλαίσια της εγκατάστασης και χρήσης νέων μηχανών αναζήτησης). Το Κεφάλαιο 5 υπεισέρχεται ενδελεχώς σε πιο λεπτομερειακά και εξειδικευμένα θέματα υλοποίησης, τα οποία χρίζουν ιδιαίτερης προσοχής ή αποπνέουν αλγοριθμικό ενδιαφέρον, και επιπλέον αναφέρονται οι πλατφόρμες ανάπτυξης, τα χρησιμοποιηθέντα προγραμματιστικά εργαλεία, οι απαιτήσεις σε software – hardware και η διαδικασία εγκατάστασης της εφαρμογής μας σε έναν υπολογιστή. Στο Κεφάλαιο 6 παρουσιάζεται αναλυτικά ένα πλήρες σενάριο χρήσης του συστήματος, το οποίο περιλαμβάνει και τα αντίστοιχα screenshots, ενώ μπορεί να αξιοποιηθεί και ως βοήθημα ή εγχειρίδιο χρήσης. Το Κεφάλαιο 7 αποτελεί τον επίλογο του συγκεκριμένου εγγράφου, στον οποίο εξάγουμε τα συμπεράσματα που σχετίζονται με την συμβολή της διπλωματικής μας εργασίας στον αντίστοιχο χώρο, τις λύσεις που αυτή προσφέρει, τις απαιτήσεις που ικανοποιεί, καθώς και τις εναπομείνασες προς υλοποίηση εργασίες και τυχόν μελλοντικές επεκτάσεις. Τέλος, το Κεφάλαιο 8 διαδραματίζει τον ρόλο του παραρτήματος των βιβλιογραφικών αναφορών, παραπομπές στις οποίες απαντώνται καθ' όλη την έκταση του κειμένου.

2

Θεωρητικό Υπόβαθρο και Σχετικές Εργασίες

Το συγκεκριμένο κεφάλαιο αφορά στην ανάδειξη των τεχνικών μεθοδολογιών και των μοντέλων, επί των οποίων βασίστηκε η παρούσα διπλωματική εργασία, καθώς και στην συνοπτική παρουσίαση άλλων επιστημονικών έργων που σχετίζονται με το εν λόγω αντικείμενο ή με παραπλήσιες θεματικές περιοχές.

2.1 Mindmaps

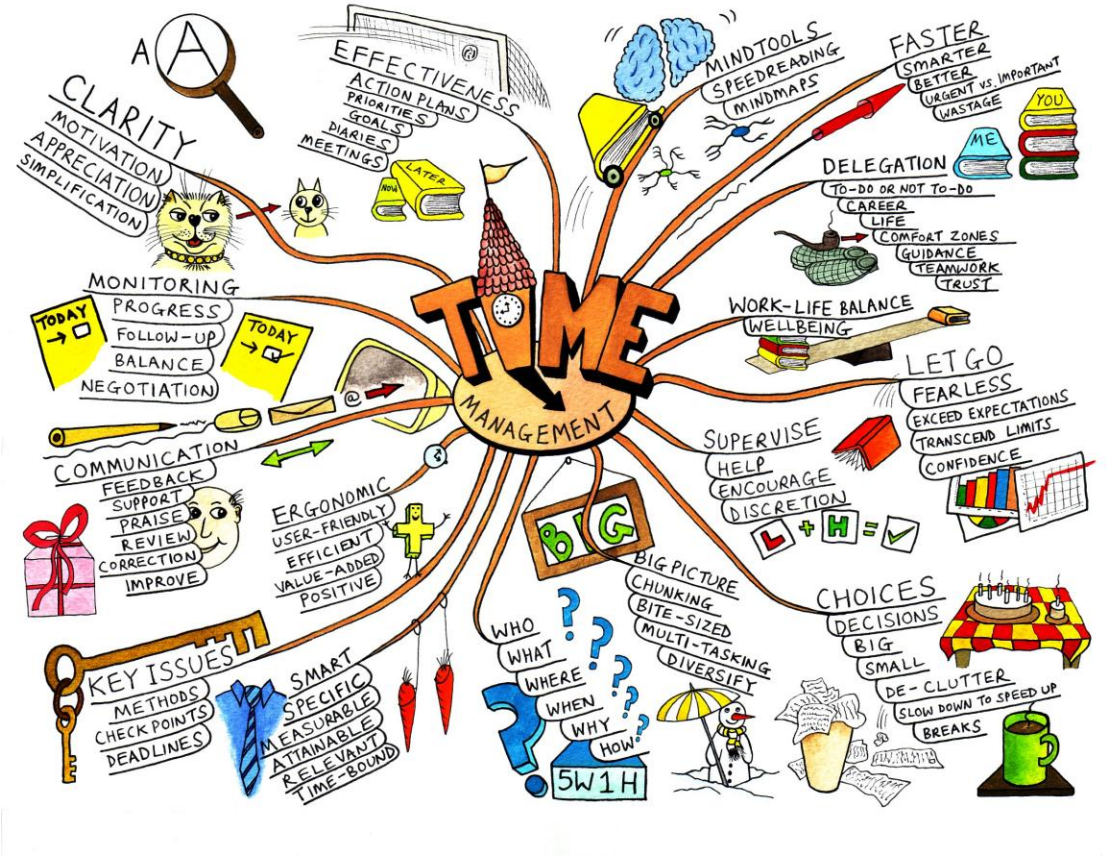
Βασική έννοια του μελετούμενου θεματικού τομέα είναι εκείνη του mindmap (χάρτης σκέψεων). Ετούτη η έννοια είναι συναφής με αυτές των concept map (χάρτης ιδεών-εννοιών) και ontology (οντολογία), ωστόσο παρουσιάζει συγκεκριμένες και ουσιαστικές διαφορές [Gol01]. Η σχετιζόμενη τεχνική του mind-mapping (χαρτογράφηση & διαχείριση σκέψεων) υποστηρίζεται και προωθείται από ορισμένα εργαλεία (όπως το FreeMind και το MindManager), κάθε ένα από τα οποία εμφανίζει μερικά θετικά και αρνητικά χαρακτηριστικά, γεγονός που μας επιτρέπει να τα συγκρίνουμε μεταξύ τους, να τα αξιολογούμε και να επιλέγουμε το εκάστοτε κατάλληλο.

2.1.1 Ορισμός

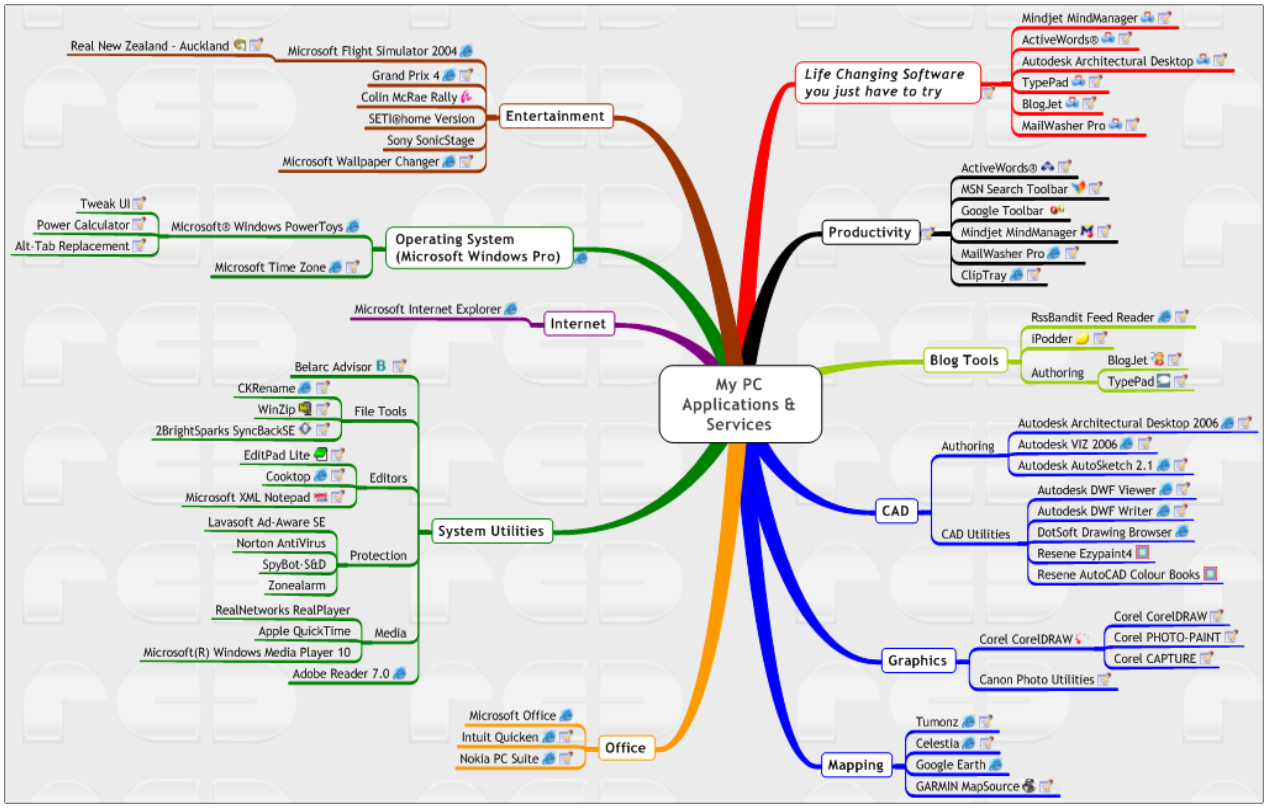
Ως *mindmap* (χάρτης σκέψεων) ορίζεται ένα διάγραμμα το οποίο χρησιμοποιείται για την αναπαράσταση εννοιών, ιδεών, εργασιών ή άλλου είδους στοιχείων, που περιστοιχίζουν και συνδέονται με μία κεντρική ιδέα ή έννοια-κλειδί [WP1].

Τα mindmaps χρησιμεύουν για την παραγωγή, οπτικοποίηση, δόμηση και κατηγοριοποίηση ιδεών ή σκέψεων και ως βοηθήματα στην μελέτη, την καταγραφή, την οργάνωση, την επίλυση προβλημάτων και την λήψη αποφάσεων, ακόμα και στην ανάκληση αναμνήσεων. Τα διάφορα στοιχεία ενός mindmap είναι διατεταγμένα με αυθόρμητο τρόπο σύμφωνα με το ενδιαφέρον ή την ιδιαίτερη σημασία των εμφανιζόμενων εννοιών και διαχωρίζονται σε ομάδες ή περιοχές. Αυτές διασυνδέονται μεταξύ τους με κλάδους, με απώτερο σκοπό την αναπαράσταση σημασιολογικών (ή άλλου τύπου) ζεύξεων ανάμεσα σε τμήματα πληροφορίας. Μέσω της καταγραφής σκέψεων με έναν τρόπο που θυμίζει εκείνο των γράφων, τα mindmaps ενθαρρύνουν μία ανορθόδοξη προσέγγιση στην τεχνητή πρόκληση εμπνεύσεων, που είναι σε θέση να παράγει νέες ιδέες και να αναδείξει πρωτόγνωρα μονοπάτια στο νου, δίχως να καταφεύγει σε ένα αυστηρό και τυπικό σύστημα ιεραρχικής οργάνωσης πληροφορίας. Αν και πιο ελεύθερα και εξατομικευμένα, τα mindmaps, είναι παρόμοια με τις πιο «δύσκαμπτες» δομές παράστασης των σημασιολογικών δικτύων (semantic networks) και των χαρτών αντίληψης (cognitive maps).

Παρατίθενται ακολούθως δύο mindmaps (ένα χειρόγραφο και ένα που έχει παραχθεί με την βοήθεια υπολογιστικού εργαλείου mind-mapping) ως παραδείγματα:



Σχήμα 2.1 – Παράδειγμα χειρόγραφου mindmap



Σχήμα 2.2 – Παράδειγμα mindmap με βοήθεια υπολογιστή

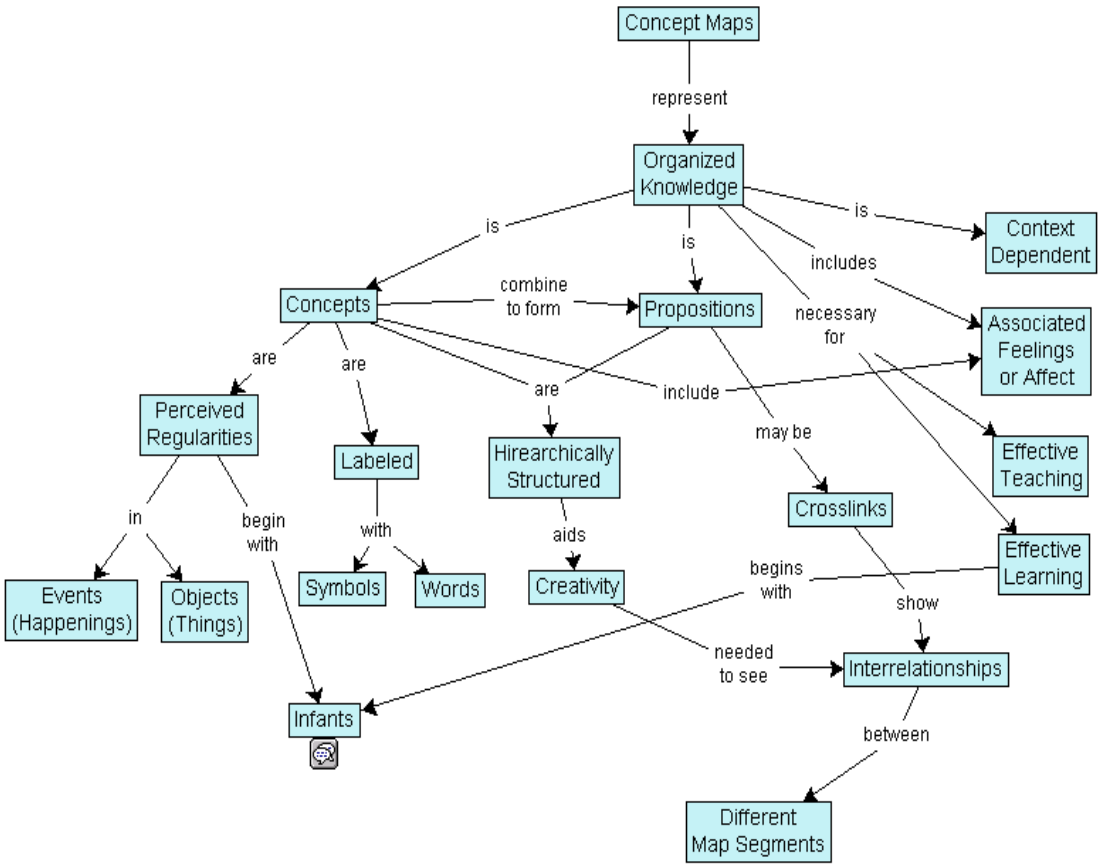
2.1.2 Διαφορές μεταξύ mind maps και concept maps

Ως *concept map* (χάρτης ιδεών-εννοιών) ορίζεται ένα διάγραμμα το οποίο δείχνει τις συσχετίσεις μεταξύ ιδεών ή εννοιών [WP2].

Τα concept maps συνιστούν γραφικά εργαλεία για την οργάνωση και αναπαράσταση γνώσης. Περιλαμβάνουν έννοιες, συνήθως εγκλεισμένες σε κύκλους ή τετράγωνα, και συνδέσεις μεταξύ των διαφόρων ιδεών, που υποδηλώνονται με ευθείες γραμμές. Οι έννοιες συνδέονται ενδεχομένως με βέλη, τα οποία φέρουν ετικέτες, δημιουργώντας μία ιεραρχική δομή σε μορφή δένδρου. Οι συσχετίσεις ανάμεσα στις διαφορετικές ιδέες διευκρινίζονται πιθανώς με συνεκτικές φράσεις (όπως «έχει ως αποτέλεσμα», «απαιτείται από», «συνεισφέρει σε» κ.ά.).

Ένα concept map μπορεί να αντιπαρατεθεί με την όμοια ιδέα ενός mind map. Η ειδοποιός διαφορά τους έγκειται στο γεγονός ότι ένα concept map βασίζεται στις ζεύξεις μεταξύ των διαφόρων καταγεγραμμένων εννοιών, ενώ ένα mind map στηρίζεται κυρίως σε ακτινωτές ιεραρχίες και δενδρικές δομές. Σημαντική είναι και η διαφορά στον τρόπο ανάπτυξης των δύο ειδών χαρτών. Έτσι, στο μεν concept map η ανάπτυξη γίνεται από πάνω προς τα κάτω, ενώ στο δε mind map η ανάπτυξη γίνεται γύρω από μία κεντρική έννοια κάθε φορά (είτε αυτή είναι η βασική είτε μία από τις περιφερειακές). Δευτερεύουσες διαφορές είναι η χρήση βελών και συνδετικών φράσεων στο concept map σε αντίθεση με το mind map.

Αμέσως παρακάτω δίνουμε ένα παράδειγμα concept map:



Σχήμα 2.3 – Παράδειγμα concept map

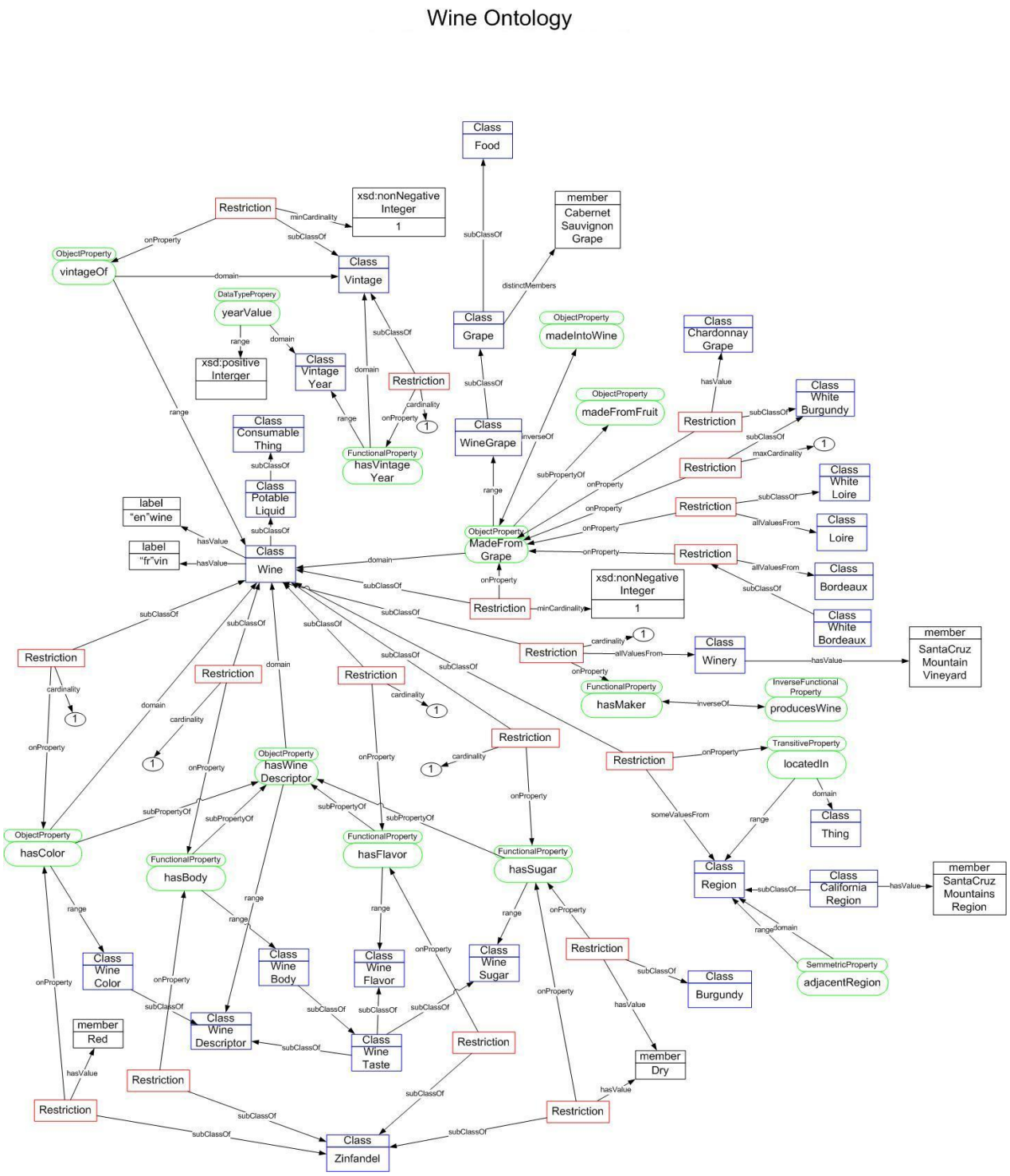
2.1.3 Διαφορές μεταξύ mind maps και ontologies

Ως *ontology* (οντολογία) ορίζεται μία τυπική αναπαράσταση ενός συνόλου εννοιών στα πλαίσια ενός domain (πεδίου), συμπεριλαμβανομένου των συσχετίσεων ανάμεσα στις έννοιες αυτές [WP3]. Οι οντολογίες έχουν κατασκευασθεί για να παρέχουν *σημασιολογία*. Κατά τον Tom Gruber, "*an ontology is a formal and explicit specification of a shared conceptualisation*".

Οι οντολογίες χρησιμεύουν στον προσδιορισμό των ιδιοτήτων ενός πεδίου και μπορούν να χρησιμοποιηθούν για τον έμμεσο καθορισμό του. Μία οντολογία παρέχει ένα λεξιλόγιο, το οποίο επαρκεί για την μοντελοποίηση ενός πεδίου, δηλαδή, για τον τύπο των αντικειμένων και / ή των εννοιών του, μαζί με τις σχετικές ιδιότητες και τις μεταξύ τους σχέσεις. Η χρήση των οντολογιών, ως ενός μέσου αναπαράστασης γνώσης, απαντάται κυρίως στους τομείς της Τεχνητής Νοημοσύνης, του Σημασιολογικού Ιστού, της Ανάπτυξης Λογισμικού, της Βιοϊατρικής Πληροφορικής και της Βιβλιοθηκονομίας.

Σε αντίθεση με ένα mindmap, μία ontology είναι πιο αυστηρά δομημένη και με σαφή κατεύθυνση προς τον προγραμματισμό. Έτσι κυρίαρχοι όροι είναι εκείνοι των class, subclass, property, subproperty, range, value, domain, member κλπ. Γίνεται άμεσα αντιληπτό ότι υπάρχει ένας περιορισμός όσον αφορά στην δημιουργικότητα και την ελευθερία, ο οποίος όμως συγκεράζεται με την ευχέρεια υλοποίησης και την σαφήνεια.

Ακολουθεί ένα χαρακτηριστικό παράδειγμα οντολογίας:



Σχήμα 2.4 – Παράδειγμα OWL ontology

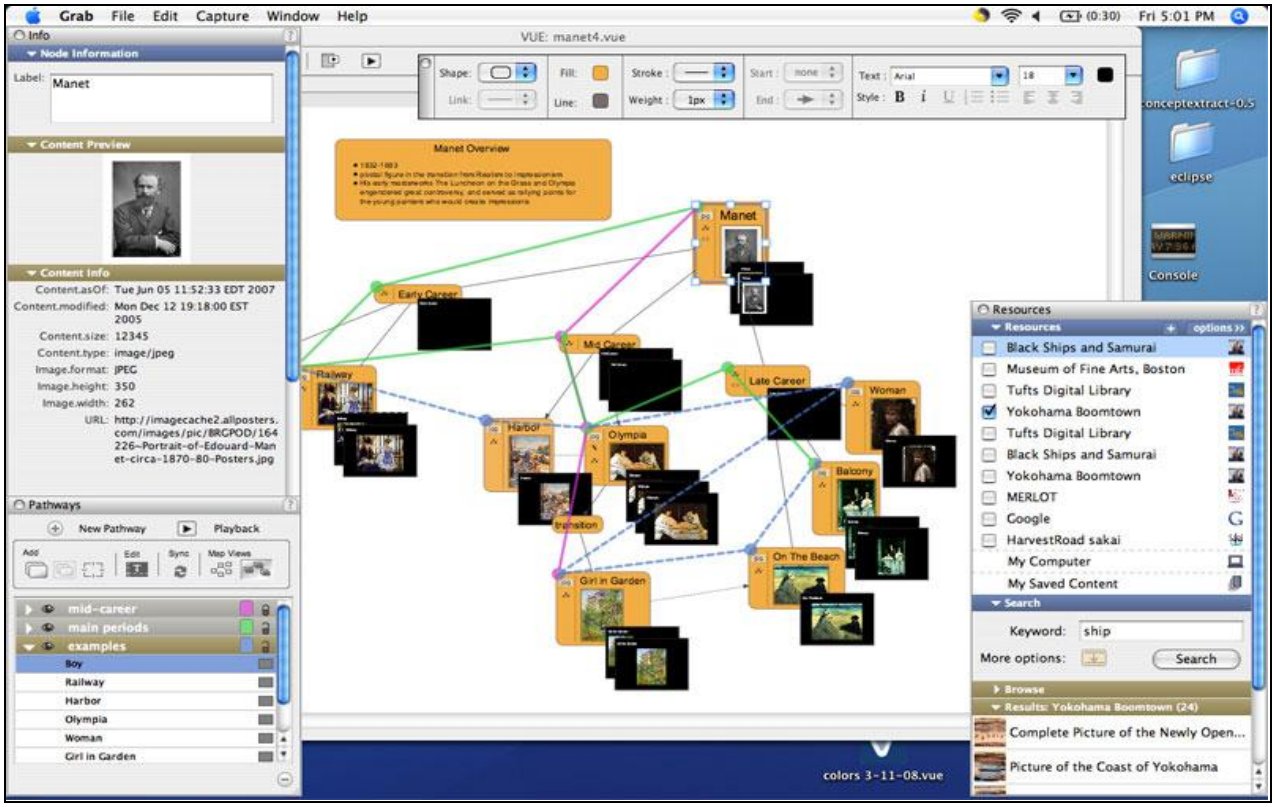
2.1.4 Εργαλεία mind-mapping, κατηγοριοποίησή τους και σχετική κριτική

Στην εποχή μας είναι διαθέσιμη μία πληθώρα εργαλείων που υλοποιούν την τεχνική του mind-mapping με την βοήθεια υπολογιστή. Τα εργαλεία αυτά, ως εφαρμογές λογισμικού, διακρίνονται σχετικά με το εάν είναι ελεύθερα (open source software ή freeware ή shareware) ή εάν κανείς πρέπει να καταβάλει κάποιο αντίτιμο προκειμένου να τα αποκτήσει, δίχως να έχει πρόσβαση στον κώδικά τους (proprietary software). Ένας άλλος διαχωρισμός αφορά στον τρόπο εκτέλεσής τους, δηλαδή εάν πρόκειται για desktop applications ή για online-web applications. Τέλος, κάποια από τα εργαλεία είναι ανεξάρτητες εφαρμογές (stand-alone applications) ή επεκτάσεις άλλων ήδη υπαρχόντων εφαρμογών (expansions). Με βάση τα προαναφερθέντα, προχωρούμε σε μία απαρίθμηση των πιο αξιομνημόνευτων εφαρμογών στην θεματική περιοχή του mind-mapping [WP4]:

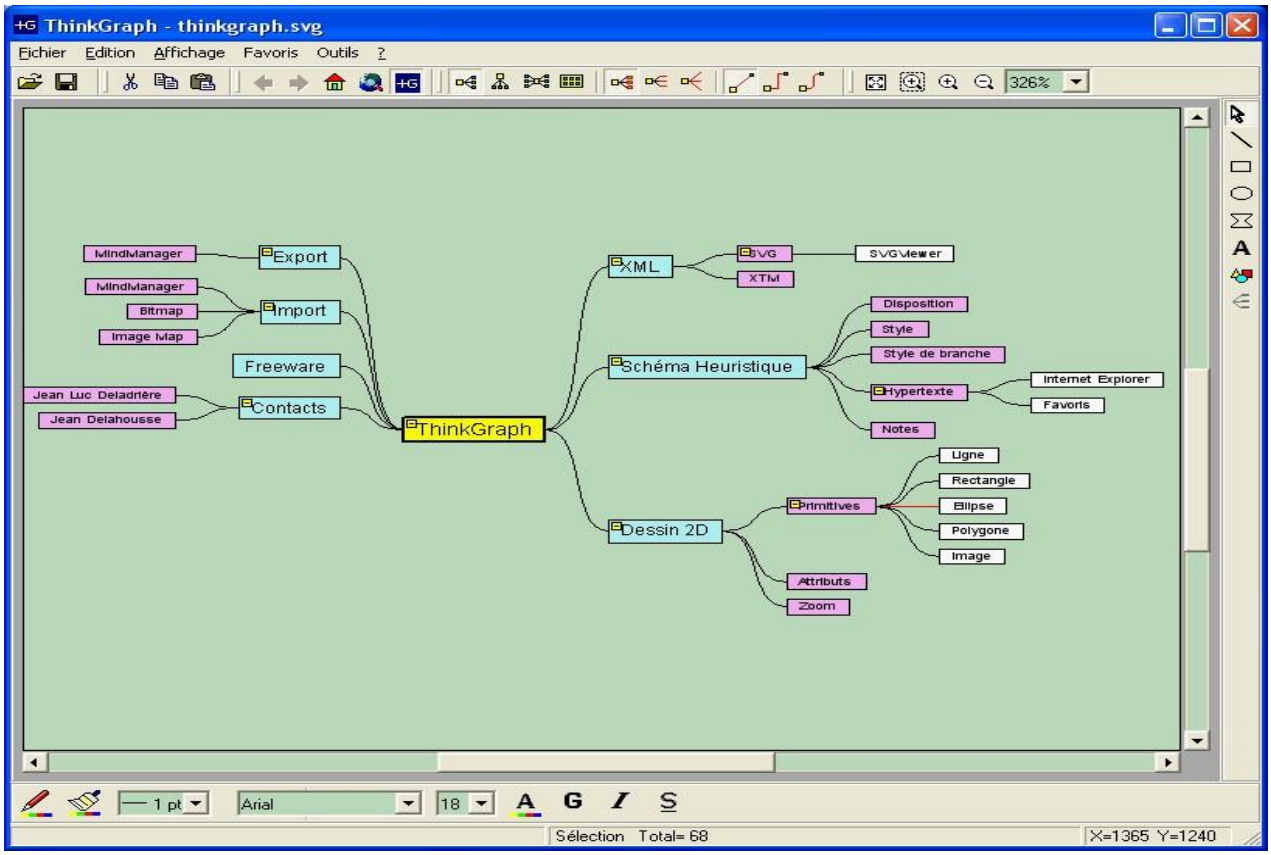
1η κατηγορία – Open source software / Freeware / Shareware:
FreeMind, Pimki, WikkaWiki, VUE, ThinkGraph
2η κατηγορία – Proprietary software:
MindManager, SmartDraw, Visual Mind, VisiMap, NovaMind, MindMapper, bCisive, HeadCase, Inspiration, OmniGraffle, Matchware OpenMind, Solution Language Tool, XMIND, SMART Ideas, Personal Brain, MindVisualizer, MindGenius, MindChart, MindApp, Mind Pad, ImindMap, Eminec MYMap, ConceptDraw MindMap, BrainMine, Aviz ThoughtMapper
3η κατηγορία – Web applications:
MindMeister (απαιτείται συνδρομή), Mindomo (δωρεάν), Mind42 (δωρεάν), Wisemapping (δωρεάν)
Πρόσθετα / Άλλα:
3D Topicscape (βοήθημα για το FreeMind και το MindManager), GyroQ (επέκταση για το MindManager), ResultsManager (επέκταση για το MindManager), EssentsialsPack (επέκταση για το MindManager), Map it!, IdeaTree

Από τις προηγούμενες εφαρμογές δημοφιλέστερες είναι οι FreeMind και MindManager. Διαδεδομένες είναι και οι SmartDraw, NovaMind, XMIND και ConceptDraw MindMap. Λόγω κυρίως του γεγονότος ότι από αυτά τα εργαλεία open source είναι μόνο το FreeMind, η παρούσα διπλωματική εργασία χρησιμοποίησε αυτό ως βάση προκειμένου να το επεκτείνει και να του προσδώσει πρόσθετες λειτουργικότητες.

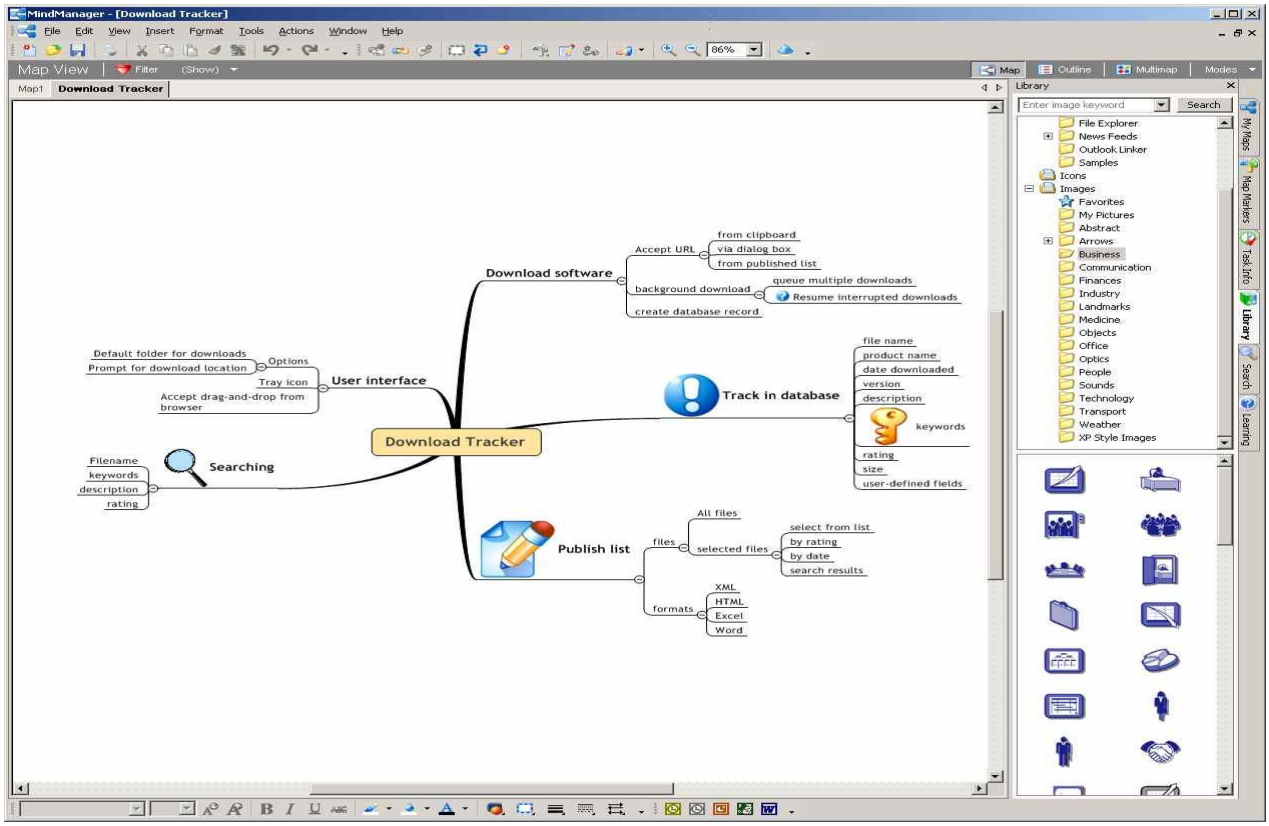
Στην συνέχεια παραθέτουμε σχετικά screen-shots από διάσημα εργαλεία για την κάθε μία προαναφερθείσα κατηγορία.



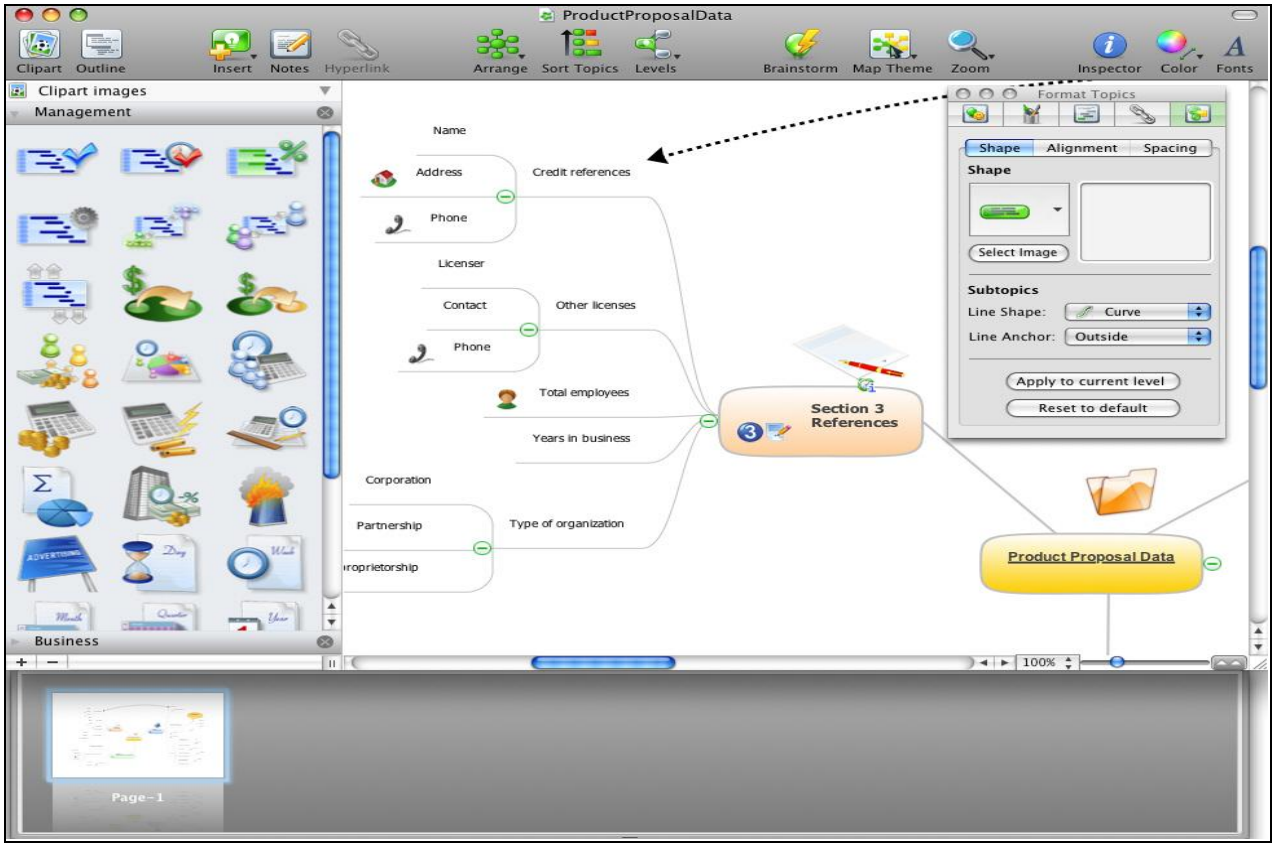
Στιγμιότυπο οθόνης 2.1 – Το περιβάλλον mind-mapping VUE (1η κατηγορία)



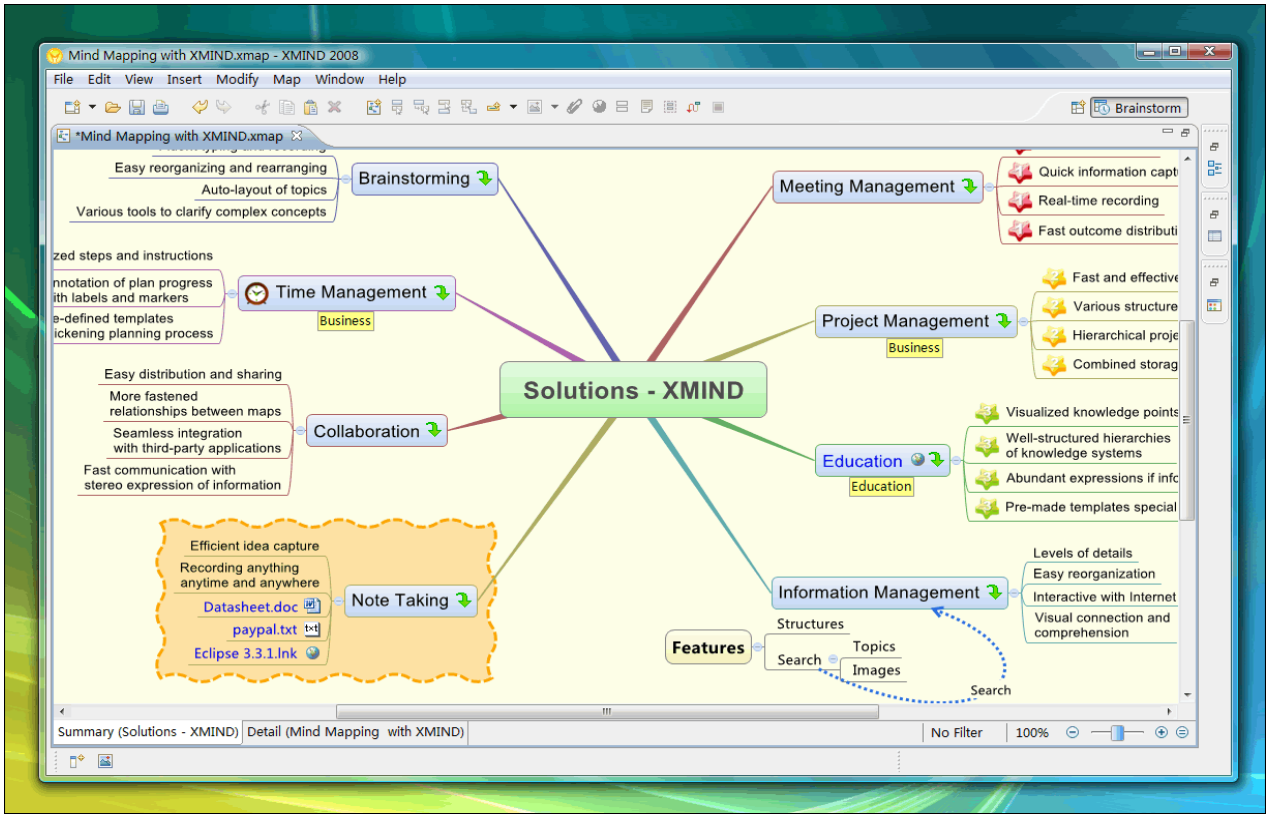
Στιγμιότυπο οθόνης 2.2 – Το εργαλείο ThinkGraph (1η κατηγορία)



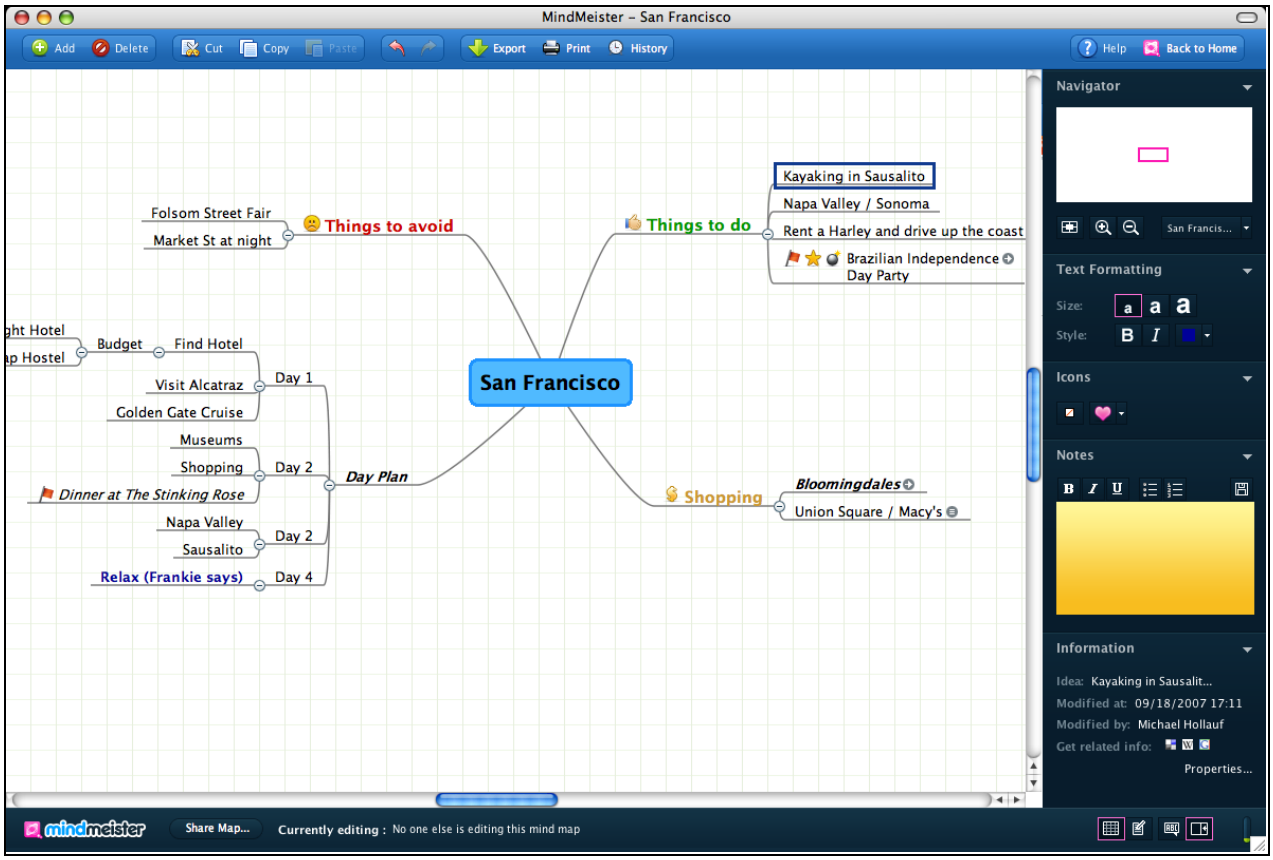
Στιγμιότυπο οθόνης 2.3 – Η δημοφιλής εφαρμογή MindManager (2η κατηγορία)



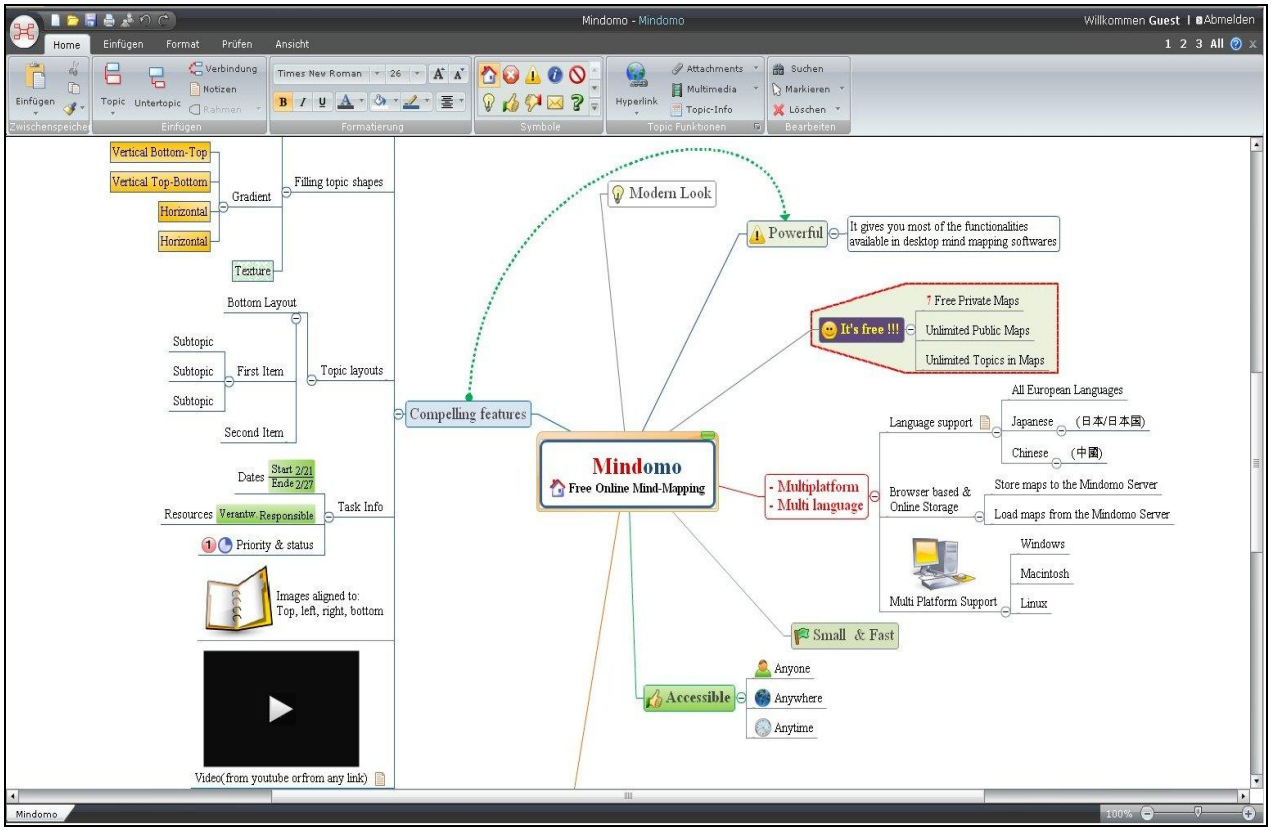
Στιγμιότυπο οθόνης 2.4 – Το περιβάλλον ConceptDraw MindMap (2η κατηγορία)



Στιγμιότυπο οθόνης 2.5 – Το mind-mapping εργαλείο XMIND (2η κατηγορία)



Στιγμιότυπο οθόνης 2.6 – Η web εφαρμογή MindMeister (3η κατηγορία)



Στιγμιότυπο οθόνης 2.7 – Το online εργαλείο Mindomo (3η κατηγορία)

2.2 Wrappers

Ένας *wrapper* είναι ένα εργαλείο που επιτρέπει σε κάποιο σύστημα την πρόσβαση στην πληροφορία που παρέχει ένα άλλο σύστημα και που δίχως αυτόν η διεπαφή μεταξύ των εν λόγω συστημάτων θα ήταν αδύνατη [CKG+06]. Πιο συγκεκριμένα, ο wrapper «περιβάλλει» την πηγή πληροφορίας, επιτελώντας τέσσερα βασικά έργα:

1) Ενημερώνεται από το σύστημα που χρειάζεται τα δεδομένα για το είδος, την μορφή και το πλήθος των δεδομένων (τις περισσότερες φορές ανακτώντας ένα μοτίβο δεδομένων – σύνολο κανόνων «εξόρυξης» πληροφοριών).
2) Εξάγει την πληροφορία από την πηγή και μάλιστα επιλεκτικά, απορρίπτοντας τα προκαθορισμένα ως «περιττά» δεδομένα, ακολουθώντας πιστά την εφαρμογή του δοθέντος μοτίβου (pattern matching process – διαδικασία ταιριάσματος μοτίβου).
3) Αναλαμβάνει την μεταφορά της πληροφορίας από την πηγή, προστατεύοντάς την από τυχόν αλλοίωση ή αλλαγή μορφής.

4) Παραδίδει τα δεδομένα στο σύστημα που τα ζήτησε, προκειμένου το τελευταίο να καταστεί ικανό να προβεί στις περαιτέρω ενέργειες οι οποίες απαιτούνται για την προώθηση ή ολοκλήρωση δικών του λειτουργιών.

Αξιοσημείωτο είναι στο σημείο αυτό, ότι κάνοντας χρήση της παραπάνω τεχνικής του wrapping αποφεύγεται η μεταβολή του κώδικα του συστήματος-δέκτη, η οποία θα ήταν αναγκαία σε διαφορετική περίπτωση. Επιπροσθέτως, η προσθήκη νέων πηγών δεδομένων ή η τροποποίηση των ήδη υπαρχουσών στο σύστημα-πηγή δεν επηρεάζει διόλου την wrapping process, κάτι που συμβάλλει στην διαχρονικά καλή λειτουργία της εν όλω διαδικασίας. Από τα προηγούμενα γίνεται άμεσα κατανοητό, ότι ένας wrapper λειτουργεί τελικά σαν «διαμεσολαβητής» μεταξύ δύο συστημάτων, όπου το ένα έχει ανάγκη πληροφορίες που διαθέτει το άλλο, κάνοντας δυνατή την απαιτούμενη μεταβίβαση δεδομένων. Πρέπει να τονιστεί ωστόσο, ότι η εξαγωγή των επιθυμητών πληροφοριών κάθε φορά από άλλη πηγή δεδομένων δεν είναι μία εύκολη υπόθεση, γιατί απαιτεί την όσο το δυνατόν καθολικότερη ικανότητα αντιμετώπισης διαφορετικών πηγών πληροφορίας. Επειδή εν προκειμένω ο λόγος γίνεται περί web servers (εξυπηρετητές Ιστού) όσον αφορά τα συστήματα-πηγές, κανείς οφείλει να αναλογισθεί το σύνολο των διαφόρων χρησιμοποιούμενων τεχνολογιών (σε επίπεδο programming και scripting) στο Διαδίκτυο. Το γεγονός αυτό, σε συνδυασμό με την επιτρεπόμενη ελαστικότητα κατά την παραγωγή σχετικού κώδικα από τα ορισθέντα πρότυπα (κυρίως εκείνων του W3C – World Wide Web Consortium), μας κάνουν να αντιληφθούμε γιατί παρουσιάζεται τόση δυσκολία στην εξαγωγή της επιθυμητής πληροφορίας. Επίσης, αναφέρουμε ότι αρκετές φορές συγχέεται ο όρος wrapper με εκείνον του extractor. Ένα wrapping σύστημα επιτελεί και την λειτουργία του data extraction, όχι μόνο αυτή όμως , όπως προαναφέρθηκε.

Βέβαια, η λειτουργία ενός wrapper ως εξαγωγέα πληροφορίας είναι η δυσκολότερη (σε σχέση με τις υπόλοιπες λειτουργίες), όμως η ουσιαστικότερη και παράλληλα η πιο ενδιαφέρουσα. Η μέθοδος του *web scraping* (διαδικτυακή απόξεση) υλοποιεί αυτή την λειτουργία της εξαγωγής και συνιστά ένα σύνολο κανόνων εξόρυξης περιεχομένου από έναν ιστότοπο, με σκοπό το συγκεκριμένο περιεχόμενο να γίνει διαθέσιμο προς χρήση σε μία άλλη εφαρμογή (desktop ή web application), στην παρούσα μορφή ή σε κάποια παραλλαγή της. Δύο ακόμα επικρατούσες ονομασίες για την εν λόγω μέθοδο είναι εκείνες του *screen scraping* (απόξεση οθόνης) και *web harvesting* (διαδικτυακή συγκομιδή), ενώ λιγότερο συχνά απαντώνται και οι όροι data scraping, HTML scraping και webpage scraping. Σημειώνουμε εδώ, ότι η λήψη της ενδιαφέρουσας πληροφορίας μπορεί να γίνει είτε από την δομή της πηγής (εφαρμογή web scraping επί του DOM μίας HTML σελίδας) είτε από την απεικόνιση των δεδομένων της πηγής στην οθόνη (εφαρμογή screen scraping επί του output κάποιου προγράμματος). Παρατηρούμε ότι οι έννοιες web scraping και screen scraping δεν

ταυτίζονται, όμως λόγω της ομοιότητάς τους και του κοινού τους αποτελέσματος (επιλεκτική λήψη και διάθεση πληροφορίας από κάποια πηγή), στην πράξη τις θεωρούμε ως ίδιες.

Μία από τις πιο διαδεδομένες εφαρμογές της τεχνικής του scraping είναι οι web crawlers (γνωστοί και ως web spiders, web robots ή web scutters), στους οποίους βασίζουν την λειτουργία τους οι μηχανές μετα-αναζήτησης πληροφοριών στο Internet. Οι μηχανισμοί αυτοί συνιστούν ένα automated script ή γενικότερα ένα πρόγραμμα, το οποίο ψάχνει τον Παγκόσμιο Ιστό με έναν μεθοδικό και αυτοματοποιημένο τρόπο. Πολλές ιστοσελίδες, ειδικά μηχανές αναζήτησης (όπως αναφέρθηκε πιο πάνω), χρησιμοποιούν το spidering ως τεχνική για την λήψη ενημερωμένων δεδομένων, η οποία αφορά στην αναζήτηση πληροφοριών στο Διαδίκτυο με έναν μεθοδικό και αυτοματοποιημένο τρόπο. Οι web crawlers αξιοποιούνται κυρίως στην δημιουργία αντιγράφων όλων των επισκεπτόμενων ιστοσελίδων, που χρησιμεύουν στην περαιτέρω επεξεργασία τους από μία μηχανή αναζήτησης, η οποία θα εφαρμόσει indexing επί των downloaded ιστοσελίδων, προκειμένου να είναι σε θέση να εκτελεί γρήγορες αναζητήσεις.

Πιο συγκεκριμένα τώρα, όσον αφορά την εξαγωγή πληροφορίας, αυτή μπορεί να επιχειρηθεί επί ελεύθερων – μη δομημένων, ημιδομημένων ή δομημένων εγγράφων. Παραθέτουμε αμέσως τους σχετικούς ορισμούς και τις απαραίτητες διασαφηνίσεις:

- Ελεύθερα έγγραφα είναι κυρίως εκείνα που περιλαμβάνουν φυσική γλώσσα.
- Ημιδομημένα έγγραφα είναι αυτά που ακολουθούν κάποιο πρότυπο, το οποίο όμως δεν είναι απόλυτο, όπως τα HTML αρχεία.
- Δομημένα έγγραφα είναι όσα κινούνται αυστηρά εντός του πλαισίου που επιβάλλει το πρότυπό τους, όπως τα XML και τα DTD αρχεία.

Στο σημείο αυτό εισάγουμε την έννοια του wrapper factory. Τα *Wrapper Factories* (συστήματα παραγωγής wrappers) αποτελούν εφαρμογές που αφορούν στην ανάπτυξη εργαλείων τα οποία υλοποιούν την μέθοδο του wrapping. Αξίζει τώρα να αναφερθεί η κατηγοριοποίηση των wrapper generating συστημάτων βάσει του βαθμού αυτοματοποίησής τους. Έτσι διακρίνουμε τις ακόλουθες τέσσερεις κατηγορίες [MMK99]:

1) Supervised wrapper factories: Σε ένα σύνολο σελίδων με δεδομένα ο χρήστης τονίζει με πρόσφορο τρόπο την ενδιαφέρουσα πληροφορία και το factory system αναλαμβάνει την «έκδοση» του κατάλληλου wrapper.
2) Semi-supervised wrapper factories: Σε ένα σύνολο σελίδων δεδομένων εξάγονται οι πληροφορίες, στη συνέχεια ο χρήστης καλείται να επιλέξει μοτίβο εξαγωγής επιθυμητής πληροφορίας και τελικά το factory παράγει τον αντίστοιχο wrapper.

3) Un-supervised wrapper factories: Σε ένα σύνολο σελίδων με δεδομένα το factory system αναλαμβάνει τον καθορισμό των πληροφοριών προς εξαγωγή, δίχως καμία ενέργεια από πλευράς του χρήστη, και συντάσσει τον ανάλογο wrapper.
4) Manual wrapper factories: Ο ίδιος ο χρήστης καλείται να αναπτύξει τον δικό του wrapper, αξιοποιώντας μία γλώσσα προγραμματισμού. Ακολούθως παρατίθενται τρία από τα σημαντικότερα σύγχρονα wrapper generating συστήματα και παρουσιάζεται συνοπτικά ο τρόπος λειτουργίας τους.

2.2.1 Το WysiWyg wrapping σύστημα W4F

Το σύστημα W4F (WysiWyg Web Wrapper Factory) συνιστά μία Java εργαλειοθήκη για την παραγωγή wrappers που αφορούν διαδικτυακές πηγές [SA99]. Το W4F αποτελείται από μία γλώσσα ανάκτησης που αποσκοπεί στην ταυτοποίηση των πηγών του Διαδικτύου, μία δηλωτική γλώσσα εξαγωγής για την έκφραση εύρωστων κανόνων εξόρυξης και από μία διεπαφή απεικόνισης που χρησιμεύει στην μεταβίβαση της εξηγμένης πληροφορίας σε ορισμένες από τον χρήστη δομές δεδομένων. Προκειμένου να βοηθηθεί ο χρήστης και να γίνει η δημιουργία wrappers γρήγορη και εύκολη, το εν λόγω σύστημα παρέχει υποστήριξη τύπου wysiwyg ("*what you see, is what you get*") μέσω wizards. Έτσι είναι δυνατή η ταχεία και ημιαυτόματη παραγωγή έτοιμων προς χρήση wrappers, οι οποίοι δίνονται ως Java κλάσεις. Το σύστημα W4F έχει χρησιμοποιηθεί επιτυχώς στην ανάπτυξη wrappers για συστήματα βάσεων δεδομένων και ως software agents, κάνοντας διαδικτυακό περιεχόμενο εύκολα διαθέσιμο σε εφαρμογές λογισμικού.

Πιο αναλυτικά, η διαδικασία παραγωγής wrappers συντελείται σε τρία βήματα:

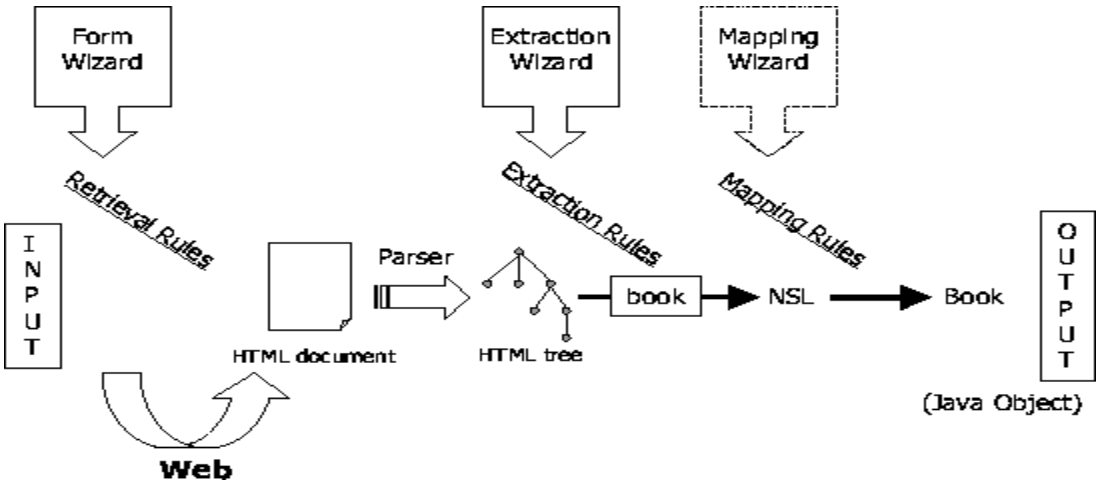
1) Το επίπεδο ανάκτησης της πληροφορίας,
2) Το επίπεδο εξαγωγής των δεδομένων,
3) Το επίπεδο απεικόνισης των ωφέλιμων πληροφοριών.

Στο επίπεδο ανάκτησης, ο wrapper λαμβάνει από την διαδικτυακή πηγή το ενδιαφέρον έγγραφο υπό την μορφή HTML κώδικα και στη συνέχεια το οδηγεί σε έναν HTML-cleaner αρχικά και ακολούθως σε έναν HTML-parser, προκειμένου να γίνει τελικά η αναπαράσταση του HTML εγγράφου σε δενδρική κατά DOM μορφή.

Στο επίπεδο εξαγωγής, εφαρμόζεται ένα σύνολο κανόνων εξόρυξης δεδομένων επί του δένδρου που κατασκευάστηκε προηγουμένως και οι εξηγμένες πληροφορίες αποθηκεύονται ως nested string list (NSL) δομές. Οι κανόνες εξαγωγής είναι εκφρασμένοι στην γλώσσα HEL (HTML extraction language), η οποία εκμεταλλεύεται την υπάρχουσα δενδρική μορφή και αποζητά σε αυτήν τα μονοπάτια που ταιριάζουν στο μονοπάτι-μοτίβο (pattern-path), αποβλέποντας στην εκμαίευση των επιθυμητών δεδομένων. Στην αμέσως προηγούμενη

διαδικασία αξιοποιούνται κανονικές εκφράσεις (regular expressions) για τον ακριβή προσδιορισμό των δεδομένων μέσω περιορισμών και μία τεχνική παράλληλου ελέγχου των τελικών τμημάτων των μονοπατιών, για τα μονοπάτια των οποίων τα αρχικά τμήματα έως κάποιο σημείο της διαδρομής ταυτίζονται.

Στο τελευταίο επίπεδο, το επίπεδο απεικόνισης, εφαρμόζεται ένα σύνολο κανόνων απεικόνισης για τα δεδομένα των NSL δομών. Ακολούθως δίνουμε μία σχηματική αναπαράσταση του συστήματος W4F:



Σχήμα 2.5 – Το σύστημα W4F

2.2.2 Το web-scraping σύστημα του MIT Piggy Bank

Το *Piggy Bank* είναι μία επέκταση (extension) του φυλλομετρητή Ιστού (web browser) Mozilla Firefox, το οποίο μετατρέπει τον εν λόγω browser σε μία mashup πλατφόρμα, επιτρέποντας την εξαγωγή δεδομένων από διαφορετικούς ιστοτόπους και την ανάμειξή τους [HMK07]. Επίσης, επιτρέπει στον χρήστη την αποθήκευση της εξαγόμενης πληροφορίας τοπικά, προκειμένου να είναι διαθέσιμη και αργότερα, είτε για μελέτη είτε για ανταλλαγή της συγκεντρωμένης πληροφορίας με άλλους χρήστες. Σημειώνουμε, ότι το Piggy Bank δημιουργήθηκε στα πλαίσια του *Simile Project* του MIT που αποσκοπεί στην γενικότερη οργάνωση των ψηφιακών δεδομένων.

Κύριος στόχος του Piggy Bank είναι να φέρει τον Σημασιολογικό Ιστό (Semantic Web) στον browser ενός απλού χρήστη. Η πρωτοβουλία για τον Σημασιολογικό Ιστό οραματίζεται ένα Διαδίκτυο, στο οποίο η πληροφορία είναι διαθέσιμη ανεξαρτήτως της μορφής αναπαράστασής της, καθιστώντας δυνατή μία αποτελεσματικότερη ανταλλαγή και μία ευκολότερη ανάμειξη μεταξύ ιστοτόπων και ιστοσελίδων [DOS03] (σημ.: Η μορφή δεδομένων που χαρακτηρίζεται από τα προηγούμενα και αφορά «αγνή», «καθαρή» πληροφορία είναι η RDF-Resource Description Framework). Κινούμενο σε αυτή την κατεύθυνση και βασιζόμενο στην ιδέα της πιο ευέλικτης πρόσβασης στην πληροφορία, το εν

λόγω σύστημα δίνει την δυνατότητα στους χρήστες να αξιοποιήσουν το περιεχομένου του Σημασιολογικού Ιστού εντός του διαδικτυακού περιεχομένου, καθώς οι χρήστες χρησιμοποιούν τον Παγκόσμιο Ιστό. Οπουδήποτε δεν είναι διαθέσιμο περιεχόμενο του Σημασιολογικού Ιστού, το Piggy Bank αφυπνίζει screen scrapers προκειμένου να ανοικοδομήσει κατάλληλα την πληροφορία εντός κοινών ιστοσελίδων, προσδίδοντάς τους σημασιολογικό περιεχόμενο. Μέσω της αξιοποίησης τεχνολογιών του Σημασιολογικού Ιστού, το συγκεκριμένο σύστημα παρέχει άμεσα οφέλη στην χρήση του υπάρχοντος Ιστού. Επιπλέον, η ύπαρξη έστω και μερικών Semantic Web – enabled sites και μερικών scrapers ήδη ωφελεί τους χρήστες. Έτσι, το Piggy Bank δείχνει έναν εύκολο και συνεχώς εξελισσόμενο δρόμο, δίχως να απαιτεί μία καθ' ολοκληρίαν υιοθέτηση της θεώρησης του Σημασιολογικού Ιστού.

Όσον αφορά την λειτουργία, το Piggy Bank δηλώνει όταν ο Firefox επισκέπτεται μία ιστοσελίδα, εάν το ίδιο μπορεί να εξάγει «καθαρή» πληροφορία από αυτήν ή όχι. Σε περίπτωση που είναι σε θέση να κάνει κάτι τέτοιο, προβαίνει στην εξόρυξη των στοιχείων της εν λόγω «καθαρής» πληροφορίας. Αυτές οι μονάδες πληροφορίας μπορούν να φιλτραριστούν περαιτέρω μέχρι να λάβουμε το επιθυμητό αποτέλεσμα και στην συνέχεια να αποθηκευθούν ή ακόμα και να σημειωθούν με την χρήση ετικετών (tags) για αποτελεσματικότερη ανάκτησή τους στο μέλλον. Εκτός των μονάδων πληροφορίας, μπορούν να σημειωθούν ακόμα και ολόκληρες ιστοσελίδες με λέξεις-κλειδιά. Αναφέρουμε, ότι ο μηχανισμός σημειώσεων είναι ενιαίος, με το Piggy Bank να μην κάνει διακρίσεις μεταξύ στοιχείων πληροφορίας ή ολόκληρων εγγράφων, χάριν ομοιογενοποίησης και ίδιας αντιμετώπισης. Οι χρήστες μπορούν να πραγματοποιούν αναζητήσεις στα αποθηκευμένα στοιχεία πληροφορίας με διάφορους τρόπους (π.χ. κατά τύπο, ή σύμφωνα με μία προκαθορισμένη υποομάδα, ή βάσει μιας λέξης-κλειδί). Τονίζουμε ότι ολόκληρη η τοπικά αποθηκευμένη συλλογή δεδομένων δεν διακρίνει τα στοιχεία που την απαρτίζουν ανάλογα με τον τύπο τους ή την πηγή προέλευσής τους. Αυτές οι συλλογές μονάδων πληροφορίας μπορούν να σταλούν σε σημασιολογικές τράπεζες δεδομένων (semantic banks) και να μοιράζονται μεταξύ χρηστών από όλο τον κόσμο.

Επίσης, πέρα από τους προεγκατεστημένους στο σύστημα screen scrapers για την εξόρυξη πληροφοριών, δύναται ο ίδιος ο χρήστης να προσθέσει κάποιον άλλο που έχει υπόψη του ή και τον δικό του, απλώς παρέχοντας τα μεταδεδομένα (metadata) του scraper και ενεργοποιώντας τον, προκειμένου να θεωρείται έμπιστος (trusted). Ένα άλλο extension του Firefox, το *Solvent*, είναι σχετικά απαραίτητο, καθώς βοηθά στην ανάπτυξη screen scrapers και στην ενσωμάτωσή τους στο Piggy Bank. Επειδή η παραγωγή screen scrapers μπορεί να αποβεί δύσκολη και απαιτητική σε επίπεδο λεπτομερειών, ένα καλό εργαλείο είναι απαραίτητο. Έτσι το Solvent, επιτρέπει:

- ✓ Την διαδραστική υπογράμμιση τμημάτων μίας σελίδας προς scraping, απευθείας στον web browser μας και την λήψη των κατάλληλων Xpaths για τα τμήματα αυτά.
- ✓ Την εξέταση του DOM των δεσμευμένων στοιχείων και την ανάθεση ονομάτων μεταβλητών σε αυτά.
- ✓ Την αυτόματη παραγωγή κώδικα Javascript, ο οποίος υλοποιεί τα πιο κοινά γνωρίσματα, όπως οι επαναλήψεις σε xpath αποτελέσματα.
- ✓ Την επιλογή μεταξύ διαφορετικών screen scraping templates, βάσει του τύπου της ιστοσελίδας προς scraping (ανεξάρτητη ιστοσελίδα, πολλαπλή ιστοσελίδα κλπ).
- ✓ Την επεξεργασία και την εκτέλεση του κώδικα του scraper κατευθείαν στον browser, καθιστώντας την φάση ανάπτυξης γρήγορη και αποδοτική.
- ✓ Την διάθεση των εξορυγμένων αποτελεσμάτων απευθείας στο Piggy Bank (ακόμα κι αν δεν προηγηθεί εγκατάσταση του scraper).
- ✓ Την αποθήκευση και δημοσίευση του scraper με τα απαραίτητα metadata, έτσι ώστε άλλοι να μπορούν να τον βρουν και να τον χρησιμοποιήσουν.
- ✓ Την παροχή της απαραίτητης βοήθειας σχετικά με Javascript, Xpath, DOM, RDF και την προβολή τοποθεσιών όπου είναι διαθέσιμα RDF λεξικά.

Επιπροσθέτως, απαριθμούμε τις διαφορετικές μεθόδους με τις οποίες το Piggy Bank συλλέγει την πληροφορία:

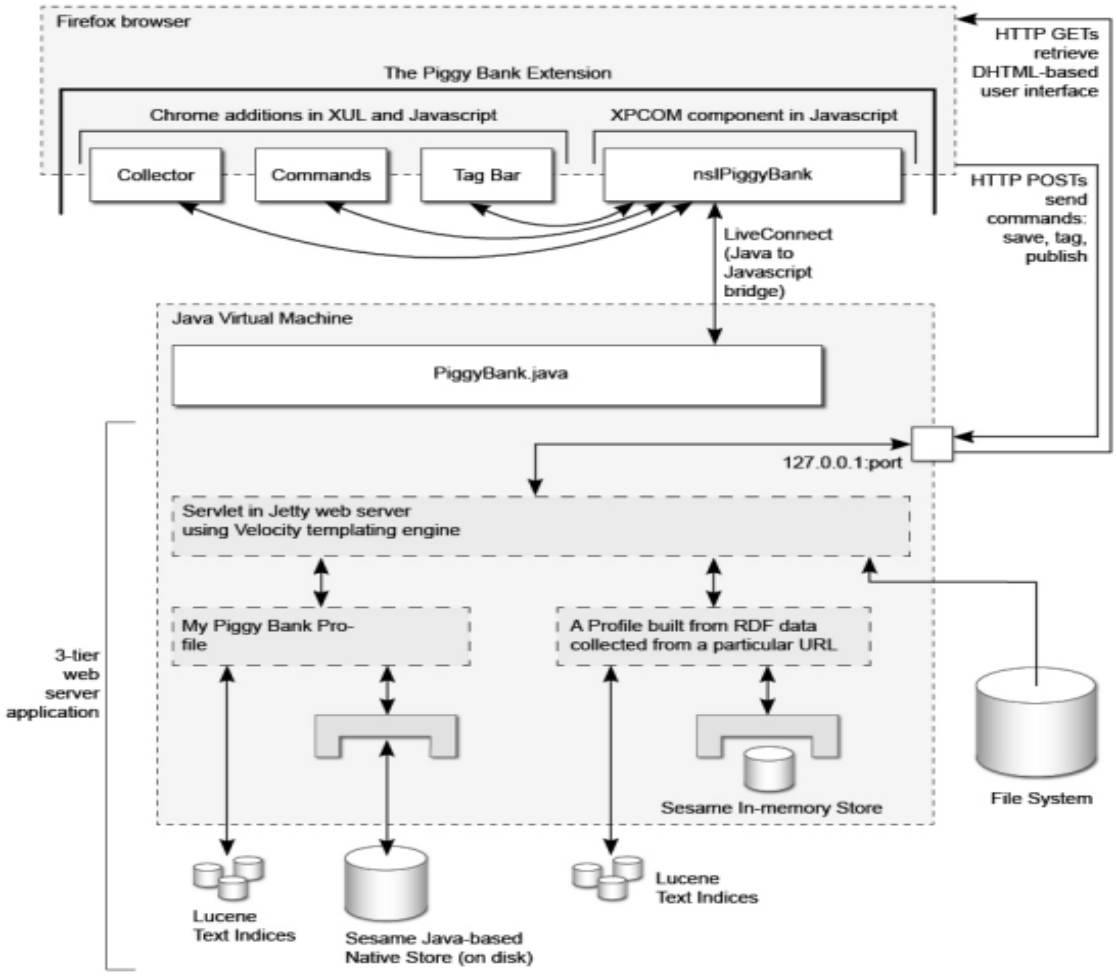
- ✓ Μέσω της ανάκτησης συνδέσμων από την υπό επεξεργασία ιστοσελίδα προς αντίστοιχες διαδικτυακές πηγές σε RDF/XML ή RSS μορφή και της μετατροπής των αντικειμένων-στόχων τους σε RDF μορφή.
- ✓ Με διαθέσιμους και εφαρμόσιμους XSL μετασχηματισμούς επί του DOM της υπό επεξεργασία ιστοσελίδας.
- ✓ Μέσω διαθέσιμου και εφαρμόσιμου Javascript κώδικα επί του DOM της υπό επεξεργασία ιστοσελίδας, ανακτώντας κι άλλες όμοιες ιστοσελίδες για επεξεργασία εφόσον κάτι τέτοιο είναι απαραίτητο.

Για την ευκολότερη κατανόηση της λειτουργίας του Piggy Bank σε συνεργασία με το Solvent, θα αναφέρουμε συνοπτικά ένα σενάριο χρήσης. Έστω λοιπόν ότι επιθυμούμε την εύρεση των τοποθεσιών όλων των σημείων πώλησης καφέ Starbucks στο Cambridge της Μασαχουσέτης των ΗΠΑ. Η διαδικασία θα αξιοποιήσει μέσω του Solvent την τεχνική του screen-scraping, για την ανεύρεση των ζητούμενων διευθύνσεων, και μετά θα εκμεταλλευθεί το Piggy Bank, προκειμένου να προβάλλει τα ευρεθέντα δεδομένα επί ενός χάρτη της περιοχής. Αρχικά ανοίγουμε τον Mozilla Firefox (στον οποίο έχουμε εγκαταστήσει ήδη τις δύο συζητούμενες επεκτάσεις), επισκεπτόμαστε την ιστοσελίδα των Starbucks και μεταβαίνουμε στον τομέα όπου βρίσκεται η λίστα με τα υπάρχοντα υποκαταστήματα. Αφού

ενεργοποιήσουμε το μενού του Solvent, επιλέγουμε με το ποντίκι του υπολογιστή μας μία από τις διαθέσιμες διευθύνσεις, ενώ αμέσως παρατηρούμε ότι λόγω της λειτουργίας του συγκεκριμένου extension, ο browser μας διακρίνει και τα υπόλοιπα όμοια στοιχεία. Ακολούθως, δηλώνουμε τα δομικά τμήματα που συνθέτουν ένα τέτοιο στοιχείο (URI – title – address) και προκαλούμε την δημιουργία του αντίστοιχου wrapper σε Javascript. Έπειτα, εκτελούμε τον παραχθέντα wrapper με είσοδο την ιστοσελίδα αυτή και παίρνουμε ως έξοδο στο Piggy Bank τα σχετικά ωφέλιμα δεδομένα. Στην συνέχεια, η εν λόγω δεύτερη επέκταση του φυλλομετρητή ιστοσελίδων επιφορτίζεται με το μαρκάρισμα των σημείων των Starbucks Cafes σε κατάλληλο χάρτη (έχοντας την βοήθεια της υπηρεσίας Google Maps). Εύκολα κατανοεί κανείς την σημασία της αυτοματοποιημένης εξόρυξης δεδομένων για την καθημερινότητα του απλού ατόμου, πόσο μάλλον για επιστημονικούς ή ερευνητικούς σκοπούς. Σημειώνουμε επιπροσθέτως, ότι το προαναφερθέν χαρακτηριστικό σενάριο μπορεί να βρεθεί υπό την μορφή εκτελέσιμου flash animation στην ακόλουθη ιστοσελίδα:

http://simile.mit.edu/solvent/screencasts/solvent_screencast.swf

Τέλος, δίνουμε την αρχιτεκτονική του συστήματος Piggy Bank και ένα screenshot από την διαδικασία του scraping με χρήση του Solvent:



Σχήμα 2.6 – Η αρχιτεκτονική του Piggy Bank

Σχήμα 2.7 – Screen scraping με το Solvent

2.2.3 Το οπτικό & διαδραστικό wrapping σύστημα Lixto

Το *Lixto* είναι ένα πλήρως οπτικό και διαδραστικό εργαλείο παραγωγής wrappers, το οποίο επιτρέπει στον χρήστη να δημιουργήσει, με μία οπτική μέθοδο, μία εφαρμογή για την εξαγωγή πληροφορίας σχετικής με ένα πρότυπο [BFG01]. Η δημιουργία αυτή μπορεί να γίνει βάσει μίας ενδεχομένως μεταβαλλόμενης ιστοσελίδας ή από ένα σύνολο ομοίως δομημένων ιστοσελίδων, ενώ επίσης το σύστημα είναι σε θέση να κατασκευάζει webpage XML companions. Στο υπόβαθρο, αόρατη στον χρήστη, εργάζεται η δηλωτική, λογική γλώσσα προγραμματισμού Elog, ενώ υπάρχει client και servlet έκδοση του Lixto.

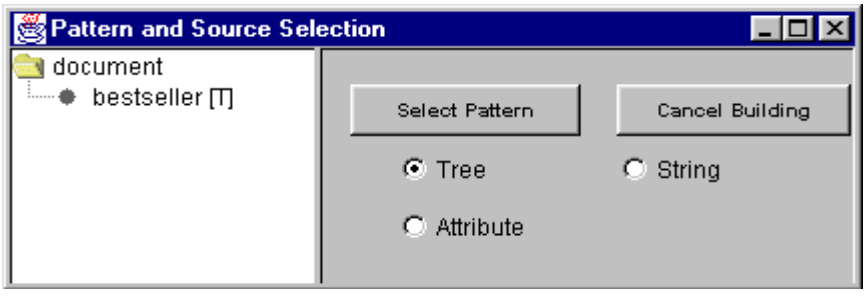
Οι χρήστες επικοινωνούν με το σύστημα προκειμένου να καθορίσουν περιοχές ενδιαφέροντος και να δημιουργήσουν φίλτρα πληροφοριών. Η παραγωγή wrappers με το Lixto βασίζεται σε παραδείγματα που παρέχει ο ίδιος ο χρήστης. Αρχικά, ο χρήστης ανοίγει μία ιστοσελίδα-παράδειγμα κι έπειτα προσθέτει patterns για την εξόρυξη των σχετικών δεδομένων. Αυτά τα patterns φέρουν προκαθορισμένα από τον χρήστη ονόματα, τα οποία χρησιμοποιούνται ως default ονόματα XML elements. Κάθε μοτίβο χαρακτηρίζει ένα είδος πληροφοριών και αποτελείται από διάφορα φίλτρα. Κάθε φίλτρο με την σειρά του απαρτίζεται από διάφορες συνθήκες. Τα στιγμιότυπα ενός pattern είναι τα πραγματικά αντικείμενα-στόχοι (HTML elements, element lists, strings) και τα οποία ικανοποιούν όλες τις συνθήκες τουλάχιστον ενός φίλτρου του αντίστοιχου μοτίβου.

Κατά τον ορισμό ενός φίλτρου, ο χρήστης προσδιορίζει ένα παράδειγμα αντικειμένου-στόχου (κάνοντας απλώς click στην εμφανισμένη ιστοσελίδα) και έναν μηχανισμό επιλογής attributes, ενώ στο υπόβαθρο το σύστημα συντάσσει έναν βασικό κανόνα σε Elog, επιλέγοντας ένα κατάλληλο element path και τα default attributes. Ύστερα, ακολουθεί η προβολή στον χρήστη όλων των αντικειμένων-στόχων που ταιριάζουν στο παρόν φίλτρο. Εφόσον έχουν ταιριαχθεί και μη επιθυμητά αντικείμενα-στόχοι, ο χρήστης έχει την επιλογή του επανακαθορισμού του κανόνα (για παράδειγμα με το να καθορίσει ένα element το οποίο πρέπει να εμφανίζεται πριν το μοτίβο του επιθυμητού αντικειμένου-στόχου). Εάν δεν έχει γίνει ταίριασμα και σε μη επιθυμητά αντικείμενα-στόχους, αλλά μόνο σε ένα υποσύνολο των επιθυμητών, ο χρήστης μπορεί να αποθηκεύσει το φίλτρο και να κατασκευάσει ένα άλλο φίλτρο χρησιμοποιώντας ως νέο παράδειγμα ένα αντικείμενο-στόχο από εκείνα που δεν έχουν συναρμοστεί έως τώρα. Εναλλακτικά, παραθέτοντας συνθήκες και προσθέτοντας νέα φίλτρα μπορεί να χαρακτηρίσει την επιθυμητή πληροφορία πλήρως.

Αφού ολοκληρωθεί η παραγωγή του wrapper, ο τελευταίος είναι σε θέση να χρησιμοποιηθεί κανονικά, προκειμένου να κατασκευάσει την XML έξοδο με την ενδιαφέρουσα πληροφορία από ένα σύνολο ομοίων ιστοσελίδων.

Για την καλύτερη κατανόηση της λειτουργίας του συστήματος Lixto, παραθέτουμε εδώ ένα απλό και σχετικό σενάριο χρήσης, το οποίο αναδεικνύει λεπτομερώς, μέσω οπτικής αναπαράστασης, την δημιουργία ενός μοτίβου και υποδεικνύει τον τρόπο εισαγωγής πρόσθετων συνθηκών. Θεωρούμε ότι ο χρήστης ενδιαφέρεται για τα επίκαιρα bestsellers διαφόρων βιβλιοπωλείων και για τον λόγο αυτό επισκέπτεται την ιστοσελίδα http://www.books.co.uk/. Δεν επιθυμεί να μεταβαίνει συνεχώς στον συγκεκριμένο ιστότοπο, αλλά προτιμά, κάθε φορά που υπάρχει αλλαγή στην κορυφή της κατάταξης, να ενημερώνεται μέσω ενός συντόμου μηνύματος. Πρέπει λοιπόν αρχικά να γράψει ένα πρόγραμμα που να εξάγει τον εκάστοτε τίτλο του βιβλίου με τις υψηλότερες πωλήσεις. Τα προηγούμενα μπορούν να επιτευχθούν ως εξής:

1. Δημιουργία του νέου προγράμματος.
2. Επίσκεψη της ιστοσελίδας του παραδείγματος.
3. Προσθήκη ενός νέου μοτίβου στην ρίζα *<document>*, το οποίο καλούμε *<bestseller>*.



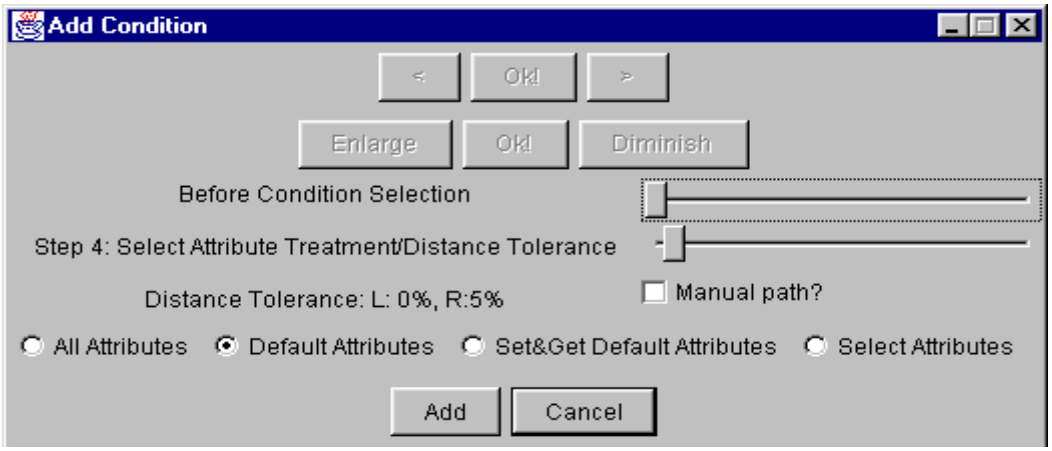
Σχήμα 2.8.1 – Προσθήκη ενός καινούριου μοτίβου

4. Επισήμανση του τίτλου – στόχου με διπλό mouse-click.
5. Επιλογή των προεπιλεγμένων ρυθμίσεων για τα attributes (για το συγκεκριμένο παράδειγμα, αυτό σημαίνει ότι τα links θεωρούνται χαρακτηριστικές ιδιότητες).
6. Έλεγχος του τρέχοντος φίλτρου.

Σχήμα 2.8.2 – Εξέταση ενός φίλτρου

Σε αυτή την περίπτωση, παρατηρούμε ότι ο έλεγχος του εν λόγω φίλτρου επιστρέφει πολλούς στόχους μαρκαρισμένους. Κάθε βιβλίο στην λίστα με τα 10 καλύτερα (όσον αφορά τις πωλήσεις τους) ταιριάζει σε αυτό το φίλτρο. Εδώ, που ο χρήστης ενδιαφέρεται για ένα μοναδικό ταίριασμα, πρέπει να περιορισθεί το χρησιμοποιούμενο μοτίβο.

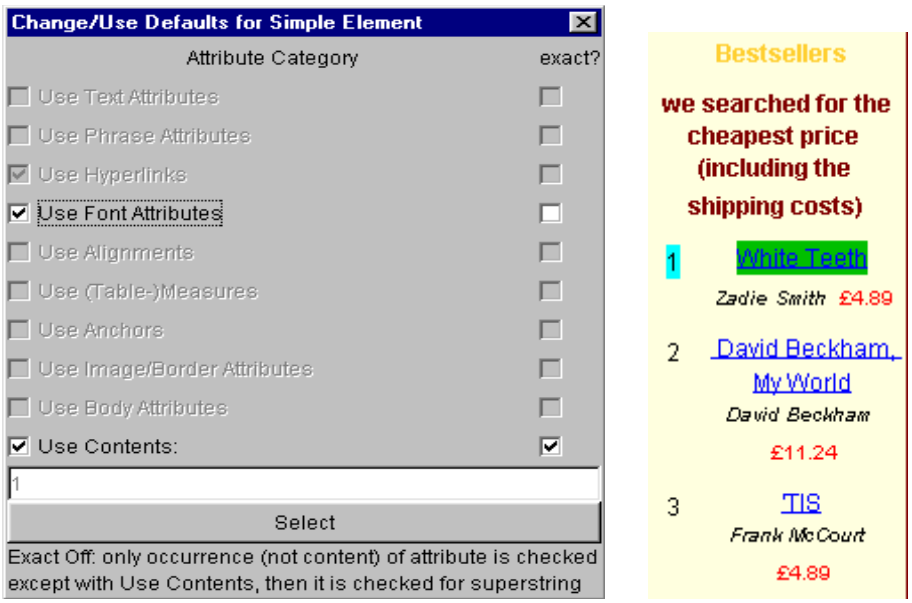
7. Προσθήκη «συνθήκης προηγούμενου στοιχείου», η οποία εκφράζει ότι ένα συγκεκριμένο στοιχείο πρέπει να εμφανίζεται ακριβώς πριν το επιθυμητό μοτίβο του στόχου.
8. Επιλογή του λεγόμενου «προηγούμενου στοιχείου», με το σύστημα να επιτρέπει την εξερεύνηση μέχρι την επίτευξη της ακριβούς ρύθμισης.
9. Καθορισμός των παραμέτρων ανεκτών αποστάσεων, δηλαδή σε ποιο διάστημα το στοιχείο-στόχος μπορεί να εμφανισθεί.



Σχήμα 2.8.3 – Προσθήκη ενός «προηγούμενου» στοιχείου

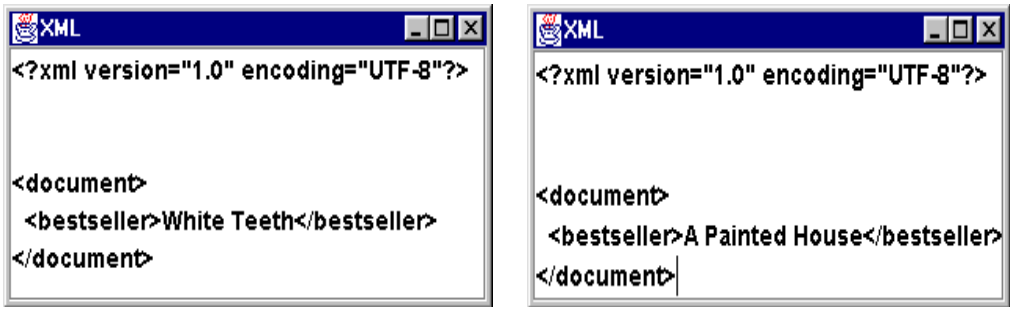
Σε αυτό το είδος επιλογής, τοποθετούμε τα attributes σε πολλές ομαδοποιήσεις, έτσι ώστε οι χρήστες που δεν διαθέτουν γνώσεις HTML, να είναι σε θέση να προσδιορίσουν για παράδειγμα ότι η γραμματοσειρά ενός συγκεκριμένου στοιχείου είναι χαρακτηριστική για τον στόχο, και για τον λόγο αυτό πρέπει να αξιοποιηθεί κατά την εξαγωγή.

10. Επιλογή της χρήσης των περιεχομένων ("Use Contents") και της χρήσης του ενδεικτικού αριθμού 1 ως ακριβούς προσδιοριστή ("Use Exact Contents: 1"). Σημειώνουμε, ότι το σύστημα μπορεί να διακρίνει και να αποφύγει τις ανεπιθύμητες περιπτώσεις, όπου ο εν λόγω αριθμός βρίσκεται εντός και άλλων στοιχείων (όπως π.χ. μέσα στις τιμές των βιβλίων).
11. Έλεγχος του συγκεκριμένου φίλτρου. Παρατηρούμε ότι τώρα πραγματοποιείται η εξαγωγή του επιθυμητού στοιχείου-στόχου και μόνο.
12. Αποθήκευση του φίλτρου, του μοτίβου και του προγράμματος.



Σχήμα 2.8.4 – Σημείωση ενός «προηγούμενου» στοιχείου

Αυτό το πρόγραμμα μπορεί να χρησιμοποιηθεί και σε νέες εκδόσεις της ιστοσελίδας, εκμεταλλευόμενο το υποσύστημα του συνεχούς εξαγωγέα του Lixto (που αξιοποιεί XML companions για άλλες χρονικές στιγμές, π.χ. για δύο διαφορετικές εβδομάδες).



Σχήμα 2.8.5 – XML Companions

Ομοίως, μπορεί να δημιουργηθεί ένας wrapper για την εξαγωγή των τριών πρώτων bestsellers (κάνοντας χρήση μίας «συνθήκης διαστήματος») ή για την εξαγωγή των bestsellers ενός συγκεκριμένου συγγραφέα (χρησιμοποιώντας μία «συνθήκη επόμενου στοιχείου»). Εκτός αυτών, μπορεί να αξιοποιηθεί η τεχνική της ιεραρχικής εξαγωγής, προκειμένου αρχικά να καθορίζεται μία καταχώρηση ενός βιβλίου, και έπειτα μέσα σε αυτήν να προσδιορίζονται ο τίτλος, ο συγγραφέας και η τιμή του. Άλλα ενδιαφέροντα παραδείγματα περιλαμβάνουν την παραγωγή wrappers για δημοπρασίες στο E-Bay ή το Yahoo (που παρουσιάζουν κοινό mapping scheme), το screen-scarping του DBLP, και την προσπέλαση διαφόρων wrappers για τηλεοπτικές μεταδόσεις μέσω ιστοσελίδων. Σε τέτοια σύνθετα παραδείγματα, πιο προηγμένες δυνατότητες του Lixto, όπως οι επονομαζόμενες concepts (που λαμβάνουν υπόψη τους την ημερομηνία ή την γεωγραφική τοποθεσία), τα hyperlinks και η αναδρομική εξόρυξη δεδομένων, είναι ιδιαίτερα χρήσιμες.

3

Ανάλυση απαιτήσεων συστήματος

Το κεφάλαιο αυτό αφορά στην αρχιτεκτονική του συστήματός μας, την ανάλυση των απαιτήσεων λειτουργίας, καθώς και την εκάστοτε θεωρούμενη μέθοδο πραγμάτωσης των ζητούμενων λειτουργιών.

3.1 Αρχιτεκτονική

Η ενότητα αυτή, αν και εισαγωγική, είναι ιδιαίτερα σημαντική καθώς παρουσιάζει την συνολική μορφή – την αρχιτεκτονική του συστήματος, την σύνδεση της θεωρίας περί mindmaps με την επιλεχθείσα εφαρμογή δημιουργίας τους, ενώ προβαίνει και σε ανάδειξη του συγκεκριμένου τρόπου προσέγγισης του ζητήματος της προηγμένης διαδικτυακής μετα-αναζήτησης. Τέλος, εδώ απαριθμούνται και οι κύριες λειτουργίες του συστήματός μας.

Η παρούσα διπλωματική εργασία θα βασισθεί στο εργαλείο ανάπτυξης mindmaps που καλείται *FreeMind*. Πρόκειται για μία παγκοσμίως δημοφιλή εφαρμογή χαρτογράφησης ιδεών – σκέψεων – εννοιών, η οποία είναι διαθέσιμη σε κάθε ενδιαφερόμενο χρήστη ως εφαρμογή ανοικτού κώδικα (open source application) με άδεια χρήσης & ανάπτυξης τύπου GNU General Public License. Το εν λόγω εργαλείο μπορεί να βρεθεί στον ιστότοπο http://sourceforge.net/projects/freemind/, όπου είναι διαθέσιμο για downloading. Επίσης, πέρα από την απαίτηση για σχεδίαση τέτοιων χαρτών, βασικό ζητούμενο ετούτης της εργασίας είναι η ενσωμάτωση λειτουργιών διαδικτυακής διαδραστικότητας στην συγκεκριμένη mind-mapping εφαρμογή. Έτσι, θα πρέπει να υπάρχουν νέα υποπαράθυρα, που

να ικανοποιούν την απαίτηση αυτή και τα οποία θα ενεργοποιούνται ενδεχομένως με buttons που προτιθέμεθα να προσθέσουμε στο κεντρικό παράθυρο του FreeMind. Πλέον, θα ενσωματωθεί η διαδικτυακή αναζήτηση βάσει προεγκατεστημένων μηχανών αναζήτησης (θα πρόκειται μάλιστα για προηγμένη μετα-αναζήτηση) στα πλαίσια ενδιαφέροντος που μελλοντικά θα ορίζει το εκάστοτε mindmap μας. Επιπροσθέτως, θα κατασταθεί δυνατή η πλοήγηση στον Παγκόσμιο Ιστό, ενώ ακόμα θα υποστηρίζεται η γρήγορη, εύκολη και φιλική προς τον χρήστη εγκατάσταση καινούριων μηχανών αναζήτησης που θα εκμεταλλεύεται την προαναφερθείσα τεχνική του screen-scraping. Επίσης, ειδική μέριμνα πρέπει να ληφθεί για αποτελέσματα τύπου papers, φιλοδοξώντας να αξιοποιήσει την έγκριτη βάση δεδομένων του Trier Universitaet DBLP. Ήδη από τα προηγούμενα διαφαίνονται οι θεμελιώδεις λειτουργίες του συστήματός μας. Αυτές, συνοψίζονται μονοφραστικά στις κάτωθι:

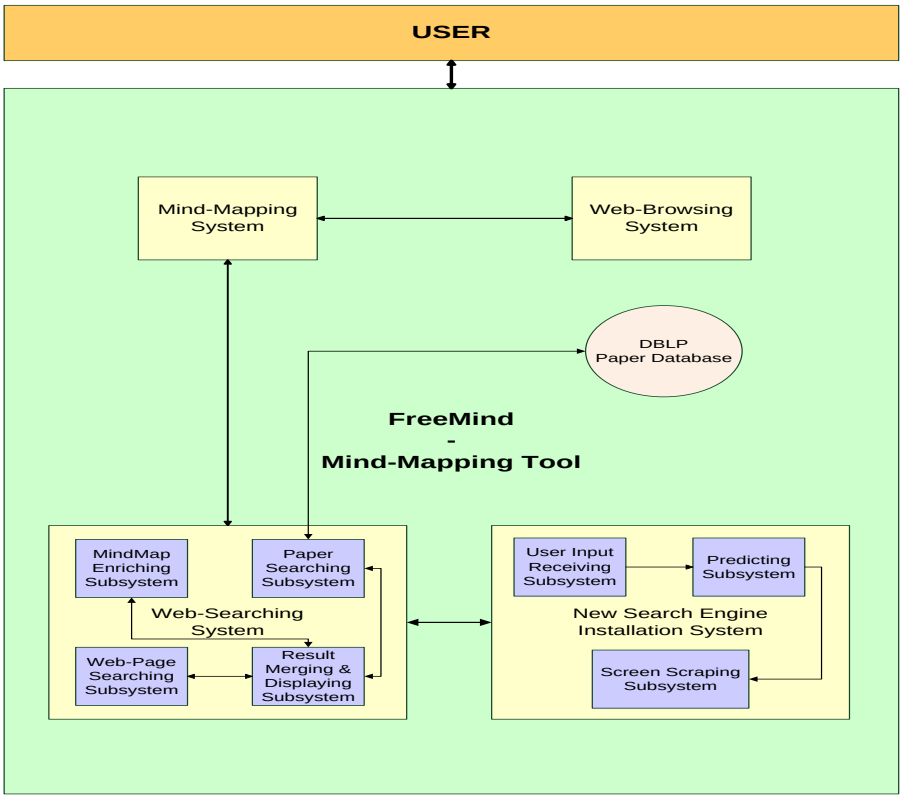
I. Λειτουργία ανάπτυξης χαρτών σκέψεων.

II. Λειτουργία αναζήτησης σχετικών πληροφοριών στο Διαδίκτυο και ενσωμάτωσής τους στον χάρτη ιδεών.

III. Λειτουργία πλοήγησης στον World Wide Web.

IV. Λειτουργία εμπλουτισμού της διαδικασίας του web searching μέσω της προσθήκης νέων μηχανών αναζήτησης.

Στο σημείο αυτό ακολουθεί ένα block διάγραμμα του συνολικού συστήματος με τα υποσυστήματά του και κάποιες λεπτομέρειες, οι οποίες επεξηγούνται παρακάτω:



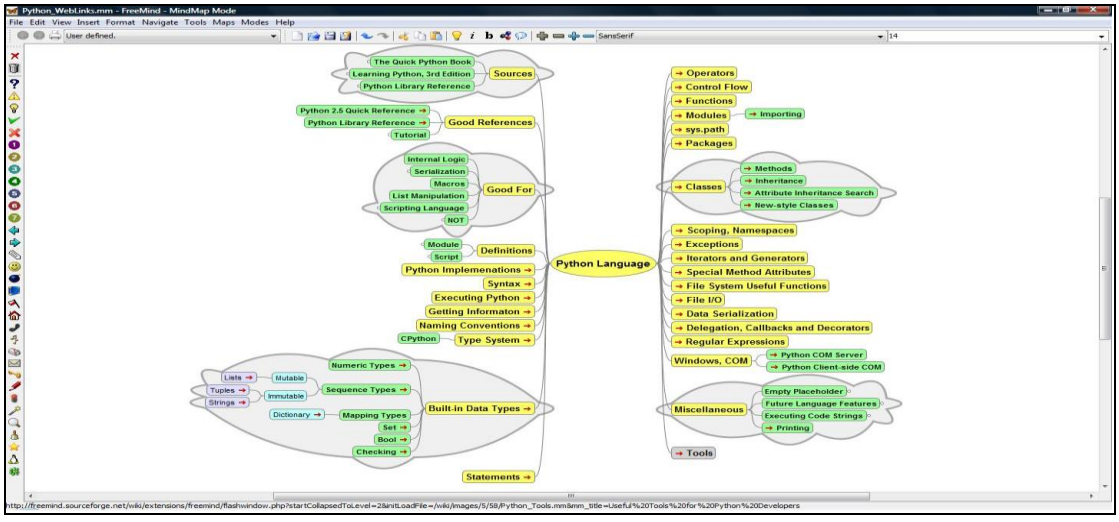
Σχήμα 3.1 – Μπλοκ διάγραμμα του συνολικού συστήματος

3.2 Περιγραφή Λειτουργιών

Στην παρούσα ενότητα περιγράφονται οι λειτουργίες που απαιτείται να εκτελεί το σύστημα. Έχοντας διαχωρίσει, ακριβώς παραπάνω, το σύστημά μας σε βασικά υποσυστήματα, εδώ θα συνεχίσουμε με την περιγραφή του κάθε υποσυστήματος ξεχωριστά. Κατά κανόνα, δίνουμε αρχικά το κείμενο με την λεκτική περιγραφή του εκάστοτε υποσυστήματος και στην συνέχεια αντίστοιχο σχετικό παράδειγμα ή διάγραμμα.

3.2.1 Υποσύστημα ανάπτυξης χαρτών – γράφων τύπου mindmap

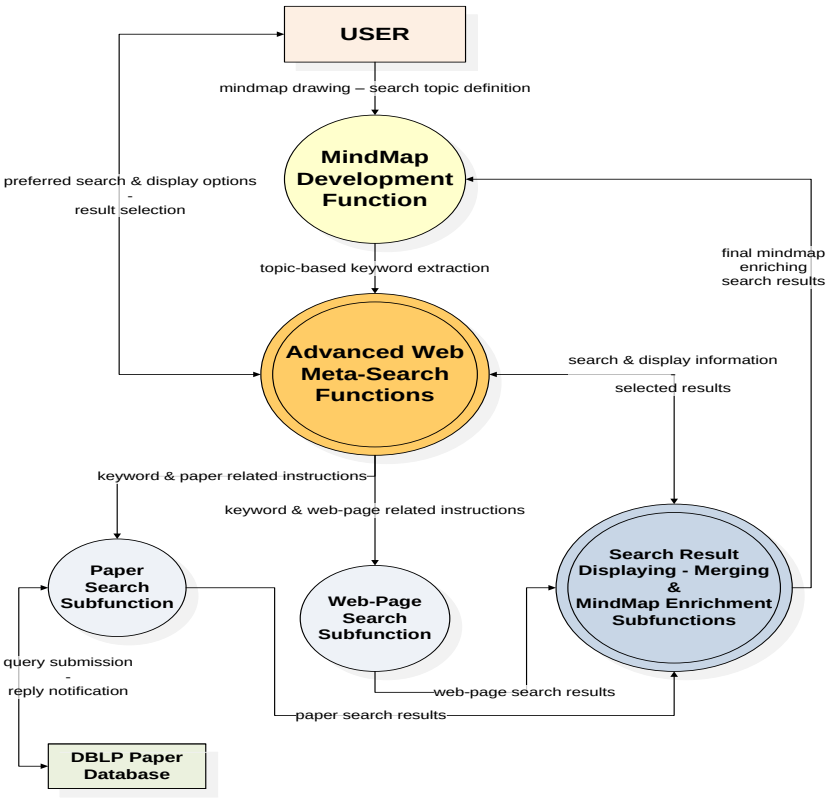
Μία από τα ουσιαστικότερες λειτουργίες του συστήματός μας είναι εκείνη της ανάπτυξης mindmaps. Το περιβάλλον εργασίας που επιλέξαμε θεωρείται ένα από τα πλέον κατάλληλα για τον σκοπό αυτό. Η κεντρική οθόνη εκτέλεσης του FreeMind έχει την μορφή ενός παραθύρου γραφικών, στο κυρίως τμήμα του οποίου αναπτύσσεται το εκάστοτε mindmap, ενώ βοηθητικές λειτουργίες παρέχονται περιφερειακά είτε με την μορφή buttons είτε με την μορφή menus. Ως εργαλείο κατασκευής χαρτών σκέψης, το FreeMind, επιτρέπει την ανάπτυξη ιδιότυπων γράφων, με κόμβους τοποθετούμενους κυκλικά γύρω από άλλους κόμβους, και κλάδους που συνδέουν τους διάφορους κόμβους μεταξύ τους. Οι κόμβοι μπορούν να σημειωθούν είτε με κάποιο ειδικό σύμβολο (όπως ένας αριθμός, ένας λαμπτήρας, ένα ερωτηματικό κ.ά.) είτε με κάποιο άλλο χαρακτηριστικό (π.χ. βέλος hyperlink, blinking ή bold γραμματοσειρά, συνημμένη σημείωση κ.ά.) που αναδεικνύει μία περαιτέρω σημασία ή λειτουργικότητα. Ακόμα, αξίζουν να σημειωθούν τόσο η δυνατότητα αναδίπλωσης και πλήρους ανάπτυξης των κόμβων (folding-unfolding), όσο και η ύπαρξη σημασιολογικών συγκροτημάτων κόμβων – νεφελών (clouds-bubbles), εφόσον κάτι τέτοιο είναι επιθυμητό. Ακολούθως παραθέτουμε ένα screenshot του FreeMind εν δράσει, δηλαδή κατά την ανάπτυξη ενός mindmap με αυτό:



Σχήμα 3.2 – MindMap για την γλώσσα προγραμματισμού Python

3.2.2 Υποσύστημα προηγμένης διαδικτυακής μετα-αναζήτησης

Το εν λόγω υποσύστημα είναι επιφορτισμένο με την λειτουργία της αναζήτησης πληροφοριών στον Παγκόσμιο Ιστό. Η αναζήτηση αυτή θα γίνεται στα πλαίσια ενός topic του υπό ανάπτυξη mindmap. Ακόμα, εξαιτίας του γεγονότος ότι θα μπορεί να συλλέγει δεδομένα από πληθώρα μηχανών αναζήτησης και όχι από μία αποκλειστικά (ενώ εν συνεχεία θα πρέπει να τα κατατάσσει κατάλληλα και τελικά να τα συγχωνεύει σε μία μοναδική λίστα), η συγκεκριμένη διαδικασία καλείται μετα-αναζήτηση (meta-search). Επιπροσθέτως, επειδή το searching θα είναι σε θέση να εξειδικεύεται με περαιτέρω περιορισμούς, η λειτουργία λαμβάνει και τον χαρακτηρισμό της «προηγμένης» (advanced search). Πρέπει επίσης να τονιστεί ότι η λίστα δεδομένων που θα επιστρέφεται ως αποτέλεσμα της αναζήτησης χρειάζεται να είναι διαθέσιμη στον χρήστη για πληροφόρηση, αλλά συγχρόνως και για προεπισκόπηση, προκειμένου να μπορεί να προβεί ακολούθως ο ίδιος σε μία διαλογή των ενδιαφερόντων search results και να επιλέξει μόνο εκείνα, τα οποία θα θεωρεί ότι καλύπτουν τις ανάγκες ή επιθυμίες του. Τελικώς, τα επιλεχθέντα αποτελέσματα θα μπορούν να χρησιμοποιηθούν για εμπλουτισμό (enrichment) του αντιστοίχου topic επί του οποίου θα έχει βασιστεί αρχικά η αναζήτηση. Όλα τα ανωτέρω αναλύονται περισσότερο στις υποενότητες που παρατίθενται παρακάτω, ενώ αμέσως τώρα δίνουμε το διάγραμμα ροής δεδομένων (data flow diagram) της συζητούμενης λειτουργίας:



Σχήμα 3.3 – Διάγραμμα ροής δεδομένων

του υποσυστήματος προηγμένης διαδικτυακής μετα-αναζήτησης

3.2.2.1 Αναζήτηση σε ιστοσελίδες γενικού περιεχομένου

Για την υλοποίηση της υπηρεσίας διαδικτυακής μετα-αναζήτησης, το σύστημά μας θα πρέπει να είναι σε θέση να εκμεταλλεύεται διάφορες αξιόπιστες και διάσημες μηχανές αναζήτησης, οι οποίες μπορούν γενικά να διακρίνονται σε δύο είδη ανάλογα με τον τρόπο ενσωμάτωσής τους στην εφαρμογή μας:

1) Μηχανές αναζήτησης που θα παρέχουν δυνατότητες υποβολής ερωτήσεων για αναζήτηση και λήψης αποτελεσμάτων ως απάντηση, με την μορφή κάποιου API (Application Programming Interface).
2) Μηχανές αναζήτησης για τις οποίες ο χρήστης θα παρέχει ορισμένες πληροφορίες, ενώ επιπλέον θα βοηθάει ο ίδιος το σύστημα να καταφέρει να μάθει να «διαβάζει» αποτελέσματα από τις αντίστοιχες ιστοσελίδες, μέσω των τεχνικών του screen-scraping (απόξεση οθόνης) και του machine-learning (εκμάθηση μηχανής).

Έτσι, θα καταλήξουμε έχοντας στην διάθεσή μας ένα πλήθος API-embedded search engines και ένα άλλο πλήθος wrapped search engines.

- Η web searching λειτουργία θα πρέπει να αξιοποιεί όλες τις διαθέσιμες μηχανές αναζήτησης, και είτε θα υποβάλλει σχετικά queries (όταν πρόκειται για μηχανές του πρώτου είδους) αναμένοντας replies, είτε θα επισκέπτεται («κρυφά» από τον χρήστη) τα αντίστοιχα websites (στην περίπτωση μηχανών του δευτέρου είδους) και θα διαβάζει τα results. Χαρακτηριστικά αναφέρουμε τις ακόλουθες μηχανές αναζήτησης ως υποψήφιες προς ενσωμάτωση (με τον έναν ή τον άλλον τρόπο) στο σύστημά μας: Google Search (αναζήτηση σε γενικό περιεχόμενο), Yahoo Search (αναζήτηση σε γενικό περιεχόμενο), Microsoft Live Search (αναζήτηση σε γενικό περιεχόμενο), Gigablast Search (αναζήτηση σε γενικό περιεχόμενο), Google Scholar Paper Search (αναζήτηση σε επιστημονικές δημοσιεύσεις), Technorati Blog Search (αναζήτηση σε ιστολόγια), YouTube Video Search (αναζήτηση σε videos) [WP5].
- Επιθυμούμε η αναζήτηση να πραγματοποιείται χρησιμοποιώντας ως λέξη-κλειδί το κείμενο του εκάστοτε επιλεγμένου κόμβου του χάρτη σκέψεων με τον οποίο θα εργαζόμαστε. Εκτός από αυτό, θα θέλαμε να είναι στην ευχέρεια του χρήστη να δώσει πρόσθετα keywords, καθώς και να του δίνεται η δυνατότητα να επιλέξει το επιθυμητό πλήθος αποτελεσμάτων.
- Τα αποτελέσματα που θα επιστρέφει η κάθε μηχανή αναζήτησης θα πρέπει να συλλέγονται, να κατατάσσονται και να συγχωνεύονται καταλλήλως (υποδεικνύουμε και αναλύουμε αργότερα έναν αλγόριθμο ο οποίος μπορεί να συμβάλλει καθοριστικά προς την κατεύθυνση αυτή), προσφέροντας στον χρήστη την καλύτερη δυνατή

συγκεντρωτική λίστα, για εύχρηστη και γρήγορη ικανοποίηση των αναγκών και επιθυμιών του.

Σημειώνουμε εδώ ότι η αναζήτηση σε ιστοσελίδες γενικού περιεχομένου οφείλει να κάνει χρήση των σχετικών πάντα μηχανών, καθώς γενικά κάθε μηχανή αναζήτησης θα πρέπει να «πριμοδοτείται» σε πλήθος και βαρύτητα αποτελεσμάτων εφόσον η εκτελούμενη αναζήτηση θα εμπίπτει στο πλαίσιο δράσης της, ενώ σε αντίθετη περίπτωση ο ρόλος της μηχανής αναζήτησης είναι αναγκαίο να υποβαθμίζεται. Ένα αποτέλεσμα μπορεί να φέρει την συνήθη μορφή που θα έβρισκε κανείς επισκεπτόμενος και τον διαδικτυακό τόπο μίας τέτοιας μηχανής, δηλαδή: τίτλος (title) – περίληψη (description ή summary, μόνο εφόσον αυτή θα είναι επιθυμητή) – σύνδεσμος (link ή URL ή WWW address).

3.2.2.2 Αναζήτηση σε επιστημονικές δημοσιεύσεις

Με όμοιο τρόπο με την αναζήτηση σε ιστοσελίδες θα πρέπει να πραγματοποιείται και η αναζήτηση σε επιστημονικές δημοσιεύσεις (papers), παρουσιάζοντας ωστόσο μερικές ουσιαστικές διαφορές. Το κύριο διαφοροποιητικό στοιχείο απαντάται στην αξιοποίηση μίας βάσης δεδομένων, η οποία θα χρησιμεύει τόσο για να επιβεβαιώσει και να πιστοποιήσει ότι μία δημοσίευση βρίσκεται και στην εν λόγω έγκριτη βάση δεδομένων, όσο και για να παράσχει επιπρόσθετες πληροφορίες για την δημοσίευση αυτή σε περίπτωση θετικού αποτελέσματος ελέγχου.

Το DBLP (Digital Bibliography & Library Project) αποτελεί έναν ιστότοπο που περιλαμβάνει οργανωμένη βιβλιογραφία της Επιστήμης των Υπολογιστών (Computer Science) και φιλοξενείται από το πανεπιστήμιο του Trier στην Γερμανία (Deutsche Universität Trier) [Ley02]. Ο κάθε ενδιαφερόμενος μπορεί να επισκεφθεί τον εν λόγω ιστότοπο, ακολουθώντας τον σύνδεσμο: http://www.informatik.uni-trier.de/~ley/db/. Το DBLP περιλαμβάνει κατά την περίοδο συγγραφής της παρούσας διπλωματικής εργασίας (Μάρτιος 2009) 1.130.693 βιβλιογραφικές καταχωρήσεις. Μεταξύ αυτών βρίσκονται οι IEEE (*Institute of Electrical and Electronics Engineers*) και ACM (*Association for Computer Machinery*) transactions, καθώς και δημοσιεύσεις επιστημονικών περιοδικών ή συνεδρίων. Όπως προαναφέρθηκε, το σύστημά μας θα αναζητά πληροφορίες με τις κατάλληλες κάθε φορά μηχανές αναζήτησης, που στην περίπτωση της έρευνας για δημοσιεύσεις, λαμβάνοντας υπόψη το σύνολο των μηχανών προς ενσωμάτωση, η πλέον ενδεδειγμένη είναι το Google Scholar Paper Search.

Η ειδοποιός διαφορά κατά τον εν λόγω τρόπο αναζήτησης πρέπει να έγκειται στο γεγονός ότι κάθε αποτέλεσμα, πριν τοποθετηθεί στην τελική συγκεντρωτική λίστα, θα διέρχεται μέσα από έναν έλεγχο. Ο έλεγχος θα αφορά στο εάν το αποτέλεσμα αυτό φέρει τον ίδιο (ή αρκετά παραπλήσιο) τίτλο με κάποια βιβλιογραφική εγγραφή στην βάση δεδομένων του DBLP. Σε περίπτωση αρνητικής απάντησης το αποτέλεσμα απαιτείται να καταχωρηθεί στην λίστα ως

έχει, ενώ σε αντίθετη περίπτωση χρειάζεται ο εντοπισμός της συγκεκριμένης εγγραφής και η ανάκτηση όλων των επιθυμητών σχετικών πληροφοριών, εμπλουτίζοντας έτσι το αποτέλεσμα, ενώ αυτό παράλληλα θα εμφανίζεται να φέρει το κύρος που του προσδίδει η εν λόγω διεθνώς αναγνωρισμένη βάση δεδομένων.

Τα αποτελέσματα που επιστρέφουν οι σχετικές μηχανές αναζήτησης είναι συνήθως των γενικών τύπων paper, article, book, citation, pdf, ενώ οι καταχωρήσεις στο DBLP είναι των τύπων article, inproceedings, book, www, phd thesis, incollection, proceedings, master's thesis. Όσα αποτελέσματα προέρχονται από το Διαδίκτυο περιλαμβάνουν συνήθως τίτλο, συγγραφείς, έτος, βιβλίο / περιοδικό / συνέδριο, περίληψη και σχετικό υπερσύνδεσμο (hyperlink). Οι βιβλιογραφικές εγγραφές του DBLP παρέχουν διαφορετικές πληροφορίες για τους διάφορους απαντώμενους τύπους. Πιο αναλυτικά, κάθε εγγραφή περιλαμβάνει ορισμένα από τα ακόλουθα πεδία: τύπος, συγγραφείς, τίτλος, περιοχή σελίδων, περιοδικό, τόμος περιοδικού, αριθμός περιοδικού, έτος συγγραφής, διαδικτυακός σύνδεσμος, ιστοσελίδα στο DBLP, ημερομηνία προσθήκης στη βάση δεδομένων, βιβλίο, ISBN, σειρά περιοδικών ή βιβλίων, ιστότοπος σειράς, εκδότης περιοδικού ή βιβλίου, ιστότοπος εκδότη και εκπαιδευτικό ίδρυμα.

3.2.2.3 Αναζήτηση και σε άλλους τύπους δεδομένων – Προηγμένη Αναζήτηση

Η αναζήτηση πληροφοριών στον Παγκόσμιο Ιστό δεν θα πρέπει να περιορίζεται σε ιστοσελίδες γενικού περιεχομένου (web-pages) και επιστημονικές δημοσιεύσεις (papers), αλλά να αφορά και ιστολόγια (blogs), έγγραφα με την ευρεία έννοια (documents, όπως αρχεία .txt, .doc, .pdf, .rtf, .xls, .ppt), χάρτες (maps), εικόνες (images), ηχητικό υλικό (audio) & οπτικό υλικό (video).

Με αυτό το σκεπτικό, οφείλουμε να προετοιμάσουμε το σύστημά μας και να το κάνουμε επιδεκτικό σε τέτοιου είδους επεκτάσεις. Πιο συγκεκριμένα, τόσο τα γραφικά στοιχεία (GUIS κ.ά.) όσο και οι τηρούμενες εσωτερικές μορφές δεδομένων χρειάζεται να κατασκευαστούν έτσι, ούτως ώστε να υποστηρίζουν όλους τους προαναφερθέντες τύπους δεδομένων. Εκείνο που θα απομείνει τελικά είναι η συγγραφή του αντίστοιχου κώδικα, που θα καταστήσει και τους εν λόγω τύπους λειτουργικούς. Ακόμα, άξιο λόγου είναι το χαρακτηριστικό της προηγμένης αναζήτησης που θα υποστηρίζει, αλλά, στα πλαίσια της παρούσας διπλωματικής, δεν θα ενσωματώσει το σύστημά μας. Έτσι μελλοντικά, για κάθε έναν από τους προηγούμενους τύπους δεδομένων, θα δίνεται η δυνατότητα στον χρήστη να πραγματοποιεί μία πιο εξειδικευμένη αναζήτηση, εκμεταλλευόμενος τα διαθέσιμα φίλτρα περιορισμού των επιστρεφόμενων αποτελεσμάτων.

I. Πιο αναλυτικά, η λίστα αποτελεσμάτων όσον αφορά μία προηγμένη αναζήτηση ανάμεσα σε general web-pages θα μπορεί να περιλαμβάνει επιπρόσθετα κριτήρια

όπως εκείνο της ημερομηνίας (date-based search, π.χ. έως 2 μήνες παλαιά αποτελέσματα), αυτό του διαδικτυακού τόπου προέλευσης (domain-based search, π.χ. .com, .org ή .edu), εκείνο του τύπου αρχείου (file type-based search, π.χ. .html, .pdf ή .xml) και αυτό της περιοχής προέλευσης (region-based search, π.χ. .gr, .eu ή .us).

II. Για την αναζήτηση μεταξύ papers, τα φίλτρα δύναται να σχετίζονται με τον συγγραφέα (author-based search, π.χ. Neumann), τον εκδότη (publisher-based search, π.χ. IEEE Journal), την ημερομηνία (date-based search) και την θεματική περιοχή (subject-based search, π.χ. Nuclear Physics).

III. Όσον αφορά τα blogs, οι περιορισμοί θα αναφέρονται σε συγκεκριμένο ιστολόγιο (blog-based search, π.χ. http://computersciencestudy.blogspot.com/) ή ανάρτηση (post-based search, π.χ. Meta-Searching) ή ετικέτα (tag-based search, π.χ. web-crawler) ή ιστοσελίδα στην οποία συνδέονται blogs (URL-based search, π.χ. http://edition.cnn.com/services/rss/).

IV. Σχετικά με την αναζήτηση σε documents, αυτή θα μπορεί να εξειδικευθεί βάσει της ημερομηνίας (date-based search), του διαδικτυακού τόπου προέλευσης (domain-based search), του τύπου αρχείου (file-type based search, π.χ. .txt, .doc, ή .rtf) και του μεγέθους αρχείου (size-based search, π.χ. μικρότερο ή μεγαλύτερο του 1MB).

V. Εκτός των ανωτέρω, σημειώνουμε πως όταν η αναζήτηση θα γίνεται σε maps, το σύστημα δεν θα προβλέπει πρόσθετα περιοριστικά στοιχεία.

VI. Επιπλέον, για αναζήτηση αναφερόμενη σε images, η εφαρμογή μας θα μπορεί να διαθέτει φίλτρα είτε βασισμένα στον χρωματισμό (coloration-based search, π.χ. color, grayscale ή black & white), είτε στον διαδικτυακό τόπο προέλευσης (domain-based search), είτε στον τύπο αρχείου (file type-based search, π.χ. .bmp, .jpg ή .gif), είτε στο μέγεθος αρχείου (size-based search, π.χ. μικρό, μεσαίο ή μεγάλο).

VII. Επίσης, όταν ο χρήστης αναζητά ορίζοντας ως επιθυμητό τύπο δεδομένων το audio, χρειάζεται να υπάρχει μέριμνα για λήψη αποτελεσμάτων ανάλογα με την ποιότητα ήχου (minimum bitrate-based search, π.χ. ανώτερη των 192Kbps), την διάρκεια (minimum duration-based search, π.χ. μεγαλύτερη των 4 λεπτών), τον τύπο αρχείου (file type-based search, π.χ. .wav, .mp3 ή .wma) και την ενδεχόμενη κατηγορία που θα υπάγεται το εν λόγω ηχητικό απόσπασμα (genre-based search π.χ. Jazz, Rock ή Country).

VIII. Τέλος, για αναζητήσεις σε video, τα αποτελέσματα θα μπορούν να φιλτραριστούν σύμφωνα με την ημερομηνία (date-based search), την διάρκεια (minimum duration-based search), τον τύπο αρχείου (file type-based search, π.χ. .avi, .mpg ή .mov) και το μέγεθος αρχείου (size-based search, π.χ. μικρό, μεσαίο ή μεγάλο).

3.2.2.4 Εμφάνιση των αποτελεσμάτων αναζήτησης και εμπλουτισμός του mindmap

Το σύστημά μας θα είναι σε θέση να παρέχει τέσσερεις τρόπους οργάνωσης και παρουσίασης των αποτελεσμάτων στον χρήστη. Αυτοί θα έχουν να κάνουν με το εάν προτιμάται συγκεντρωτική λίστα ή ταξινόμηση ανά μηχανή αναζήτησης, και σύντομη ή εκτενής μορφή αποτελέσματος.

Στην μεν περίπτωση των συγκεντρωμένων αποτελεσμάτων, ο χρήστης θα λαμβάνει τελικά επί της οθόνης του έναν κατάλογο, ο οποίος θα περιέχει τα διάφορα αποτελέσματα συγχωνευμένα μεταξύ τους και ταξινομημένα κατά φθίνουσα σημαντικότητα. Στην δε περίπτωση των διαχωρισμένων αποτελεσμάτων, η λίστα που θα φθάνει στον χρήστη θα περιλαμβάνει τα αποτελέσματα ανά μηχανή αναζήτησης, όπου και πάλι είναι απαραίτητο να ακολουθείται η λογική της μειούμενης βαρύτητας στην θέση εμφάνισης των αποτελεσμάτων, όμως εδώ η κατάταξη θα πρέπει να αφορά μόνο στοιχεία του ίδιου συνόλου (δηλαδή αποτελέσματα προερχόμενα από την ίδια μηχανή αναζήτησης), ενώ οι διάφορες ομάδες αποτελεσμάτων θα έχουν συμπεριληφθεί στον συνολικό κατάλογο κατά την αλφαβητική σειρά των ονομασιών των μηχανών προέλευσης.

Η σύντομη μορφή παρουσίασης ενός αποτελέσματος θα θέλαμε να περιλαμβάνει μόνο τα ουσιαστικότερα δεδομένα του (π.χ. title, link), ενώ η εκτενής μορφή επιθυμούμε να προσφέρει όλες τις διαθέσιμες πληροφορίες (π.χ. και summary).

Τονίζουμε στο σημείο αυτό, ότι η μεθοδολογία της εκάστοτε απαιτούμενης κατάταξης και συγχώνευσης αποτελεσμάτων αποφασίσαμε να υποδεικνύεται από τον διάσημο και «δημοκρατικό» αλγόριθμο Borda – Fuse [AM01].

- Σύμφωνα με αυτόν, κάθε μηχανή αναζήτησης θα έχει ορισμένη βαθμολογία για κάθε τύπο δεδομένων, όπως επίσης και κάθε αποτέλεσμα θα φέρει έναν βαθμό.
- Όσο καλύτερη θα θεωρείται μία μηχανή για έναν τύπο δεδομένων, τόσο υψηλότερη θα είναι και η βαθμολογία της (π.χ. 5 = άριστα, 3 = μέτρια, 1 = κακώς), ενώ όσο υψηλότερα στην λίστα αποτελεσμάτων της μηχανής θα απαντάται ένα αποτέλεσμα, τόσο καλύτερος θα είναι ο βαθμός του [LM06] (π.χ. για 200 αποτελέσματα, το πρώτο βαθμολογείται με 200, το μεσαίο στην λίστα με 100 και το τελευταίο με 1).
- Έτσι, κάθε αποτέλεσμα θα έχει τον δικό του βαθμό και εκείνον της μηχανής προέλευσής του, επομένως πολλαπλασιάζοντας τους δύο αυτούς αριθμούς θα προκύπτει η τελική βαθμολογία του (π.χ. το 43^{ο} αποτέλεσμα από 100 που περιλαμβάνει η λίστα, την οποία επέστρεψε η μηχανή αναζήτησης με βαθμό 2 / 5 στον x τύπο δεδομένων, έχει συνολική αξιολόγηση: (100 – 43) * (2 / 5) = 22.8 στα 100).

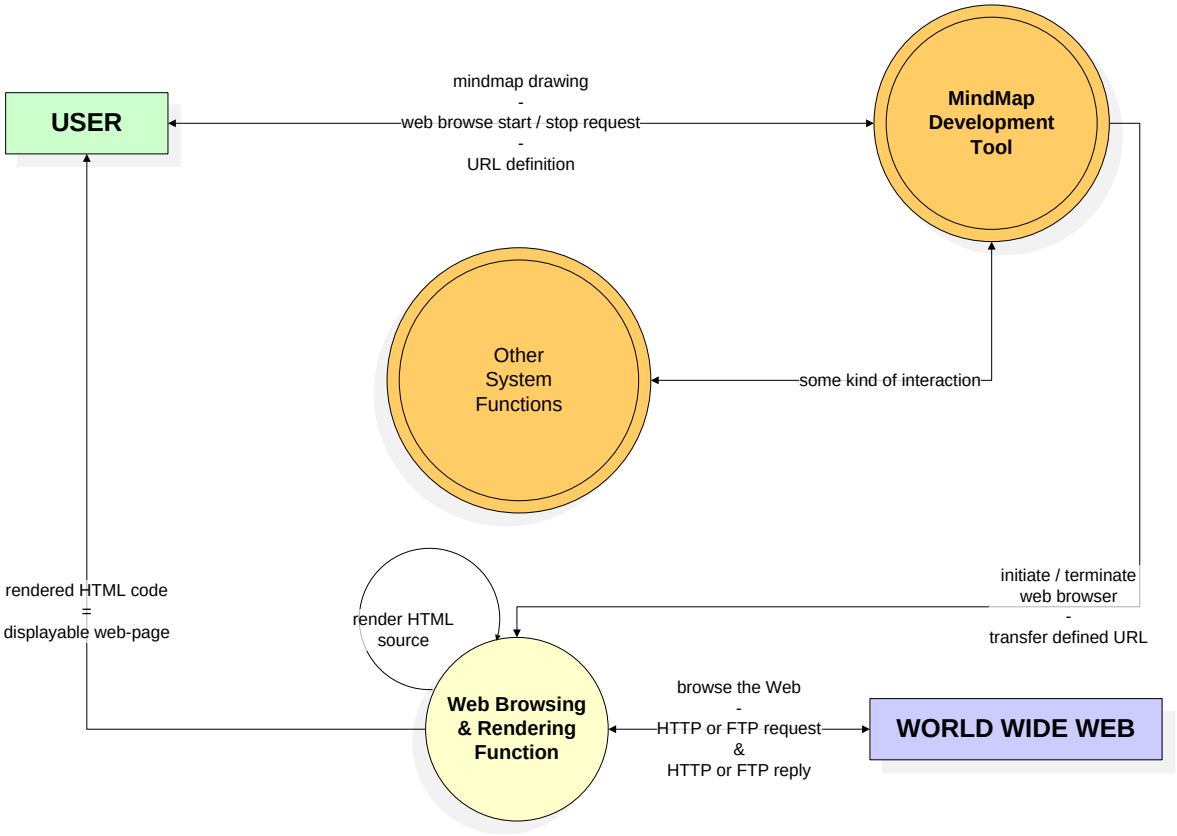
Είτε λοιπόν ο κατάλογος αποτελεσμάτων είναι συγκεντρωτικός (επομένως ένα σύνολο αποτελεσμάτων) είτε διαχωρισμένος (άρα τόσες ομάδες όσες και οι μηχανές αναζήτησης) ο τρόπος βαθμολόγησης του κάθε αποτελέσματος θα είναι ο ίδιος, οπότε οι βαθμοί των διαφόρων αποτελεσμάτων θα μπορούν να συγκριθούν μεταξύ τους και να προκύψει η ζητούμενη κατάταξη. Αναφέρουμε στο σημείο αυτό ακόμα, ότι εφόσον θεωρήσουμε τον κάθε βαθμό ως μία ψήφο και την τελική βαθμολογία ως σύνολο ψήφων, μπορούμε να κατανοήσουμε και τον χρησιμοποιούμενο όρο «δημοκρατικός» αλγόριθμος, καθώς «εκλέγεται» σε καλύτερη θέση το αποτέλεσμα με τις περισσότερες ψήφους.

Επιπροσθέτως, κάθε αποτέλεσμα χρήσιμο θα ήταν να εμφανίζεται με ένα checkbox δίπλα του, το οποίο θα χρησιμεύει στην επιλογή ή όχι του συγκεκριμένου αποτελέσματος. Αρχικά ας είναι επιλεγμένα όλα τα αποτελέσματα, ενώ ο χρήστης πρέπει να μπορεί να ελέγξει περαιτέρω το κάθε ένα από αυτά κάνοντας click επί του link που θα υπάρχει στα στοιχεία του. Αμέσως, το σύστημα τότε θα ανοίγει ένα άλλο εσωτερικό παράθυρο που χρειάζεται να λειτουργεί ως ενσωματωμένος web-browser και να επισκέπτεται την αντίστοιχη ιστοσελίδα.

Σημειώνουμε εδώ, ότι μόνο τα αποτελέσματα με τα σημειωμένα checkboxes δίπλα τους θα θεωρούνται επιλεγμένα. Αυτά πλέον, θα μπορούν να μεταφερθούν με το πάτημα ενός button στο topic του mindmap, στο κείμενο του οποίου θα έχει βασιστεί η αναζήτηση. Η επέκταση του mindmap απαιτούμε να υλοποιείται με την δημιουργία subtopics (κόμβων-παιδιών) υπό το εν λόγω topic, κάθε ένα από τα οποία θα πρέπει να φέρει ως text τον τίτλο ενός result και ως hyperlink την ιστοσελίδα του ίδιου result. Ως αποτέλεσμα θα λαμβάνουμε τον επιλεκτικά εμπλουτισμένο με ενδιαφέρον υλικό από το Internet χάρτη σκέψεων.

3.2.3 Υποσύστημα πλοήγησης στο Διαδίκτυο

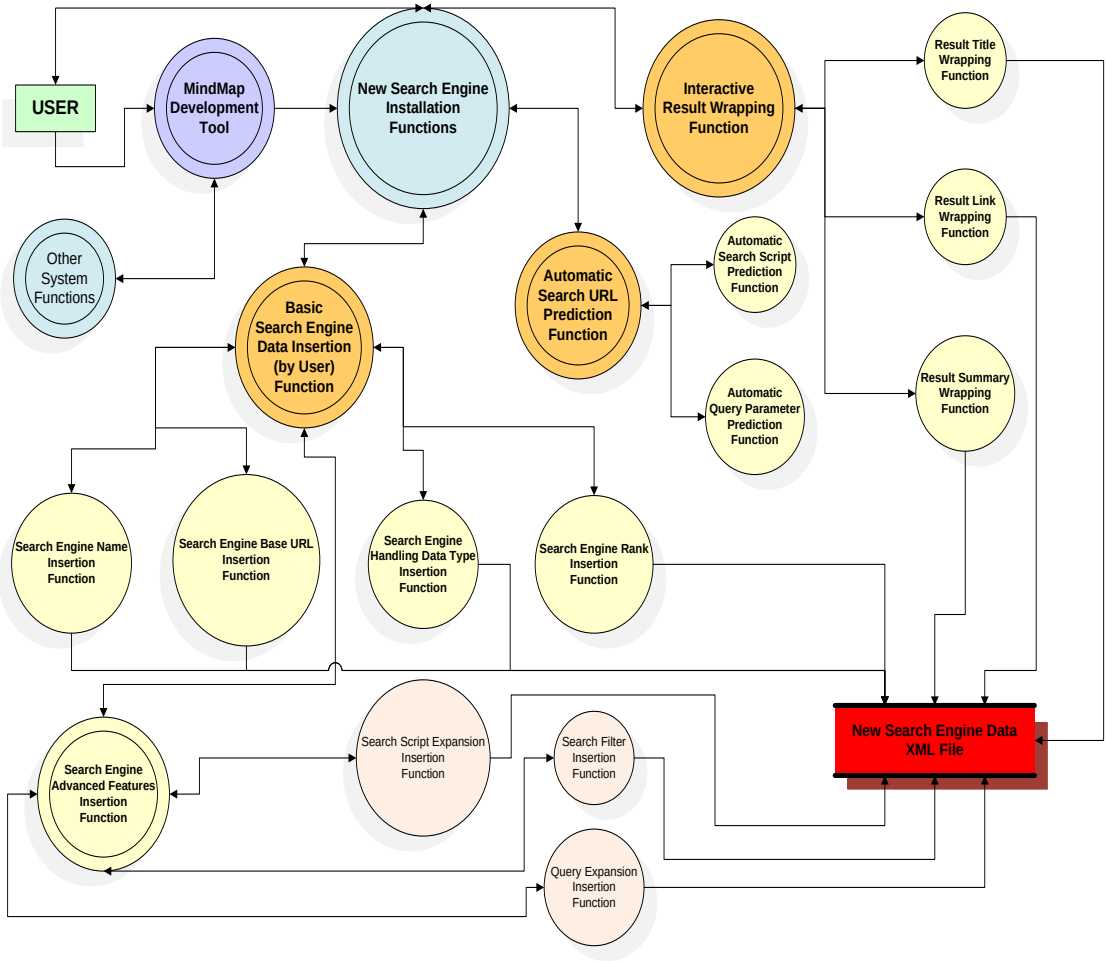
Σε μία προσπάθεια ολοκλήρωσης της διαδικτυακής διαδραστικότητας και προκειμένου να καταστήσουμε την εφαρμογή μας πληρέστερη, θεωρούμε απαραίτητη την ενσωμάτωση ενός φυλλομετρητή ιστοσελίδων (web-browser) σε αυτήν. Τις απαιτήσεις μας ικανοποιεί απόλυτα το εργαλείο *WebRenderer*, ενώ αρχικά είχαμε καταφύγει σε ένα άλλο, τον *Lobo Browser*, ο οποίος όμως κάλυπτε μόνο εν μέρει τις ανάγκες μας. Ο WebRenderer είναι ένας πλήρως ενσωματώσιμος browser, ο οποίος έχει οικοδομηθεί πάνω στην Mozilla τεχνολογία (την ίδια ακριβώς όπως στον Firefox 2.0), αποτελεί δηλαδή ουσιαστικά έναν κλώνο του Mozilla Firefox. Στα μοναδικά χαρακτηριστικά του συγκαταλέγονται η ικανότητά του να υποστηρίζει τα ευρέως διαδεδομένα διαδικτυακά πρότυπα (όπως HTML 4.01, CSS 1 & 2, HTTP, SSL, Java Applets, JavaScript, XHTML, XML, XSL, XSLT, WC3 DOM κλπ), ο πλήρης έλεγχος της πλοήγησης, η ταχύτητα, η ασφάλεια (security), η ευστάθεια (stability) και η ευρωστία (robustness). Δίνουμε αμέσως παρακάτω το διάγραμμα ροής δεδομένων (data flow diagram) του συζητούμενου υποσυστήματος:



Σχήμα 3.4 – Διάγραμμα ροής δεδομένων του υποσυστήματος πλοήγησης στο Διαδίκτυο

3.2.4 Υποσύστημα εγκατάστασης νέων μηχανών αναζήτησης

Η εφαρμογή μας οφείλει να υποστηρίζει την ενσωμάτωση νέων μηχανών αναζήτησης στο σύνολο των ήδη προαναφερθέντων, προκειμένου και αυτές να μπορούν να χρησιμοποιηθούν από το υποσύστημα της προηγμένης μετα-αναζήτησης πληροφοριών στον Παγκόσμιο Ιστό. Όπως έχει γραφθεί παραπάνω, οι μηχανές αναζήτησης θα υλοποιούνται στο σύστημά μας είτε με την μορφή ενός προγράμματος (API-based search engines), είτε μέσω της τεχνικής του screen-scraping, υποβοηθούμενης από την διάθεση ορισμένων στοιχείων από τον ίδιο τον χρήστη (wrapped search engines). Το πρώτο είδος απαιτεί ο χρήστης να έχει και γνώσεις προγραμματισμού ηλεκτρονικών υπολογιστών, δηλαδή αποτελεί μία ιδιαίτερα εξειδικευμένη μέθοδο, ενώ το δεύτερο είδος θεωρεί πως το άτομο είναι γνώστης απλής χρήσης PCs. Έτσι, ο συνιστώμενος τρόπος εγκατάστασης μίας καινούριας μηχανής αναζήτησης είναι ο δεύτερος, εκτός εξαιρετικών περιπτώσεων. Ακολούθως παραθέτουμε το διάγραμμα ροής δεδομένων (data flow diagram) του εν λόγω υποσυστήματος της εφαρμογής μας, ενώ αμέσως παρακάτω δίνουμε τις υποενότητες με την πλήρη σχετική ανάλυση των διαφόρων βημάτων που συνιστούν την διαδικασία ενσωμάτωσης νέων μηχανών.



Σχήμα 3.5 – Διάγραμμα ροής δεδομένων

του υποσυστήματος εγκατάστασης νέων μηχανών αναζήτησης

3.2.4.1 Στάδιο εισαγωγής δεδομένων από τον χρήστη

Με το πάτημα απλώς ενός button, επιθυμούμε να εμφανίζεται ένας φιλικός προς τον χρήστη και εύκολος στην χρήση wizard, ο οποίος θα απαιτεί την εισαγωγή ορισμένων θεμελιωδών πληροφοριών που θα αφορούν την προς εγκατάσταση μηχανή. Αυτές οι πληροφορίες συνοψίζονται στην ονομασία της καινούριας μηχανής αναζήτησης (π.χ. AltaVista), τον βασικό διαδικτυακό σύνδεσμο αναζήτησης (όπως http://search.lycos.com/), τον τύπο δεδομένων για τον οποίο πρόκειται να χρησιμοποιηθεί η εν λόγω νέα μηχανή (web page search / blog search / document search / paper search / map search / image search / audio search / video search) και την βαθμολογία της μηχανής για τον συγκεκριμένο τύπο δεδομένων (με κλίμακα από 1 έως 5 και χαρακτηρισμούς άριστη, καλή, μέτρια, κακή και πτωχή).

Προκειμένου να προσδοθεί σε αυτήν την μηχανή αναζήτησης προηγμένη λειτουργικότητα, ο χρήστης πρέπει να καλείται να εισάγει πρόσθετα δεδομένα, τα οποία όμως θα είναι προαιρετικά, ενώ πρέπει να αναφερθεί ότι μάλλον θα απαιτούν κάποιες πιο ιδιαίτερες

(τεχνικού είδους) γνώσεις από τον χρήστη. Επεξηγώντας, υπενθυμίζουμε ότι κάθε ένας τύπος δεδομένων θέλουμε να υποστηρίζει διαφορετικά φίλτρα αποτελεσμάτων (π.χ. author-, date-, publication-, subject-based search για papers). Έτσι, για κάθε έναν από τους περιορισμούς αυτούς και εφόσον θα είναι επιθυμητή η ενσωμάτωσή τους στα πλαίσια της αναζήτησης με την εν λόγω μηχανή, θα πρέπει ο ίδιος ο χρήστης να δώσει είτε την κατάλληλη επέκταση του search script, είτε ένα search filter, είτε την απαιτούμενη query expansion.

Πιο αναλυτικά, ως *search script* καλείται εκείνο το τμήμα του search URL, το οποίο διαχωρίζεται από το βασικό URL της μηχανής αναζήτησης (π.χ. έχοντας το search URL http://www.google.com/search?hl=en&q=*keyword*, ως βασικό URL θεωρούμε το http://www.google.com/, ενώ ως search script το /search?hl=en&q=*keyword*). Επομένως, χαρακτηριστικό παράδειγμα μίας search script expansion για προσθήκη λειτουργικότητας αναζήτησης σε ιστοσελίδες γενικού περιεχομένου με το Google βάσει της ημερομηνίας είναι η ακόλουθη: &as_qdr=, ενώ σε αυτή την περίπτωση το συνολικό search URL θα ήταν το εξής: http://www.google.com/search?hl=en&q=*keyword*&as_qdr=*date*.

Επιπροσθέτως, ως search filtering ορίζουμε την δυνατότητα ορισμένων μηχανών αναζήτησης να δέχονται δίπλα στις λέξεις-κλειδιά άλλες λέξεις, των οποίων προηγούνται ειδικοί τελεστές, κάθε ένας από τους οποίους μπορεί να εξειδικεύσει τα επιστρεφόμενα αποτελέσματα, περιορίζοντας την αρχική λίστα αποτελεσμάτων σύμφωνα με το εκάστοτε εφαρμοζόμενο φίλτρο. Παραδείγματος χάριν ένα search filter για αναζήτηση general web pages με το Google και μόνο σε domain με κατάληξη .edu, είναι της μορφής: ***site***:*.edu* .

Ακόμα, ως query expansion εννοούμε την αυτόματη παράθεση κατάλληλων βοηθητικών keywords δίπλα στο δοθέν (ή τα δοθέντα) keyword (ή keywords), αποσκοπώντας με αυτόν τον απλό τρόπο στην στοιχειώδη εξειδίκευση των αποτελεσμάτων. Επί παραδείγματι, στην περίπτωση αναζήτησης ιστοσελίδων γενικών περιεχομένου με το Google και με επιθυμητή γλώσσα αποτελεσμάτων τα γερμανικά, μπορούμε να παραθέσουμε πέρα από την λέξη κλειδί *keyword*, που έχει προσδιοριστεί από τον ίδιο τον χρήστη, και την κατατοπιστική λέξη ***deutsch***.

Ως γενικό κανόνα μπορούμε να πούμε ότι οι search script expansions προτιμούνται των άλλων δύο μεθόδων, ενώ τα search filters ενδείκνυνται έναντι των query expansions. Υπενθυμίζουμε στο σημείο αυτό, πως όλα τα τελευταία είναι βεβαίως προαιρετικά, όμως καθίστανται αναγκαία εφόσον επιθυμούμε την ενσωμάτωση στοιχείων προηγμένης λειτουργίας στην εισαγόμενη μηχανή αναζήτησης.

Σημειώνουμε τέλος, ότι το σύστημά μας οφείλει να προβλέπει κάθε δυνατή περίπτωση συμπεριφοράς του χρήστη και να αντιμετωπίζει αυτή την συμπεριφορά κατάλληλα, ακόμα και αν αυτή δεν κινείται εντός του αναμενόμενου πλαισίου (δηλαδή περιπτώσεις παραλείψεων στοιχείων ή εισαγωγής εσφαλμένης μορφής δεδομένων, όπως συμβολοσειρά

εκεί που έπρεπε να εισαχθεί αριθμός, κ.ά.), ενώ επίσης το εν λόγω στάδιο της εγκατάστασης μίας καινούριας μηχανής αναζήτησης καλό θα ήταν να ολοκληρώνεται με την παρουσίαση στον χρήστη όλων των προσφάτως εισηγμένων σχετικών στοιχείων.

3.2.4.2 Στάδιο αυτοματοποιημένης διαδικασίας πρόγνωσης του search URL

Στην προηγούμενη ενότητα αναφερθήκαμε στους όρους *search URL* και *search script*. Η εφαρμογή μας φιλοδοξούμε να παρέχει την δυνατότητα της εν μέρει πρόβλεψης του search URL, και πιο συγκεκριμένα να «μαντεύει» το search script και την *query parameter*. Στο στάδιο εισαγωγής δεδομένων για την μηχανή αναζήτησης, μεταξύ των άλλων ο χρήστης θα πρέπει να προσφέρει και το βασικό URL της εν λόγω μηχανής (το οποίο εμείς καλούμε *base URL*). Το base URL σε συνδυασμό με το search script συνιστούν το search URL, ενώ η query parameter αποτελεί μία μεταβλητή του search script. Αναλυτικότερα, query parameter είναι εκείνη η μεταβλητή, της οποίας έπεται (και στην οποία «ανατίθεται») η ενδιαφέρουσα λέξη-κλειδί. Για να γίνουν πιο κατανοητά τα τελευταία, παραθέτουμε το εξής παράδειγμα: Έστω το URL http://www.google.com/search?hl=en&q=*keyword*, με το οποίο πραγματοποιείται η αναζήτηση σε ιστοσελίδες γενικού περιεχομένου με την διάσημη μηχανή αναζήτησης Google, επομένως μπορούμε να το χαρακτηρίσουμε και ως search URL. Σε αυτό, ως base URL θεωρείται το http://www.google.com/, ως search script το /search?hl=en&q=*keyword*, ενώ ως query parameter η q. Έτσι, στην εφαρμογή μας, το search URL της εκάστοτε προς εγκατάσταση μηχανής αναζήτησης συντίθεται μέσω του base URL που θα δίνει ο χρήστης και του search script (το οποίο περιλαμβάνει την query parameter) που θα προβλέπει το ίδιο το σύστημα.

Αμέσως μετά το στάδιο εισαγωγής πληροφοριών για την μηχανή αναζήτησης από τον χρήστη, επιθυμούμε να ανοίγει αυτομάτως ο ενσωματωμένος εσωτερικός φυλλομετρητής Ιστού (WebRenderer), για τον οποίο έχει γίνει λόγος παραπάνω στο ίδιο κεφάλαιο, και αυτός να επισκέπτεται το δοθέν base URL. Η εν λόγω πρόγνωση του search script και της query parameter θα θέλαμε να είναι δυνατή απλώς με το πάτημα ενός button. Πιο συγκεκριμένα, το συζητούμενο υποσύστημα οφείλει να προβάλλει την λίστα με τα ευρεθέντα scripts και τις αντίστοιχες parameters, από την οποία θα καλείται ο χρήστης να επιλέξει εκείνο το script και εκείνη την parameter, που θα αποτελέσουν το search script, και μαζί με το base URL θα οδηγήσουν στην σύνθεση του search URL, με την γνώση του οποίου θα μπορεί να καθίσταται ικανό το αρμόδιο υποσύστημα υλοποίησης των διαδικτυακών αναζητήσεων να θέτει queries με το εκάστοτε keyword στην εισηγμένη (πλέον) search engine. Στην εν λόγω λίστα χρήσιμο θα ήταν να σημειώνεται ως συνιστώμενο (recommended) το πιο κατάλληλο ζεύγος script-parameter (το οποίο να είναι και το ζητούμενο στην πλειονότητα των περιπτώσεων). Ο χρήστης βέβαια θα είναι σε θέση είτε να επιλέξει κάποιο από τα παρεχόμενα ζεύγη (το

ενδεικνυόμενο ή άλλο), είτε να εισάγει το δικό του (εφόσον θα κατέχει ιδιαίτερες γνώσεις). Με αυτή την μέθοδο μπορεί να γίνει δυνατός ο πλήρης προσδιορισμός του URL διαμέσω του οποίου θα δίνεται η ευχέρεια πραγματοποίησης ερωτήσεων, βάσει προσδιορισμένων λέξεων-κλειδιών προς συγκεκριμένες μηχανές αναζήτησης, ενώ παράλληλα θα προσφέρονται ως απόκριση τα αντίστοιχα αποτελέσματα της αναζήτησης.

3.2.4.3 Στάδιο εκμάθησης ανάγνωσης των αποτελεσμάτων αναζήτησης στην μηχανή

Το συγκεκριμένο στάδιο αποσκοπούμε να αποτελεί μία διαδραστική διαδικασία, η κατάληξη της οποίας θα είναι η απόκτηση της ικανότητας εκ μέρους της εφαρμογής να εξάγει την χρήσιμη πληροφορία από μία ιστοσελίδα αποτελεσμάτων αναζήτησης με έναν συστηματικό τρόπο. Ένα σύνηθες αποτέλεσμα περιλαμβάνει γενικά έναν τίτλο (title), έναν διαδικτυακό σύνδεσμο (link ή URL) και μία περιγραφή (description ή summary). Για κάθε ένα από αυτά τα τρία συστατικά στοιχεία, ο χρήστης θα πρέπει να βοηθήσει την μηχανή στην εκμάθηση της ανάγνωσής τους (machine learning). Αυτό μπορεί να επιτευχθεί αξιοποιώντας την πολυσυζητημένη (στα πλαίσια της παρούσας διπλωματικής εργασίας) τεχνική του screen-scraping.

Μετά την πρόγνωση του search URL από το σύστημά μας, στον ήδη ανοιγμένο εσωτερικό web browser είναι επιθυμητό να πραγματοποιείται αυτομάτως μία αναζήτηση βάσει προκαθορισμένης λέξης-κλειδιού και η αντίστοιχη ιστοσελίδα με τα αποτελέσματα να εμφανίζεται στην οθόνη. Εκείνο που απλώς θα απαιτείται εν συνεχεία από τον χρήστη είναι διαδοχικά η επιλογή του τίτλου, του διαδικτυακού συνδέσμου και της περιγραφής ενός αποτελέσματος.

Λέγοντας «επιλογή», εννοούμε το μαρκάρισμα με το ποντίκι (text highlighting via mouse selection), ενώ μετά από κάθε τέτοια επιλογή θα πρέπει να ακολουθεί η δήλωση του τρέχοντος είδους (δηλαδή title / URL / summary) στοιχείου προς το σύστημα με το πάτημα ενός από τα τρία αντίστοιχα buttons. Σημειώνουμε εδώ ότι θα θέλαμε να μην είναι απαραίτητη η επιλογή ενός ακέραιου στοιχείου από ένα αποτέλεσμα, αλλά να επαρκεί μόνο ένα τμήμα του, υπό την προϋπόθεση ότι αυτό θα είναι αρκετά μεγάλο, έτσι ώστε να μην επανεμφανίζεται και σε διαφορετικό μέρος της ιστοσελίδας και να αποφεύγονται κατά αυτόν τον τρόπο οι συγχύσεις.

Επιπροσθέτως, πρέπει να αναφερθεί ότι στην ουσία μόνο ο τίτλος και η περιγραφή ενός αποτελέσματος συγκαταλέγονται στα «υποχρεωτικά» προς επιλογή είδη, αφού οι σύνδεσμοι μπορούν να εκμαιεύονται από τους ίδιους τους τίτλους, καθώς στην συντριπτική πλειοψηφία των περιπτώσεων οι τελευταίοι συμπεριλαμβάνουν αντίστοιχο διαδικτυακό υπερσύνδεσμο (hyperlink). Έτσι, το σύστημά μας θα παρέχει την δυνατότητα (μέσω δύο checkboxes) να

καθορίσει ο χρήστης αν επιθυμεί την αυτόματη πρόβλεψη των links ή αν θέλει να τα υποδείξει ο ίδιος μέσω της μεθόδου του wrapping, ή και τα δύο.

Όταν θα δηλώνεται μία επιλογή μέσω του αντίστοιχου από τα τρία buttons, η εφαρμογή θα αποπειράται να διαβάσει τα όμοια στοιχεία της ιστοσελίδας και θα επαναφορτώνει την σελίδα των αποτελεσμάτων, εμφανίζοντας με χρωματισμένο φόντο τα στοιχεία αυτά (για κάθε μία από τις τρεις επιλογές θα πρέπει να χρησιμοποιείται διαφορετικό χρώμα για το υπόβαθρο). Εφόσον ο χρήστης μείνει ικανοποιημένος από την προσπάθεια ανάγνωσης του εκάστοτε στοιχείου των αποτελεσμάτων από την μηχανή, θα συνεχίζει με το επόμενο από τα τρία στοιχεία.

Σε αντίθετη περίπτωση, θα χρειάζεται να προσπαθήσει να κατευθύνει την μηχανή περαιτέρω και να την βοηθήσει προκειμένου είτε αυτή να περιορίσει τα θεωρούμενα ως ορθά στοιχεία, αφαιρώντας πρόσθετα στοιχεία που εσφαλμένα θα έχουν συμπεριληφθεί, είτε να επικεντρωθεί σε πιο ομοειδή στοιχεία. Για να επιτευχθούν αυτά, ο χρήστης θα πρέπει να μπορεί να δώσει (μέσω της επιλογής ενός checkbox) εντολή για περιορισμό στην ακριβή θέση του επιθυμητού στοιχείου εντός των αποτελεσμάτων (π.χ. όταν το σύστημα θα αντιλαμβάνεται ως στοιχεία ευρύτερες ενότητες) ή να διαλέξει μία από τις προσφερόμενες (μέσω μίας drop-down list / ενός combo-box) *κανονικές εκφράσεις* (*regular expressions*) για εφαρμογή.

Τελικά, ο χρήστης θα καλείται να επιλέξει ανάμεσα στην επικύρωση του συγκεκριμένου τμήματος της διαδικασίας, την επανάληψή του με τους καινούριους περιορισμούς και την ακύρωσή του. Επίσης, ανά πάσα στιγμή ο χρήστης θα είναι σε θέση να διακόψει την συνολική διαδικασία εκμάθησης ανάγνωσης των αποτελεσμάτων στην μηχανή (για να την επαναλάβει ενδεχομένως από την αρχή), ενώ μόνο αν όλες οι φάσεις της διαδικασίας διεκπεραιωθούν με επιτυχία θα πρέπει αυτός να δώσει το πράσινο φως προκειμένου να συντελεστεί η εισαγωγή της νέας μηχανής αναζήτησης στην λίστα των ήδη χρησιμοποιούμενων.

Αναφέρουμε στο σημείο αυτό, ότι κατά την διάρκεια της εγκατάστασης, εκείνο που θα συντελείται «στο παρασκήνιο» είναι η καταγραφή συγκεκριμένων ωφέλιμων δεδομένων για την εισαγόμενη μηχανή σε ένα λιτό, σαφές και προκαθορισμένου τύπου XML αρχείο, το οποίο θα επαρκεί για την διάθεση όλων των απαραίτητων πληροφοριών, έτσι ώστε η εγκατεστημένη μηχανή να είναι πλήρως λειτουργική και να ενσωματώνεται αρμονικά στο συνολικό σύστημα.

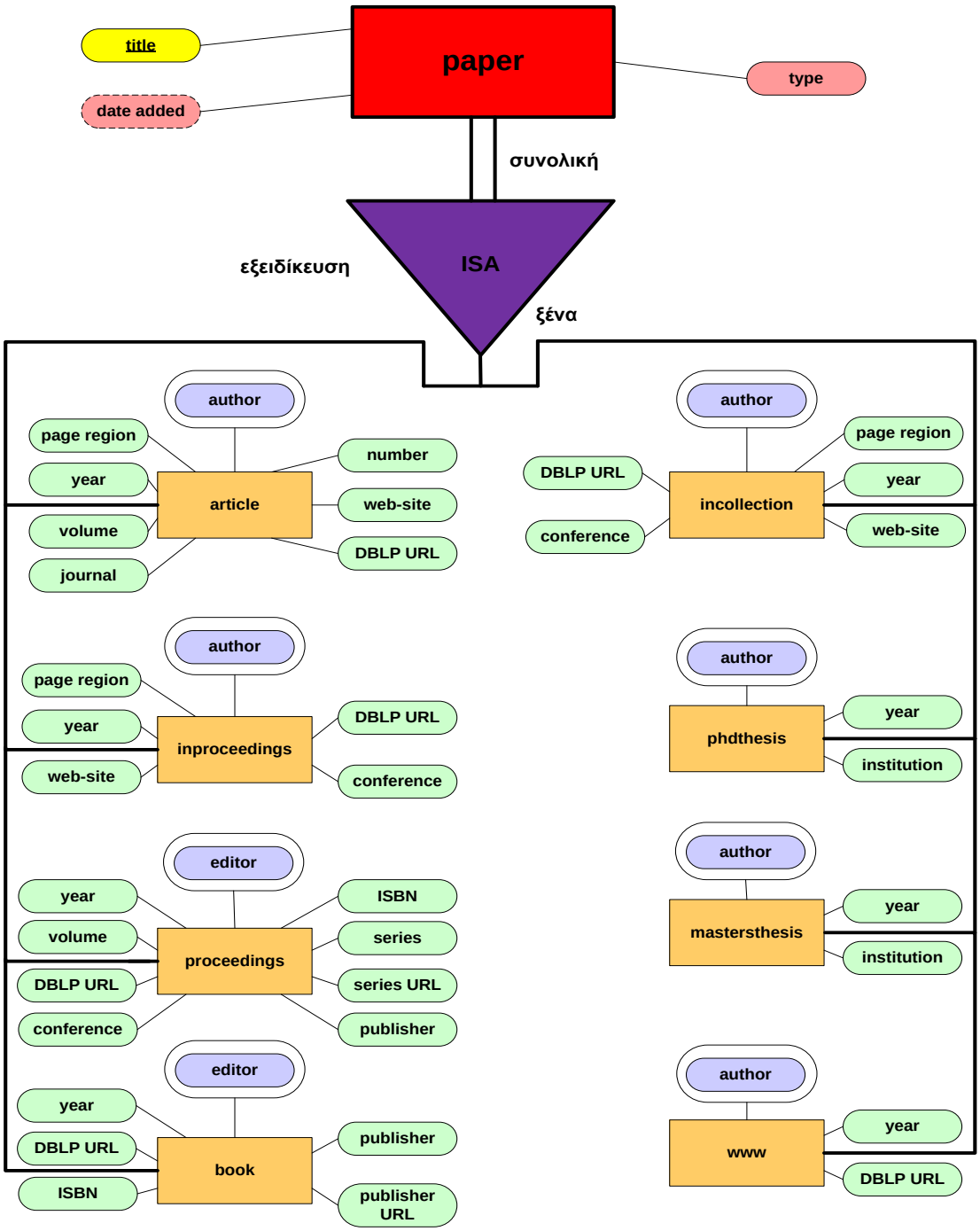
Πρέπει επιπλέον να τονισθεί, ότι η τεχνική του screen-scraping αναμένεται να λειτουργεί ικανοποιητικά μόνο σε περίπτωση που οι επιστρεφόμενες σελίδες αποτελεσμάτων από την εγκαθιστάμενη μηχανή αναζήτησης υλοποιούνται με καλοδομημένο HTML κώδικα. Κανείς μπορεί να αντιληφθεί την σημαντικότητα της τελευταίας παρατήρησης, καθώς από την

επιτυχία της result wrapping process διαφαίνεται να εξαρτάται και εκείνη της συνολικής διαδικασίας εγκατάστασης. Αξιοσημείωτη είναι επίσης η πρόκληση της απρόσκοπτης ενσωμάτωσης δημοφιλών μηχανών αναζήτησης με την συζητούμενη μέθοδο στο σύστημά μας (π.χ. Google, Yahoo, Microsoft Live, Gigablast κ.ά.).

Συνοψίζοντας, η εγκατάσταση μίας νέας μηχανής αναζήτησης στην εφαρμογή μας θα αφορά στην συλλογή των σχετικών αναγκαίων πληροφοριών. Ορισμένες από τις πληροφορίες αυτές θα εισάγονται από χρήστη (όπως η ονομασία, το base URL, ο σχετιζόμενος τύπος δεδομένων, η αντίστοιχη βαθμολογία, η απόφαση για συμπερίληψη στοιχείων προηγμένης αναζήτησης με την παράλληλη διάθεση της σχετικής script expansion ή / και του search filter ή / και της query expansion). Άλλες θα προβλέπονται από ίδιο το σύστημα (όπως το search script και η query parameter, που μαζί με το base URL απαρτίζουν το πολύτιμο search URL). Τέλος, το πιο ενδιαφέρον και φιλόδοξο στάδιο είναι εκείνο της διαδραστικής εκμάθησης της εξόρυξης των στοιχείων των αποτελεσμάτων στην μηχανή, το οποίο θα εκμεταλλεύεται την τεχνική του screen-scraping και την ύπαρξη μίας νοητής δενδρικής δομής πίσω από κάθε HTML σελίδα. Αναλογιζόμενοι την δυσκολία του εγχειρήματος, θα είμαστε απόλυτα ικανοποιημένοι, εφόσον το σύστημα συμπεριφέρεται αρκετά καλά σε μεγάλη πληθώρα περιπτώσεων μηχανών αναζήτησης που θα κληθεί να αντιμετωπίσει.

3.3 Μοντέλο Οντοτήτων – Συσχετίσεων της ΒΔ του DBLP

Όπως έχει ήδη προαναφερθεί, η αναζήτηση πληροφοριών στον Παγκόσμιο Ιστό που σχετίζεται με επιστημονικές δημοσιεύσεις (papers) αξιοποιεί την έγκριτη Βάση Δεδομένων του DBLP. Το E-R διάγραμμα (entity-relationship diagram / διάγραμμα οντοτήτων-συσχετίσεων) της εν λόγω βάσης δεδομένων παρατίθεται κάτωθι:



Σχήμα 3.6 – Διάγραμμα Οντοτήτων-Συσχετίσεων της ΒΔ του DBLP

4

Σχεδίαση Συστήματος

Στο παρόν κεφάλαιο θα παραθέσουμε την σχεδίαση της εφαρμογής μας. Αναλυτικότερα, θα περιγράψουμε την αρχιτεκτονική του συστήματος, θα παρουσιάσουμε την λειτουργία των επιμέρους τμημάτων του, θα δούμε σε μεγαλύτερο βάθος τα όσα αφορούν την τηρούμενη βάση δεδομένων με τις επιστημονικές δημοσιεύσεις του DBLP και θα αναδείξουμε τις χρησιμοποιούμενες κωδικοποιήσεις αρχείων σαν εκείνη του δημιουργούμενου XML αρχείου για κάθε νεοεγκαθιστάμενη μηχανή αναζήτησης.

4.1 Αρχιτεκτονική

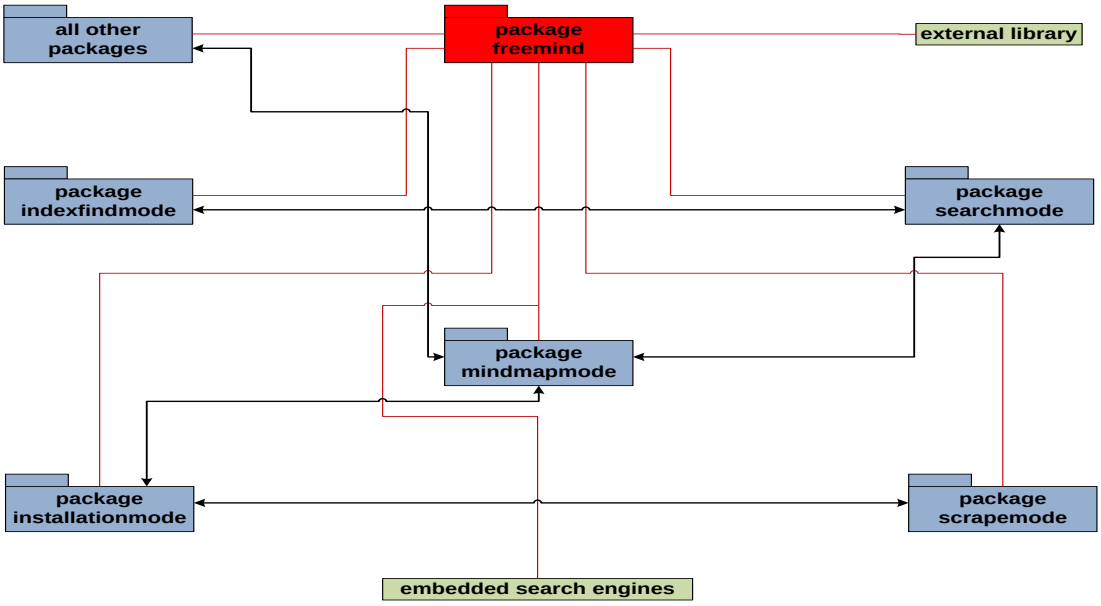
Σε προηγούμενο κεφάλαιο έχουμε καταστήσει σαφείς τους λόγους για τους οποίους επιλέξαμε το *FreeMind* ως κύριο εργαλείο ανάπτυξης mindmaps. Δύο επιπρόσθετοι λόγοι (οι οποίοι διαδραμάτισαν αποφασιστικό ρόλο στην λήψη της τελικής μας απόφασης) είναι οι ακόλουθοι:

1. Πρόκειται για λογισμικό ανοικτού κώδικα (open source software), επομένως πέρα από την εφαρμογή αυτή καθ' αυτήν μας είναι διαθέσιμος και ο κώδικάς της.
2. Το FreeMind έχει αναπτυχθεί με μία από τις σημαντικότερες και δημοφιλέστερες γλώσσες αντικειμενοστραφούς προγραμματισμού, την *Java*.

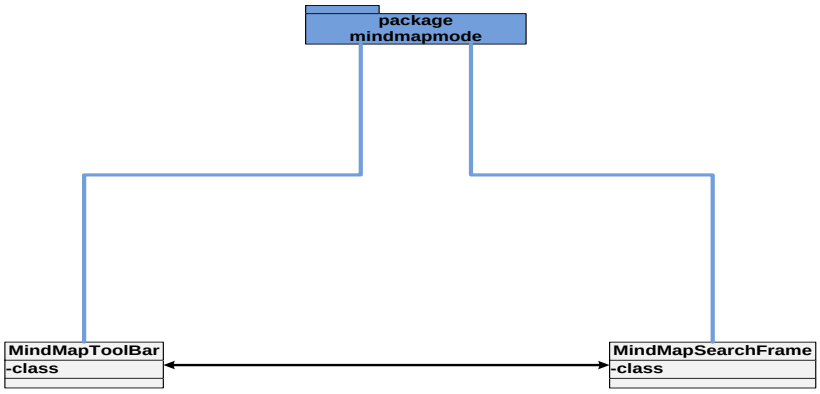
Πέρα από την συγκεκριμένη γλώσσα, η οποία είναι η κυριότερη που χρησιμοποιήσαμε κατά την συγγραφή του δικού μας κώδικα, έγινε χρήση κι άλλων, όπως οι HTML (HyperText

Markup Language), JavaScript, XML (eXtensible Markup Language), XSLT (eXtensible Stylesheet Language Transformations), DTD (Document Type Definition) και RDF (Resource Description Framework) / OWL Full (Web Ontology Language). Εφόσον η γλώσσα με την οποία αναπτύξαμε την εφαρμογή μας είναι η Java, δηλαδή μία object-oriented programming language, τα τμήματα από τα οποία έχει οικοδομηθεί ο κώδικάς μας είναι ουσιαστικά οι διάφορες κλάσεις.

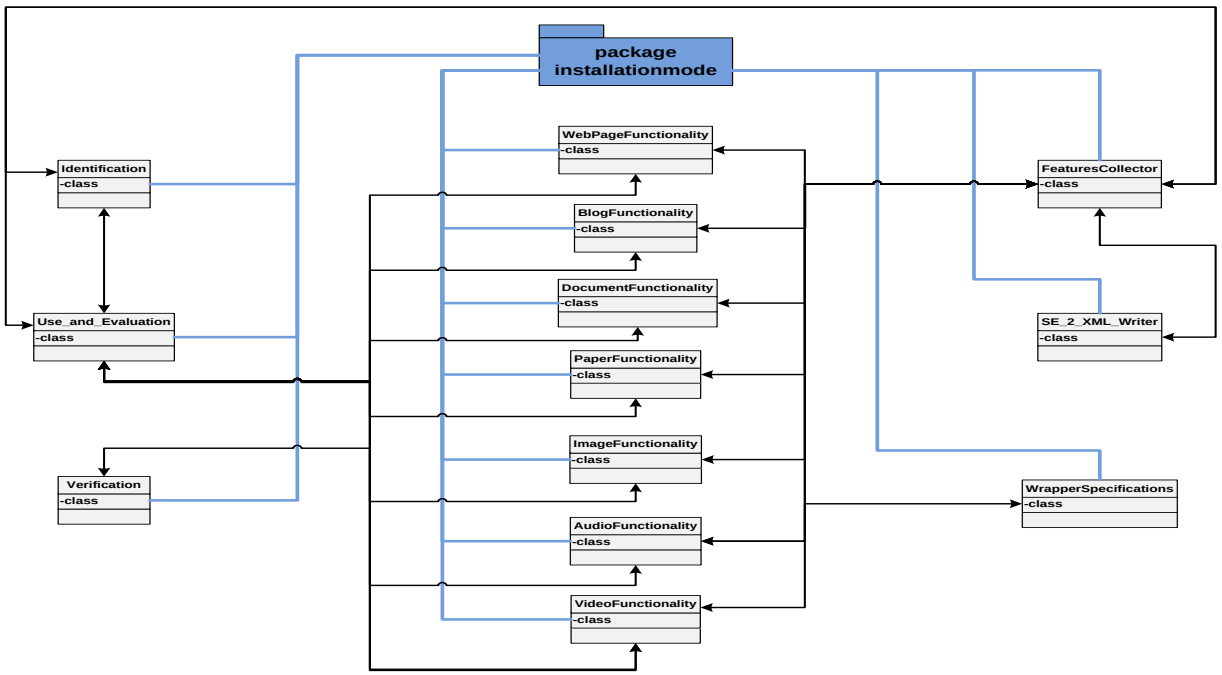
Στο σημείο αυτό παραθέτουμε ένα γενικό block διάγραμμα των πακέτων κλάσεων, καθώς και τα block διαγράμματα των κλάσεων για κάθε πακέτο, όπου φαίνονται ποιες είναι οι κλάσεις, πού υπάγονται και πώς επικοινωνούν μεταξύ τους.



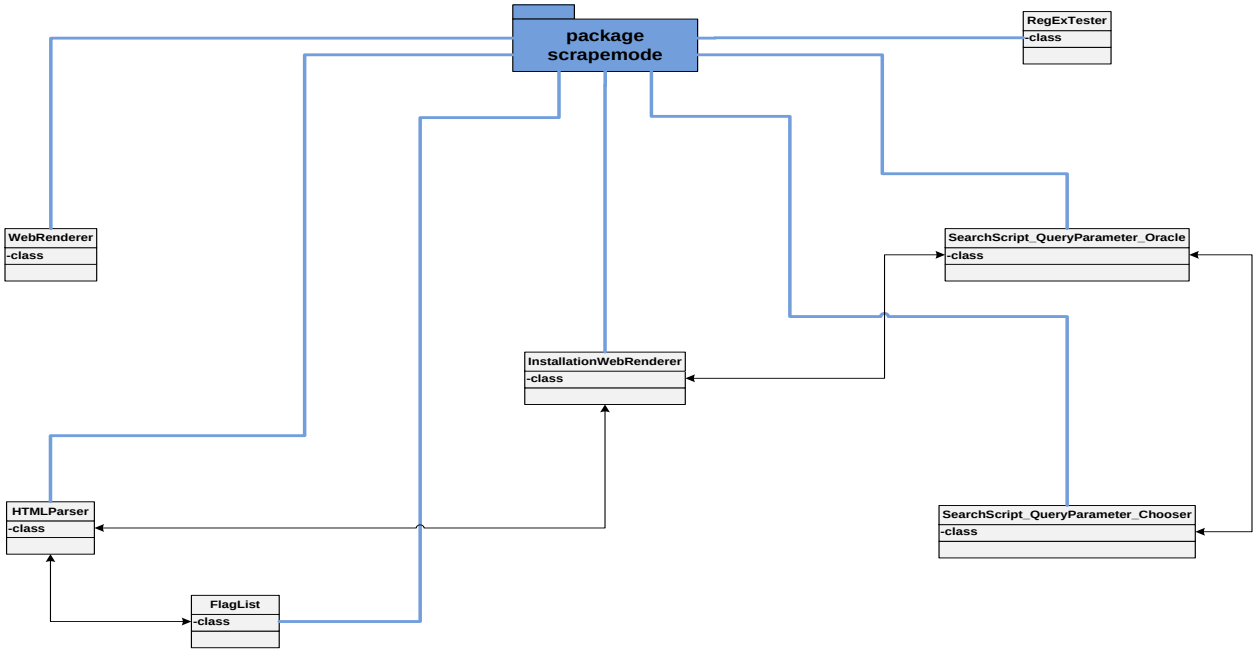
Σχήμα 4.1 – Block διάγραμμα των πακέτων κλάσεων της εφαρμογής



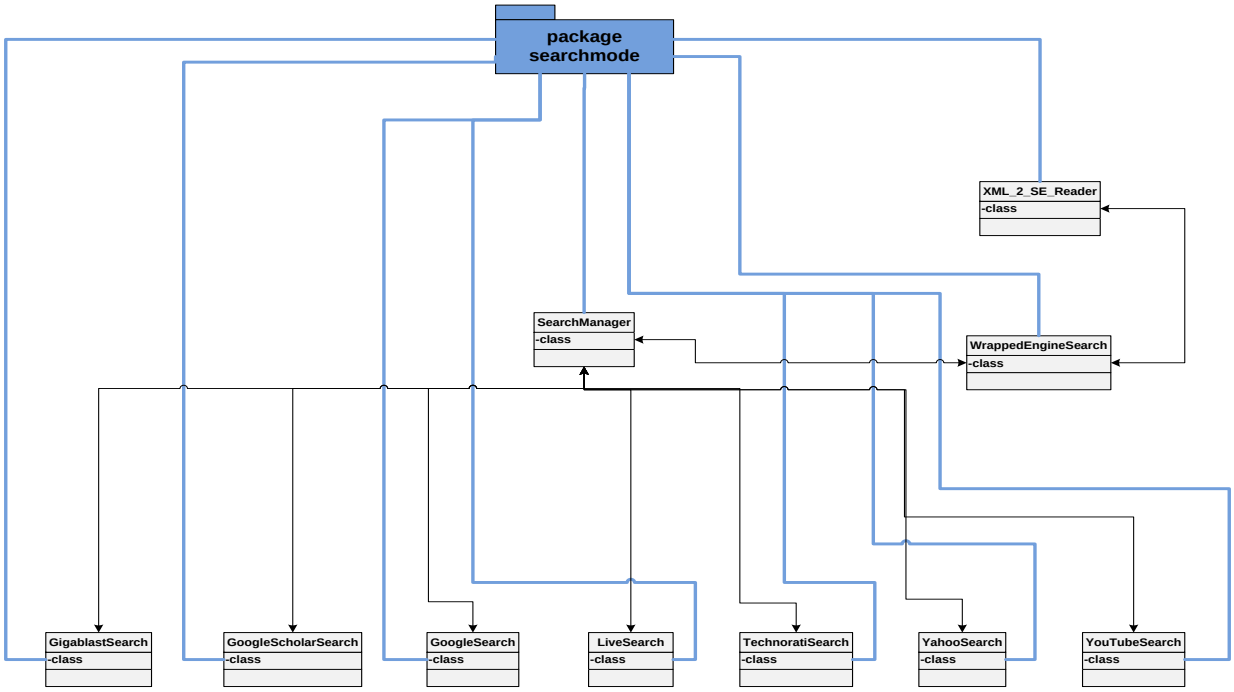
Σχήμα 4.2 – Block διάγραμμα κλάσεων για το πακέτο mindmapmode



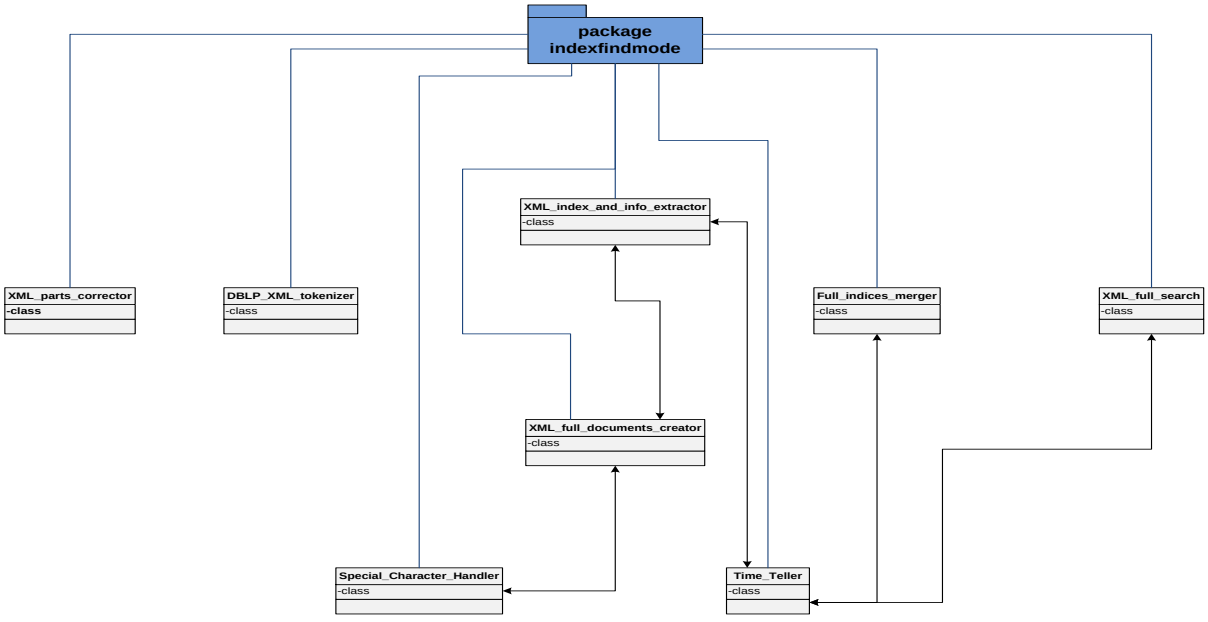
Σχήμα 4.3 – Block διάγραμμα κλάσεων για το πακέτο installationmode



Σχήμα 4.4 – Block διάγραμμα κλάσεων για το πακέτο scrapemode



Σχήμα 4.5 – Block διάγραμμα κλάσεων για το πακέτο searchmode



Σχήμα 4.6 – Block διάγραμμα κλάσεων για το πακέτο indexfindmode

Στην συνέχεια θα απαριθμήσουμε τις νέες κλάσεις που φτιάξαμε ή τις ήδη υπάρχουσες που τροποποιήσαμε και παράλληλα θα δίνουμε μία σύντομη περιγραφή τους, ενώ η αναλυτική τους παρουσίαση θα γίνει στην επόμενη ενότητα.

4.1.1 Το περιβάλλον για το mind-mapping και η διαπροσωπεία του web-searching

Η κλάση MindMapToolBar προϋπήρχε στο FreeMind και είναι υπεύθυνη για την εμφάνιση της μπάρας με τα βασικά buttons λειτουργίας, στα οποία κι εμείς προσθέσαμε τα δικά μας. Μία άλλη κλάση είναι η WebRenderer, με την οποία υλοποιείται ο εσωτερικός ενσωματωμένος κλώνος του Mozilla Firefox. Επιπροσθέτως, η κλάση MindMapSearchFrame είναι από τις σημαντικότερες, καθώς αφορά το παράθυρο που υλοποιεί την διεπαφή (μεταξύ χρήστη και μηχανής) για την προηγμένη διαδικτυακή μετα-αναζήτηση.

4.1.2 Η διαδικασία εγκατάστασης μίας καινούριας μηχανής αναζήτησης

Εκτός αυτών, η Identification είναι η πρώτη από μία σειρά κλάσεων που σχετίζονται με την εγκατάσταση νέων μηχανών αναζήτησης στο σύστημά μας και κατασκευάζει την πρώτη καρτέλα του installation wizard που αναμένει την εισαγωγή της ονομασίας και του base URL της καινούριας μηχανής από τον χρήστη. Η δεύτερη καρτέλα του ίδιου wizard υλοποιείται με την βοήθεια της κλάσης Use_and_Evaluation και είναι υπεύθυνη για την λήψη του χρησιμοποιούμενου τύπου δεδομένων από την εν λόγω μηχανή αναζήτησης, καθώς και της βαθμολογία της. Η τρίτη καρτέλα έχει να κάνει με την ενδεχόμενη εισαγωγή στοιχείων υποστήριξης προηγμένης λειτουργικότητας για την μηχανή αναζήτησης και επειδή κάθε τύπος δεδομένων προωθεί διαφορετικά τέτοια στοιχεία, αναλόγως της επιλογής στην προηγούμενη καρτέλα, χρησιμοποιείται μία από τις κλάσεις WebPageFunctionality, BlogFunctionality, DocumentFunctionality, PaperFunctionality, ImageFunctionality, AudioFunctionality και VideoFunctionality. Η τέταρτη καρτέλα του wizard που παρουσιάζει συγκεντρωτικά και συνοπτικά στον χρήστη τα στοιχεία που έχει εισαγάγει ως αυτό το σημείο πηγάζει από την κλάση Verification.

4.1.3 Η μεθοδολογία του wrapping στην διαδικασία εγκατάστασης

Λόγω της ανάγκης ύπαρξης web-browser για την διαδικασία εγκατάστασης μίας καινούριας μηχανής αναζήτησης, αξιοποιείται η κλάση InstallationWebRenderer, η οποία μπορεί φαινομενικά να δείχνει όμοια με αυτή του WebRenderer, όμως στην ουσία κάνει πολλά παραπάνω, καθώς υποστηρίζει τόσο τις λειτουργίες που σχετίζονται με την πρόγνωση του search script και της query parameter, όσο και εκείνες που αφορούν την διαδραστική εκμάθηση εξόρυξης των τμημάτων των αποτελεσμάτων αναζήτησης στην μηχανή. Η κλάση SearchScript_QueryParameter_Oracle είναι αρμόδια για την πρόβλεψη των εν λόγω

στοιχείων μέσω της επίσκεψης της κύριας ιστοσελίδας της εισαγόμενης μηχανής αναζήτησης και την κατάλληλη επεξεργασία του HTML κώδικά της. Ακόμα, η κλάση SearchScript_QueryParameter_Chooser εμφανίζει ένα frame στον χρήστη με τα ευρεθέντα ζεύγη από search scripts και query parameters, το συνιστώμενο ζεύγος, ενώ του δίνει επίσης την δυνατότητα παράκαμψης των δοθέντων αυτών ζευγών και την συγγραφή του δικού του, εφόσον αυτός κατέχει τις απαιτούμενες, σχετικές, τεχνικές γνώσεις.

Πέραν των ανωτέρω, η κλάση HTMLParser συνεισφέρει καθοριστικά στην υλοποίηση της διαδικασίας του wrapping, δημιουργώντας το εικονικό δένδρο που αντιστοιχεί στον HTML κώδικα μίας search result web-page και βρίσκοντας τα μονοπάτια σε αυτό το δένδρο, στην άκρη των οποίων υπάρχουν τα στοιχεία που απαρτίζουν το κάθε αποτέλεσμα. Εξαιτίας της χρήσης κανονικών εκφράσεων (regular expressions), και στην προηγούμενη κλάση αλλά και σε πολλές άλλες, κατασκευάσαμε την κλάση RegExTester, η οποία λαμβάνοντας ως είσοδο μία συμβολοακολουθία και μία κανονική έκφραση, μας δίνει το αποτέλεσμα της προσπάθειας εφαρμογής της συγκεκριμένης regular expression επί του εν λόγω string. Επιπροσθέτως, ως βοηθητική κλάση (στην HTMLParser) χρησιμεύει και η FlagList, η ανάλυση της οποίας θα γίνει στην αμέσως επόμενη ενότητα καθώς παρουσιάζει εξειδικευμένη λειτουργία, ενώ ουσιαστικά ορίζει έναν καινούριο τύπο δεδομένων, που αφορά μία λίστα από στοιχεία (list), με το κάθε στοιχείο να είναι ένα ζεύγος μίας λογικής μεταβλητής (boolean) και ενός δυναμικού πίνακα (vector).

Ακόμα, η κλάση WrapperSpecifications προσφέρει την δυνατότητα στον χρήστη κατά την result screen-scraping process, να επικυρώσει, να επαναλάβει ή να ακυρώσει το συγκεκριμένο κομμάτι της διαδικασίας εκμάθησης ανάγνωσης αποτελεσμάτων που σχετίζεται με ένα από τα τρία στοιχεία ενός result (title, link, description), ενώ στην περίπτωση της επανάληψης μπορούν να επιλεχθούν οι νέες διαφοροποιημένες παράμετροι ή / και οι πρόσθετοι περιορισμοί (π.χ. μέσω της εφαρμογής σχετικών regexes ή δια του φιλτραρίσματος της θέσης του εκάστοτε στοιχείου εντός του αποτελέσματος). Εκτός από τα προηγούμενα, κατά την διαδικασία της εγκατάστασης μίας άλλης μηχανής αναζήτησης, η κλάση FeaturesCollector είναι αρμόδια να κρατά στην μνήμη του υπολογιστή τις εισαγόμενες από τον χρήστη πληροφορίες, τις οποίες συλλέγει στο πέρας των διαφόρων φάσεων, ενώ η κλάση SE_2_XML_Writer καταγράφει παράλληλα τα εν λόγω δεδομένα σε ένα λιτό και αυστηρό XML αρχείο προσυμφωνημένης μορφής (υπακούει σε DTD πρότυπο).

4.1.4 Οι ήδη ενσωματωμένες υπηρεσίες διαδικτυακής αναζήτησης

Επιπλέον, οι κλάσεις GigablastSearch, GoogleScholarSearch, GoogleSearch, LiveSearch, TechnoratiSearch, YahooSearch και YouTubeSearch πραγματώνουν μέσω APIs (Application Programming Interface) την διαδικτυακή αναζήτηση πληροφοριών στις ομώνυμες μηχανές

αναζήτησης. Επίσης, η κλάση XML_2_SE_Reader είναι υπεύθυνη για την ανάγνωση των XML αρχείων, που βρίσκονται στον φάκελο ενσωματωμένων μηχανών αναζήτησης, έχουν προκύψει από την διαδικασία εγκατάστασης νέων μηχανών από τον χρήστη και περιλαμβάνουν όλα τα απαραίτητα στοιχεία που προσδιορίζουν πλήρως τις καινούριες αυτές μηχανές, ενώ ταυτοχρόνως μαζί με την κλάση WrappedSearchEngine θέτουν τις μεν μηχανές αναζήτησης (wrapped search engines) σε λειτουργία, αλλά και σε συνεργασία με τις δε (API-embedded search engines), για την παροχή μίας συνολικής υπηρεσίας, που συντονίζεται με την βοήθεια της κλάσης SearchManager. Από τα τελευταία, γίνεται άμεσα αντιληπτό ότι σε κάθε τέτοια εισηγμένη μηχανή αναζήτησης αντιστοιχεί ένα κατάλληλο XML αρχείο.

4.1.5 Ο εμπλουτισμός των paper results μέσω της Βάσης Δεδομένων του DBLP

Η κλάση XML_parts_corrector επεμβαίνει σε κάθε ένα από τα 1.000 κομμάτια (τα λεγόμενα DBLP primal parts) του DBLP.xml (διαθέσιμη offline βάση δεδομένων του DBLP με 1.132.629 βιβλιογραφικές εγγραφές επιστημονικών δημοσιεύσεων), διορθώνοντας λάθη που προέκυψαν στην αρχή και το τέλος του κατά την ισομερή κατάτμηση (που είχε ως αποτέλεσμα η τελευταία εγγραφή ενός τμήματος να ξεκινά στο πέρας του και να ολοκληρώνεται στο επόμενο), δημιουργώντας έτσι 1.000 διορθωμένα κομμάτια (τα DBLP parts). Η κλάση DBLP_XML_tokenizer κατακερματίζει το κάθε ένα (π.χ. dblp-Part479of1000) από τα 1.000 αυτά κομμάτια σε στοιχειώδη XML αρχεία (κάθε ένα από τα οποία αντιστοιχεί σε ένα paper – π.χ. dblp-Token481763), τα οποία τοποθετεί σε έναν καινούριο φάκελο (π.χ. DBLP tokens \ DBLP token group 479).

Εκτός αυτών, η κλάση XML_full_documents_creator χρησιμεύει για την δημιουργία εγγράφων κατάλληλου είδους (ένα document για κάθε token, το οποίο περιέχει όλη την χρήσιμη πληροφορία), που χρησιμοποιεί το εργαλείο κατασκευής ευρετηρίων (indexing tool) Lucene [AD05]. Μια άλλη κλάση, που φέρει το όνομα XML_index_and_info_extractor, οικοδομεί 1.000 ευρετήρια (ένα για κάθε φάκελο με tokens – π.χ. full indices \ full index 859), βασιζόμενη σε documents του τύπου που προαναφέρθηκε και είναι μεστά με τα ενδιαφέροντα δεδομένα. Επίσης, η κλάση Full_indices_merger συντελεί το έργο της συγχώνευσης των εν λόγω ευρετηρίων σε ένα και μοναδικό (full index) που αφορά όλα τα tokens, επομένως ολόκληρο το DBLP.xml.

Πλέον, η κλάση XML_full_search είναι σε θέση να εκτελεί ερωτήσεις (κάνοντας χρήση π.χ. της τιμής ενός attribute κάποιου paper) προς την συγκεκριμένη βάση δεδομένων και να λαμβάνει τις απαντήσεις (π.χ. λοιπές τιμές των attributes του ίδιου paper ή άλλα papers με ίδια τιμή στο εν λόγω attribute) πολύ γρήγορα. Σημειώνουμε εδώ, ότι η λειτουργία των κλάσεων Time_Teller και Special_Character_Handler είναι επικουρική. Η πρώτη από αυτές είναι υπεύθυνη για την ενημέρωση του χρήστη περί του χρόνου ολοκλήρωσης του εκάστοτε

από τα επιχειρούμενα έργα στην κατάλληλη πάντοτε μορφή, ενώ η δεύτερη διαχειρίζεται τους μη λατινικούς χαρακτήρες που απαντώνται σε διάφορες δημοσιεύσεις (όπως κάποιοι γερμανικοί, γαλλικοί, ισλανδικοί, δανέζικοι, κ.ά. χαρακτήρες) βάσει του ISO 8859-1 προτύπου.

Στις τελευταίες τρεις σελίδες δόθηκε μία απαρίθμηση των κλάσεων που συνιστούν την επέκτασή μας στο FreeMind, συνοδευόμενων από σύντομες περιγραφές του πυρήνα λειτουργίας της κάθε μιας. Η λεπτομερής ανάλυση των διαφόρων λειτουργιών, μεθόδων και συναρτήσεων ακολουθεί στην επόμενη ενότητα.

4.2 Περιγραφή Κλάσεων

Στην παρούσα ενότητα παρουσιάζουμε συνοπτικά τις λειτουργίες με τις οποίες είναι επιφορτισμένη η κάθε κλάση του project μας. Για λόγους πληρότητας και ευκολότερης κατανόησης γίνεται κάθε φορά αναφορά (με συστηματικό τρόπο) στις μεθόδους της εκάστοτε κλάσης, ενώ οι διάφορες περιγραφές δίνονται σύντομα, προκειμένου ο ενδιαφερόμενος να είναι σε θέση να ανατρέξει στο σημείο που επιθυμεί γρήγορα και να λάβει την επιθυμητή πληροφόρηση. Σημειώνουμε εδώ, ότι στην ανάλυσή μας δεν περιλαμβάνεται ο κώδικας της εφαρμογής ή τμήματα αυτού.

4.2.1 Πακέτο κλάσεων mindmapmode

- Τροποποιεί το κύριο περιβάλλον του FreeMind και κατασκευάζει το παράθυρο της υπηρεσίας προηγμένης διαδικτυακής μετα-αναζήτησης.

Το εν λόγω πακέτο περιλαμβάνει τις ακόλουθες κλάσεις: MindMapToolBar & MindMapSearchFrame. Σε γενικές γραμμές, αφορούν την διαπροσωπεία μεταξύ συστήματος και χρήστη, όσον αφορά την υλοποίηση των αποσκοπούμενων επεκτάσεων – πρόσθετων λειτουργικοτήτων.

- ***Κλάση MindMapToolBar***
  - ✓ Επιφέρει τις απαραίτητες αλλαγές στην βασική μπάρα εργαλείων του FreeMind, προκειμένου αυτή πλέον να υποστηρίζει τις επιχειρηθείσες προσθήκες.

Είναι υπεύθυνη για την κατασκευή και προβολή της κύριας μπάρας εργαλείων του FreeMind, η οποία πλέον περιέχει και 3 buttons ακόμα, ένα για πλοήγηση στο Internet, ένα για την πραγματοποίηση της προηγμένης λειτουργίας μετα-αναζήτησης και ένα για εγκατάσταση νέων μηχανών αναζήτησης. Παραθέτουμε ακολούθως τις σημαντικότερες μεθόδους της εν λόγω κλάσης, ενώ παράλληλα παρουσιάζουμε συνοπτικά την λειτουργία τους:

1) Μέθοδος *gigablast_harvest*: Προβαίνει στην συγκομιδή των αποτελεσμάτων αναζήτησης που αφορούν την Gigablast Search Engine, με παραμέτρους τις extra

λέξεις-κλειδιά προς χρησιμοποίηση και το επιθυμητό πλήθος αποτελεσμάτων. Υπενθυμίζουμε πως η κεντρική keyword είναι γνωστή, καθώς αυτή λαμβάνεται από το mindmap.

2) Μέθοδος *google_harvest*: Ομοίως με την προηγούμενη, χρησιμοποιώντας ωστόσο την διάσημη μηχανή αναζήτησης στο Διαδίκτυο του Google.
3) Μέθοδος *googlescholar_harvest*: Ομοίως με την πρώτη, κάνοντας χρήση της Google Scholar Paper Search Engine.
4) Μέθοδος *live_harvest*: Ομοίως με την αρχική, αξιοποιώντας όμως την Microsoft Live Web Search Engine.
5) Μέθοδος *technorati_harvest*: Ομοίως με όλες τις άλλες, χρησιμοποιώντας βέβαια την δημοφιλή μηχανή αναζήτησης ιστολογίων Technorati.
6) Μέθοδος *yahoo_harvest*: Ομοίως με τις υπόλοιπες, βασίζοντας όμως την λειτουργία της διαδικτυακής αναζήτησης στην Yahoo Web Search Engine.
7) Μέθοδος *youtube_harvest*: Ομοίως με τις προηγούμενες, εκμεταλλευόμενη την δυνατότητα αναζήτησης video, που παρέχει ένας από τους πιο πολυσύχναστους ιστοτόπους, εκείνος του YouTube.
8) Μέθοδος *wrapped_engine_harvest*: Ομοίως με τις προαναφερθείσες, κάνοντας ωστόσο χρήση μίας από τις wrapped search engines κάθε φορά, η οποία έχει εγκατασταθεί με την βοήθεια της τεχνικής του screen-scraping, και δεν αποτελεί API-embedded search engine (όπως όλες οι προηγούμενες).
9) Μέθοδος *web_powered_map_enrichment*: Εμπλουτίζει το mindmap με τα αποτελέσματα των διαφόρων μηχανών αναζήτησης (αφού αυτά έχουν ταξινομηθεί και συγχωνευθεί), προσθέτοντας ένα child node στον ενδιαφέροντα κόμβο, επί του οποίου βασίστηκε η αναζήτηση, για κάθε result, προσφέροντας τον τίτλο του αποτελέσματος ως label του κόμβου και τον διαδικτυακό σύνδεσμό του ως hyperlink.

- ***Κλάση MindMapSearchFrame***
  - ✓ Δημιουργεί και απεικονίζει το παράθυρο που επιτελεί την προηγμένη διαδικτυακή μετα-αναζήτηση, καθορίζει τις παραμέτρους της κάθε αναζήτησης, εμφανίζει τα αποτελέσματα με διάφορους τρόπους, ενώ επίσης επιτρέπει την διαλογή τους και την μεταφορά των επιλεγμένων στον χάρτη σκέψεων.

Η κλάση αυτή είναι επιφορτισμένη με την κατασκευή, την εμφάνιση και την διατήρηση της διαδραστικής λειτουργικότητας ενός από τα κύρια frames της εφαρμογής μας, εκείνο της προηγμένης μετα-αναζήτησης πληροφοριών στον Παγκόσμιο Ιστό. Η αναζήτηση πραγματοποιείται βάσει των λέξεων του label του επιλεγμένου mindmap node, ενώ δίνεται ακόμα η δυνατότητα στον χρήστη να προσθέσει extra keywords, να καθορίσει το επιθυμητό

πλήθος των επιστρεφόμενων αποτελεσμάτων, καθώς και να επιλέξει τύπο αποτελεσμάτων (π.χ. general web pages ή papers). Οι άλλοι τύποι αποτελεσμάτων (blogs, documents, maps, images, audio, video) μένουν ως μελλοντική επέκταση, ενώ επέκταση συνιστά επίσης η υποστήριξη των στοιχείων της προηγμένης αναζήτησης από το σύστημα (όπως η αναζήτηση σε ιστοσελίδες γενικού περιεχομένου μόνο βάσει συγκεκριμένου χρονικού διαστήματος ή η αναζήτηση σε ακουστικά αρχεία προκαθορισμένης ποιότητας κλπ). Κάνοντας την αναζήτηση, εμφανίζεται μία λίστα αποτελεσμάτων, ενώ ο χρήστης μπορεί να επιλέξει μεταξύ short και extended result form, καθώς και ανάμεσα σε mixed result list ή result lists per search engine. Κάθε αποτέλεσμα φέρει δίπλα του ένα checkbox, και εφόσον έχουν επιλεγεί τα επιθυμητά, είναι δυνατή η μεταβίβαση αυτών στον χάρτη σκέψεων, κάτι που επιτυγχάνεται μέσω της συνεργασίας της παρούσας κλάσης με την προηγούμενη. Στην εν λόγω κλάση αξιοποιούνται τα έτοιμα Java πακέτα του AWT (Abstract Windowing Toolkit) και του Swing, για την δημιουργία των ζητούμενων containers & components, με τους απαραίτητους event listeners και event handlers.

4.2.2 Πακέτο κλάσεων searchmode

- Αναλαμβάνει την διεξαγωγή των αναζητήσεων στο Διαδίκτυο, καθώς και την οργάνωση των επιστρεφόμενων αποτελεσμάτων.

Το πακέτο αυτό περιέχει τις κλάσεις που επιτελούν το έργο της διαδικτυακής αναζήτησης (χρησιμοποιώντας τόσο τις API-embedded search engines όσο και τις wrapped by user search engines) και της κατασκευής της εκάστοτε κατάλληλης λίστας αποτελεσμάτων.

- ***Κλάση GigablastSearch (API-embedded search engine)***
  - ✓ Εκτελεί ερωτήσεις προς την μηχανή αναζήτησης Gigablast Web Search και περισυλλέγει τα αποτελέσματα που αυτή δίνει ως απάντηση.

Η εν λόγω κλάση συνιστά την πρώτη από μία σειρά κλάσεων, οι οποίες έχοντας ως αφετηρία το εκάστοτε παρεχόμενο API και συνήθως ένα προσφερόμενο Developer-Key, αναλαμβάνουν το έργο της λειτουργίας ως query clients στις αντίστοιχες search engines. Αξιοποιεί την μηχανή αναζήτησης του Gigablast, προκειμένου να της θέσει ερωτήσεις και να λάβει λίστες αποτελεσμάτων ως απαντήσεις. Οι μέθοδοι που χρησιμοποιεί είναι οι εξής:

1) Μέθοδος *replies*: Δέχεται ως ορίσματα τις λέξεις-κλειδιά, στις οποίες θα βασιστεί η διαδικτυακή αναζήτηση, καθώς και το ζητούμενο πλήθος των αποτελεσμάτων, ενώ επιστρέφει ένα διάνυσμα συμβολοακολουθιών, κάθε στοιχείο του οποίου ισοδυναμεί με ένα πλήρες αποτέλεσμα (λέγοντας πλήρες εννοούμε ότι περιλαμβάνει όλα τα διαθέσιμα δεδομένα που παρέχει η μηχανή προέλευσης, όπως π.χ. title, URL & summary για web pages).

2) Μέθοδος *reply*: Καθορίζοντας έναν ακέραιο αριθμό i επιστρέφει το result της λίστας αποτελεσμάτων που βρίσκεται στην συγκεκριμένη θέση.
3) Μέθοδος *title*: Προσδιορίζει τον τίτλο του αποτελέσματος που καταλαμβάνει την θέση i στην result list.
4) Μέθοδος *url*: Ομοίως με την προηγούμενη, μόνο που αντί για τον τίτλο καθορίζει τον αντίστοιχο διαδικτυακό σύνδεσμο του αποτελέσματος i.
5) Μέθοδος *description*: Ομοίως με τις δύο παραπάνω, όμως δίνει την εκάστοτε περίληψη του αποτελέσματος.
6) Μέθοδος *number*: Γνωστοποιεί απλώς το εισαχθέν – επιδιωκόμενο πλήθος αποτελεσμάτων.

- ***Κλάση GoogleScholarSearch (API-embedded search engine)***
  - ✓ Ομοίως με την πρώτη, μόνο που χρησιμοποιεί την μηχανή αναζήτησης (για επιστημονικές δημοσιεύσεις) του Google Scholar.
- ***Κλάση GoogleSearch (API-embedded search engine)***
  - ✓ Ομοίως με τις άλλες δύο, κάνοντας χρήση ωστόσο της Google Web Search Engine.
- ***Κλάση LiveSearch (API-embedded search engine)***
  - ✓ Ομοίως με τις προαναφερθείσες, αξιοποιώντας όμως την μηχανή αναζήτησης Microsoft Live.
- ***Κλάση TechnoratiSearch (API-embedded search engine)***
  - ✓ Ομοίως με τις υπόλοιπες, βασιζόμενη στην πιο δημοφιλή μηχανή αναζήτησης ιστολογίων (blogs), εκείνη του Technorati.
- ***Κλάση YahooSearch (API-embedded search engine)***
  - ✓ Ομοίως με τις παραπάνω, κάνοντας χρήση βέβαια της πασίγνωστης Yahoo Search Engine.
- ***Κλάση YouTubeSearch (API-embedded search engine)***
  - ✓ Ομοίως με τις προηγούμενες, στηρίζοντας την λειτουργία της στην διάσημη μηχανή αναζήτησης αρχείων οπτικοακουστικού περιεχομένου (video) του YouTube.
- ***Κλάση WrappedEngineSearch (user installed search engines)***
  - ✓ Ομοίως με όλες τις υπόλοιπες, χρησιμοποιώντας όμως μία user-installed μηχανή αναζήτησης και όχι κάποια API-embedded.

Η συγκεκριμένη κλάση περιλαμβάνει μεν τις ίδιες μεθόδους με όλες τις ανωτέρω, ωστόσο, η φιλοσοφία εισαγωγής και χρήσης μίας τέτοιας μηχανής αναζήτησης είναι τελείως διαφορετική, επομένως και ο τρόπος υλοποίησής της. Στην ουσία, πρόκειται για μία κλάση που οικοδομεί ένα γενικότερο περίγραμμα ένταξης μηχανών αναζήτησης που υπάγονται σε

αυτή την κατηγορία, ενώ κάθε φορά εξειδικεύεται (για να προσομοιώσει την εκάστοτε χρησιμοποιούμενη μηχανή) βάσει μίας σειράς παραμέτρων, οι οποίες βρίσκονται αποθηκευμένες σε ένα καταλλήλως δομημένο, λιτό και αυστηρά σαφές XML αρχείο. Υπενθυμίζουμε στο σημείο αυτό, ότι κάθε μηχανή, που έχει εισαχθεί με την παροχή δεδομένων από τον ίδιο τον χρήστη συν την εκμετάλλευση της τεχνικής του screen-scraping, τηρεί στο σύστημά μας ένα τέτοιο XML αρχείο, το οποίο είναι αναγκαίο και επαρκές για την ένταξή της στην διαδικασία της διαδικτυακής μετα-αναζήτησης και το οποίο την καθιστά πλήρως λειτουργική. Ειδική αναφορά και επεξήγηση της δομής του εν λόγω XML αρχείου, καθώς και του ρόλου του ως ενδιαμέσου για την ενσωμάτωση μηχανών αναζήτησης τέτοιου τύπου στην εφαρμογή γίνεται στην ενότητα 4.4 περί "κωδικοποίησης αρχείων".

- ***Κλάση XML_2_SE_Reader***
  - ✓ Διαβάζει για μία wrapped search engine τις απαραίτητες τιμές των στοιχείων που την καθιστούν λειτουργική από το XML αρχείο που της αντιστοιχεί.

Η κλάση XML-to-Search_Engine-Reader είναι επιφορτισμένη με το έργο της εκμαίευσης των τιμών των χαρακτηριστικών παραμέτρων μίας wrapped search engine από το αντίστοιχο τηρούμενο XML αρχείο και η διάθεση αυτών στην προηγούμενη κλάση (μέσω μίας απλής δομής δεδομένων), προκειμένου η τελευταία από γενικού περιεχομένου πρόγραμμα να εξειδικευθεί, παίρνοντας την μορφή της συγκεκριμένης μηχανής αναζήτησης και συμμετέχοντας κατ' αυτόν τον τρόπο στην συνολική διαδικασία της advanced web meta-search service. Η κύρια μέθοδος της συζητούμενης κλάσης είναι η *read_XML_from_file*, η οποία λαμβάνοντας ως είσοδο το εκάστοτε κατάλληλο XML αρχείο και αξιοποιώντας το DOM μοντέλο (Document-Object Model), προβαίνει σε διάσχιση της αντίστοιχης σχηματιζόμενης εικονικής δενδρικής δομής (σημαντικές έννοιες εδώ: parent, children, root, leaf κλπ) και εξόρυξη των απαραίτητων ή / και χρήσιμων δεδομένων (είτε αυτά παρέχονται ως node names, είτε ως node values, είτε ως attributes, είτε ως attribute values).

- ***Κλάση SearchManager***
  - ✓ Καθορίζει τις κλήσεις προς τις διαθέσιμες μηχανές αναζήτησης, συντονίζει την λειτουργία τους και καταρτίζει την τελική ενιαία λίστα των αποτελεσμάτων.

Η συγκεκριμένη κλάση «κρατά την μπαγκέτα του μαέστρου» στην διαδικασία της αναζήτησης πληροφοριών στο Διαδίκτυο. Αναλαμβάνει να καθορίσει και να συντονίσει τις κλήσεις προς τις διάφορες μηχανές αναζήτησης, καθώς και τις λήψεις των απαντήσεων από αυτές, ενώ επίσης ασχολείται με την ταξινόμηση και την συγχώνευση των αποτελεσμάτων. Απαρτίζεται από τις κάτωθι μεθόδους, των οποίων η απαρίθμηση ακολουθεί αμέσως, συνοδευόμενη από αντίστοιχες σύντομες περιγραφές.

1) Μέθοδος *wrapped_engines_ embedding_system*: Είναι επιφορτισμένη με την εύρεση όλων των XML αρχείων που βρίσκονται σε ένα συγκεκριμένο path, την ανάγνωση

και την αποθήκευση των δεδομένων που αυτά φέρουν (αξιοποίηση της κλάσης XML_2_SE_Reader), καθώς και την δήλωση των wrapped search engines στην λίστα με τις χρησιμοποιούμενες μηχανές αναζήτησης, στην οποία έχουν προενσωματωθεί οι API-embedded search engines.

2) Μέθοδος *distributor*: Λαμβάνοντας ως είσοδο το επιθυμητό πλήθος των συνολικών αποτελεσμάτων, καθορίζει τον αριθμό αποτελεσμάτων που πρέπει να ζητηθεί από την κάθε μηχανή αναζήτησης, αριθμός ο οποίος είναι άμεσα σχετιζόμενος με την βαθμολογία της εκάστοτε μηχανής για τον ενδιαφέροντα τύπο δεδομένων. Προφανώς, το άθροισμα των επιμέρους πληθών δίνει το συνολικό πλήθος των αποτελεσμάτων.

3) Μέθοδος *ranker*: Επιτελεί το έργο της βαθμολόγησης, κατάταξης και συγχώνευσης των αποτελεσμάτων αναζήτησης των διαφόρων μηχανών, κάνοντας χρήση του δημοκρατικού αλγορίθμου ταξινόμησης αποτελεσμάτων διαφορετικών βαρυτήτων προερχόμενων από μηχανές αναζήτησης άλλης σημαντικότητας *Borda-Fuse* και του απλού-δημοφιλή αλγορίθμου ταξινόμησης BubbleSort.

4.2.3 Πακέτο κλάσεων indexfindmode

- Οργανώνει την Βάση Δεδομένων του DBLP, τις επισυνάπτει σχετικό ευρετήριο, πραγματοποιεί τις αναζητήσεις σε αυτήν και εμπλουτίζει αναλόγως τα διασταυρωμένα αποτελέσματα.

Το συγκεκριμένο πακέτο κλάσεων αναλαμβάνει την οργάνωση, την ευρετηριοποίηση και την αναζήτηση στην Βάση Δεδομένων – αντίγραφο του DBLP που διατηρούμε. Λέγοντας οργάνωση, εννοούμε την μετατροπή του τεράστιου παρεχόμενου XML αρχείου από το Universitaet Trier σε κατάλληλη μορφή, προκειμένου αυτό να είναι εύκολα επεξεργάσιμο, ενώ η κατασκευή ευρετηρίου εξασφαλίζει την γρήγορη προσπέλαση, αναζήτηση και εύρεση του ζητούμενου paper στην εν λόγω βάση δεδομένων. Ξεκινούμε την επεξεργασία του DBLP.xml (μεγέθους ~453MB!) με τον διαχωρισμό του σε 1.000 ισομεγέθη XML κομμάτια, αξιοποιώντας ένα κοινό freeware εργαλείο, το FileSplitter (ανάγκη μοναδικής stand-alone εκτέλεσης). Ο απώτερος σκοπός της διαμέρισης είναι η υπέρβαση των δυσκολιών διαχείρισης ενός τόσο ευμεγέθους αρχείου (χαρακτηριστικά αναφέρουμε την αδυναμία ανάπτυξης του σχετικού XML δένδρου στην μνήμη του υπολογιστή).

- ***Κλάση XML_parts_corrector (ανάγκη μοναδικής stand-alone εκτέλεσης)***
  - ✓ Επισκευάζει τα 1.000 XML μέρη στα οποία σπάσαμε το παρεχόμενο DBLP.xml.

Η συγκεκριμένη κλάση είναι επιφορτισμένη με την διόρθωση των 1.000 αρχικών XML τμημάτων, που προέκυψαν από το splitting του DBLP.xml. Η διόρθωση έγκειται στην

αποκατάσταση του πέρατος και της αρχής κάθε μέρους. Επειδή η διαμέριση έγινε σε κομμάτια ίδιου μεγέθους, η τελευταία καταχώρηση καθενός XML αρχείου μοιράζεται μεταξύ του τρέχοντος και του επόμενου. Μετά την επέμβασή μας όμως, μεταφέρεται ολόκληρη η τελευταία εγγραφή στο δεύτερο τμήμα, εξαλείφοντας το πρόβλημα. Επιπλέον, η κλάση αυτή μεριμνά για την αφαίρεση όλων των κενών γραμμών στα διάφορα XML κομμάτια.

- ***Κλάση DBLP_XML_tokenizer (ανάγκη μοναδικής stand-alone εκτέλεσης)***
  - ✓ Κατακερματίζει το κάθε ένα από τα 1.000 επισκευασμένα XML μέρη σε στοιχειώδεις εγγραφές των λίγων γραμμών, οι οποίες συνιστούν επίσης XML έγγραφα.

Η εν λόγω κλάση παίρνει ως είσοδο τα 1.000 διορθωμένα μεγάλα XML τμήματα και υλοποιεί το σπάσιμό τους σε απολύτως στοιχειώδη μικρά XML αρχεία. Κάθε θεμελιώδης εγγραφή αντιστοιχεί σε μία επιστημονική δημοσίευση (paper), συνιστά την λειτουργική μονάδα της βάσης δεδομένων μας, ενώ αναπαρίσταται από ένα XML document. Έτσι, ο διαχωρισμός γίνεται σε τέτοιου είδους ελάχιστες καταχωρήσεις, ενώ τελικά προκύπτουν 1.129.251 στο σύνολο εγγραφές. Προκειμένου να ξεπεραστούν οι περιορισμοί που θέτουν τα συνήθη λειτουργικά συστήματα του εμπορίου (όπως η επίτρεψη έως 1.000 files ανά folder που έχουν τα Windows XP ή τα Vista), μοιράζουμε το ένα εκατομμύριο και πλέον XML έγγραφα σε 1.000 φακέλους, με όσο το δυνατόν πιο ομοιόμορφη (ως προς το πλήθος) κατανομή (για να κατανοήσει κανείς το πρόβλημα του απόλυτα ίσου μοιράσματος των αρχείων στους φακέλους, αρκεί να αναλογιστεί ότι κάθε μία εγγραφή έχει διαφορετικό μέγεθος, οπότε σε κάθε ένα από τα 1.000 αρχικά XML κομμάτια υπάρχει άλλο πλήθος καταχωρήσεων).

- ***Κλάση XML_full_documents_creator (ανάγκη μοναδικής stand-alone εκτέλεσης)***
  - ✓ Είναι σε θέση να δημιουργεί Lucene documents που αντιστοιχούν στις DBLP XML εγγραφές της Βάσης Δεδομένων μας.

Η διαδικασία της ευρετηριοποίησης (indexing) πραγματοποιείται με το open source Java tool Apache Lucene. Για να προχωρήσουμε με αυτήν, πρέπει να κατασκευαστεί ένα ειδικού τύπου document (*Lucene document*) για κάθε ένα από τα στοιχειώδη XML αρχεία – εγγραφές. Αυτά τα documents περιλαμβάνουν 18 fields (type, writers, title, pages, year, volume, journal, number, link, url, date, book, isbn, series, series_url, publisher, publisher_url, school) με τις εκάστοτε αντίστοιχες τιμές και εφόσον κάποιο από αυτά τα πεδία δεν υφίσταται για την τρέχουσα επιστημονική καταχώρηση, τότε λαμβάνει μία ειδική τιμή που το υποδηλώνει αυτό ("empty"). Η λήψη των απαραίτητων τιμών των διαφόρων πεδίων γίνεται με εκμετάλλευση του DOM μοντέλου και με διάσχιση του αντίστοιχου κάθε φορά XML δένδρου.

- ***Κλάση XML_index_and_info_extractor (ανάγκη μοναδικής stand-alone εκτέλεσης)***
  - ✓ Συγγράφει τα 1.000 υποευρετήρια, κάθε ένα από τα οποία αντιστοιχεί σε έναν φάκελο με βιβλιογραφικές καταχωρήσεις.

Με την βοήθεια της συγκεκριμένης κλάσης επιτυγχάνεται το indexing καθενός από τους 1.000 φακέλους. Δεν προβήκαμε σε indexing όλων των φακέλων συγχρόνως, καθώς κάτι τέτοιο οδηγούσε σε κατάρρευση του Java προγράμματος, εξαιτίας της υπερχείλισης του σωρού στην μνήμη (heap space problem), που οφείλεται στις απαγορευτικές απαιτήσεις ενός τέτοιου εγχειρήματος. Επιπροσθέτως, απαραίτητη είναι η διακοπή της λειτουργίας οποιουδήποτε antivirus software είναι τυχόν σε λειτουργία, καθώς το τελευταίο δεν επιτρέπει την προσπέλαση τόσων πολλών αρχείων σε τόσο σύντομο χρονικό διάστημα από κάποια εφαρμογή, θεωρώντας την ως απειλή. Η συζητούμενη κλάση χρειάζεται την αμέσως προηγούμενη, προκειμένου να υλοποιήσει την ευρετηριοποίηση (μία κλήση της κατάλληλης μεθόδου για κάθε μία βιβλιογραφική εγγραφή μορφής συντόμου XML αρχείου).

- ***Κλάση Full_indices_merger (ανάγκη μοναδικής stand-alone εκτέλεσης)***
  - ✓ Προκαλεί την συνένωση των 1.000 ευρετηρίων σε ένα μοναδικό.

Η παρούσα κλάση συγχωνεύει τα 1.000 υπο-ευρετήρια (κάθε ένα από τα οποία αντιστοιχεί σε έναν φάκελο καταχωρήσεων επιστημονικών δημοσιεύσεων) σε ένα ενιαίο και τελικώς το βελτιστοποιεί ως προς την απόδοσή του.

- ***Κλάση XML_full_search***
  - ✓ Εκτελεί τις αναζητήσεις στην Βάση Δεδομένων μας, επιστρέφοντας σε περίπτωση απολύτως επιτυχούς ταιριάσματος. το ίδιο αποτέλεσμα, καταλλήλως εμπλουτισμένο.

Έχοντας στην διάθεσή μας ένα ευρετήριο που έχει πρόσβαση σε όλες τις εγγραφές – papers, είμαστε πλέον σε θέση να εκτελούμε αναζητήσεις στην Βάση Δεδομένων και να λαμβάνουμε τις απαντήσεις σε πολύ μικρό χρονικό διάστημα, κάτι το οποίο δίχως την τεχνική του indexing, θα ήταν αδύνατο. Μία query μπορεί να βασιστεί σε οποιοδήποτε field, ενώ η αντίστοιχη reply, περιλαμβάνει (εφόσον η απάντηση είναι θετική) όλες τις πληροφορίες της ζητούμενης καταχώρησης, δηλαδή τις τιμές όλων των πεδίων. Σημειώνουμε εδώ, ότι το Lucene χρησιμοποιεί ένα σύστημα αναζήτησης που στηρίζεται στον βαθμό ομοιότητας (*similarity*) του ερωτώμενου με τα διαθέσιμα. Εφόσον η ομοιότητα είναι μονάδα, αυτό σημαίνει ότι έχει βρεθεί ένα τέλειο ταίριασμα (*perfect match*). Εμείς αποζητούμε μόνο τέλεια ταιριάσματα, ενώ απορρίπτουμε κάθε άλλο ταίριασμα.

- ***Κλάση Time_Teller***
  - ✓ Ενημερώνει διαρκώς για τον χρόνο εκτέλεσης της καθεμίας από τις παραπάνω mini εφαρμογές του συγκεκριμένου πακέτου κλάσεων.

Επειδή τα προαναφερθέντα προγράμματα είναι ακόμα και σε έναν ισχυρό υπολογιστή σχετικά χρονοβόρα, η εν λόγω κλάση επιτελεί το έργο της μέτρησης της χρονικής διάρκειας της εκτέλεσής τους. Εφόσον το πρόγραμμα ή κάποια ενότητα αυτού ολοκληρωθεί, τυπώνεται στην κονσόλα η σχετική πληροφόρηση σε φιλική προς τον χρήστη μορφή (HH:mm:ss).

- ***Κλάση Special_Character_Handler***
  - ✓ Αντιστοιχίζει τις κωδικές αριθμήσεις ασυνήθιστων χαρακτήρων εγγραφών της Βάσης Δεδομένων μας, στους ίδιους τους χαρακτήρες, σύμφωνα με γνωστό πρότυπο.

Λόγω του γεγονότος ότι στις καταχωρημένες εγγραφές του DBLP απαντώνται στοιχεία σε διάφορες (λατινογενείς) γλώσσες (π.χ. Αγγλικά-Γαλλικά-Γερμανικά-Ιταλικά-Ισπανικά-Τσέχικα-Ισλανδικά-Δανέζικα κ.ά.) ανέκυψε η ανάγκη αντιμετώπισης ασυνήθιστων χαρακτήρων, που υπακούουν ωστόσο στο πρότυπο κωδικοποίησης ISO 8859-1. Η κλάση αυτή αντικαθιστά τους εμφανιζόμενους κωδικούς αριθμούς με τους αντίστοιχους χαρακτήρες που υποδεικνύονται από το συγκεκριμένο πρότυπο.

Σημείωση: Τα προγράμματα που συνοδεύονται από τον χαρακτηρισμό «*ανάγκη stand-alone μοναδικής εκτέλεσης*» περιλαμβάνουν main method, προκειμένου να εκτελούνται ανεξαρτήτως από τα υπόλοιπα τμήματα της εφαρμογής. Απαιτείται η εκτέλεσή τους μία φορά ακριβώς και κατά την σειρά που αυτά δίνονται. Αφού σχηματισθεί το τελικό ενιαίο ευρετήριο ολόκληρης της ΒΔ του DBLP, τα προγράμματα της εν λόγω κατηγορίας δεν είναι πλέον απαραίτητα.

4.2.4 Πακέτο κλάσεων installationmode

- ➢ Αφορά την εγκατάσταση μίας καινούριας μηχανής αναζήτησης στο σύστημά μας, εξαιρουμένης της διαδικασίας του wrapping.

Περιέχει το σύνολο των κλάσεων που υλοποιούν την διαδικασία εγκατάστασης μίας νέας μηχανής αναζήτησης στην εφαρμογή μας. Η ενσωμάτωση αφορά μία καινούρια wrapped search engine (σε αντίθεση με τις προϋπάρχουσες API-embedded) και επιτυγχάνεται τόσο με την εισαγωγή δεδομένων από τον χρήστη όσο και με την αξιοποίηση της τεχνικής του screen-scraping. Η όλη διαδικασία πραγματοποιείται με την βοήθεια ενός wizard, ενώ ο χρήστης μπορεί σε οποιοδήποτε βήμα είτε να οπισθοχωρήσει στην προηγούμενη φάση είτε να ακυρώσει την εγκατάσταση συνολικά. Ειδικά μηνύματα λάθους (error messages) ή ειδοποιήσεις (warnings) κάνουν την εμφάνισή τους, όταν ο χρήστης δώσει μη αποδεκτή είσοδο. Στην πλειονότητα των κλάσεων γίνεται χρήση των πακέτων του AWT και του Swing.

- ***Κλάση Identification***
  - ✓ Ζητά από τον χρήστη τα κύρια στοιχεία ταυτοποίησης της νέας μηχανής αναζήτησης.

Συνιστά την πρώτη από μία σειρά καρτελών στον οδηγό εγκατάστασης. Καλεί τον χρήστη να εισάγει το όνομα της εγκαθιστάμενης μηχανής αναζήτησης, καθώς και τον κύριο ιστότοπό της (*base URL*).

- ***Κλάση Use_and_Evaluation***
  - ✓ Καλεί τον χρήστη να δηλώσει το πεδίο δράσης και την αξιολόγηση της μηχανής.

Η δεύτερη καρτέλα που εμφανίζεται στον installation wizard και αναμένει από τον χρήστη τον προσδιορισμό του τύπου δεδομένων που υποστηρίζει η νεοεισαγόμενη μηχανή αναζήτησης (web page, blog, document, paper, map, image, audio ή video), καθώς και την αξιολόγηση της εν λόγω μηχανής σε μία κλίμακα από 1 έως 5 (η βαθμολογία μπορεί να αποδοθεί είτε μέσω του παρεχόμενου star-rating system είτε με δακτυλογράφηση του αντίστοιχου ακέραιου αριθμού).

- ***Κλάση AudioFunctionality***
  - ✓ Ο χρήστης μπορεί να προσδώσει στην εγκαθιστάμενη μηχανή αναζήτησης για audio πρόσθετες λειτουργικότητες μέσα από σχετικές επιλογές και κατάλληλη διάθεση δεδομένων.

Εφόσον ο χρήστης επιλέξει ως χρησιμοποιούμενο τύπο δεδομένων από την καινούρια μηχανή αναζήτησης το *audio*, καλείται να εισαγάγει προαιρετικά τυχόν ενδιαφέρουσες επεκτάσεις, προκειμένου να υποστηρίζονται, στην μελλοντική χρήση της συγκεκριμένης μηχανής από το υποσύστημα διαδικτυακής αναζήτησης, στοιχεία προηγμένης λειτουργικότητας. Εν προκειμένω οι δυνατότητες αυτές συνοψίζονται στα εξής:

1) bitrate-based search support,
2) duration-based search support,
3) file type – based search support &
4) genre-based search support.

Μπορεί να γίνει επιλογή όποιων από αυτές τις τέσσερεις επεκτάσεις επιθυμούμε, και στην περίπτωση προτίμησης κάποιας, απαιτούνται οι τιμές για τις ακόλουθες παραμέτρους:

1) script expansion (τμήμα script που προστίθεται στο base URL, καθιστώντας την συγκεκριμένη λειτουργία έτοιμη προς χρήση),
2) search filter (φράση ειδικής σύνταξης με έναν προσδιοριστή και την αντίστοιχη τιμή, τα οποία αποστέλλονται μαζί με τις λέξεις-κλειδιά κατά την θέση του ερωτήματος αναζήτησης),
3) query expansion (πρόκειται για την αυτόματη προσθήκη καταλλήλων extra keywords που κατευθύνουν κατά κάποιον τρόπο την αναζήτηση).

Από τα παραπάνω, κατανοούμε ότι η συμπλήρωση της παρούσας καρτέλας χρειάζεται κάποιες πιο εξειδικευμένες γνώσεις και απευθύνεται σε έμπειρους χρήστες, ενώ ο απλός χρήστης μπορεί να την αγνοήσει και να συνεχίσει με την διαδικασία ενσωμάτωσης της νέας μηχανής αναζήτησης.

- ***Κλάση BlogFunctionality***
  - ✓ Ίδια λειτουργία, με την διαφορά ότι η μηχανή αναζήτησης πρέπει να αφορά blogs.

Ομοίως με την πρώτη, εμφανίζεται η αντίστοιχη καρτέλα, εφόσον έχει επιλεγεί ως data type της search engine το *blog*. Η διαφορά έγκειται στις παρεχόμενες επεκτάσεις, τις οποίες απαρτίζουν οι κάτωθι:

1) blog-based search support,
2) post-based search support,
3) tag-based search support,
4) URL-based search support.

- ***Κλάση DocumentFunctionality***
  - ✓ Ίδια λειτουργία, με την διαφορά ότι η νέα μηχανή πρέπει να αφορά documents.

Ομοίως με τις προαναφερθείσες, παρουσιάζεται η σχετική καρτέλα, μόνο αν βρισκόμαστε στην περίπτωση *document*-related search engine. Οι διαφορετικές υποψήφιες πρόσθετες λειτουργικότητες είναι οι παρακάτω:

1) date-based search support,
2) domain-based search support,
3) file type – based search support,
4) size-based search support.

- ***Κλάση ImageFunctionality***
  - ✓ Ίδια λειτουργία, με την διαφορά ότι η μηχανή αναζήτησης πρέπει να αφορά images.

Ομοίως με όλες τις άλλες, προϋποθέτει την προεπιλογή *image*-related search engine. Οι υποστηριζόμενες προσθήκες στην προκειμένη περίπτωση είναι οι εξής:

1) coloration-based search support,
2) domain-based search support,
3) file type – based search support,
4) size-based search support.

- ***Κλάση PaperFunctionality***
  - ✓ Ίδια λειτουργία, με την διαφορά ότι η καινούρια μηχανή πρέπει να αφορά papers.

Ομοίως με τις προηγούμενες, κάνει την εμφάνισή της η αντίστοιχη καρτέλα, αφού ο χρήστης έχει επιλέξει ως χρησιμοποιούμενο τύπο από την εγκαθιστάμενη μηχανή αναζήτησης το *paper*. Οι επεκτάσεις είναι βέβαια και εδώ άλλου είδους και απαριθμούνται αμέσως:

1) author-based search support,
2) date-based search support,
3) publication-based search support,
4) subject-based search support.

- ***Κλάση VideoFunctionality***
  - ✓ Ίδια λειτουργία, με την διαφορά ότι η μηχανή αναζήτησης πρέπει να αφορά video.

Ομοίως με τις υπόλοιπες, παρουσιάζεται στον χρήστη η σχετική καρτέλα, αφού αυτός έχει επιλέξει ως τύπο της προς ενσωμάτωση μηχανής αναζήτησης το *video*. Οι διαφορετικές πρόσθετες λειτουργικότητες που είναι διαθέσιμες σε αυτή την περίπτωση δίνονται ακολούθως:

1) date-based search support,
2) duration-based search support,
3) file type – based search support,
4) size-based search support.

- ***Κλάση WebPageFunctionality***
  - ✓ Ίδια λειτουργία, με την διαφορά ότι η εν λόγω μηχανή πρέπει να αφορά webpages.

Ομοίως με όσες έχουμε αναφέρει έως αυτό το σημείο, με την προϋπόθεση ότι ο χρήστης έχει δηλώσει στο προηγούμενο βήμα την μηχανή αναζήτησης ως κατάλληλη για τον χειρισμό ιστοσελίδων γενικού περιεχομένου (*webpages*). Οι παρεχόμενες επεκτάσεις εδώ συνοψίζονται στις εξής τέσσερεις:

1) date-based search support,
2) domain-based search support,
3) file type – based search support,
4) region-based search support.

- ***Κλάση Verification***
  - ✓ Προτρέπει τον χρήστη να αναθεωρήσει και τελικώς να επικυρώσει τα δεδομένα που έχει μέχρι στιγμής εισαγάγει.

Η καρτέλα που δημιουργείται από την κλάση αυτή, αποτελεί την τελευταία, με την οποία ολοκληρώνεται η εισαγωγή δεδομένων από τον χρήστη μέσω πληκτρολόγησης. Ουσιαστικά πληροφορεί τον χρήστη για τις επιλογές που έκανε και τα στοιχεία που εισήγαγε στα αμέσως προηγούμενα βήματα του οδηγού εγκατάστασης και τον καλεί απλώς να τα επιβεβαιώσει.

- ***Κλάση FeaturesCollector***
  - ✓ Συλλέγει, κατά την διαδικασία εγκατάστασης μίας νέας μηχανής αναζήτησης, όλα τα στοιχεία που χρειάζονται, προκειμένου τελικώς να καταστεί η μηχανή λειτουργική.

Τόσο κατά την διάρκεια των προαναφερθέντων σταδίων της διαδικασίας εγκατάστασης, όσο και κατά την διάρκεια των υπολοίπων, στο background της εφαρμογής δρα η συγκεκριμένη κλάση, συλλέγοντας στο πέρας κάθε βήματος τις απαραίτητες πληροφορίες που είτε

προσέφερε ο χρήστης είτε «έμαθε» - «εξήγαγε» το ίδιο το σύστημα. Οι κύριες μέθοδοι που την απαρτίζουν είναι οι παρακάτω:

1) Μέθοδος *set_Identification*: Λαμβάνει το όνομα και το main URL της εισαγόμενης μηχανής αναζήτησης.
2) Μέθοδος *set_Use_and_Evaluation*: Της είναι ανατεθειμένο το έργο της συλλογής των εξής δύο κρίσιμων πληροφοριών:
    a. Τύπος δεδομένων με τον οποίο σχετίζεται η εν λόγω μηχανή αναζήτησης, και
    b. Βαθμολογία της συγκεκριμένης μηχανής (υποκειμενικό κριτήριο που έχει να κάνει με την δημοφιλία, την αξιοπιστία και γενικότερα την ποιότητα μίας search engine).
3) Μέθοδος *set_Functionality*: Καθορίζει το εάν η μηχανή αναζήτησης υποστηρίζει κάποια από τις τέσσερεις παρεχόμενες επεκτάσεις για τον εκάστοτε βασικό τύπο δεδομένων και σε περίπτωση θετικής απάντησης ενημερώνεται για τις αντίστοιχες τρεις παραμέτρους (script expansion, search filter, query expansion), τις οποίες έχει καταγράψει ο χρήστης.
4) Μέθοδος *set_Search_Tools*: Είναι επιφορτισμένη με τον προσδιορισμό του *base URL*, του *search script*, της *query parameter* και τελικώς του *search URL*, χρησιμοποιώντας τόσο τις πληροφορίες που έχει δώσει ο χρήστης (base URL), όσο και εκείνες που «προέβλεψε» το ίδιο το σύστημα (search script & query parameter). Όπως έχει ήδη προαναφερθεί: “search_URL” = “base_URL” + “/search_script” + “&query_parameter =”.
5) Μέθοδος *set_Result_Extraction_Main_Rules*: Μαζεύει τα τρία μονοπάτια στο HTML δένδρο μίας σελίδας αποτελεσμάτων της εγκαθιστάμενης μηχανής αναζήτησης που αφορούν αντιστοίχως τους τίτλους, τους διαδικτυακούς συνδέσμους και τις περιγραφές.
6) Μέθοδος *set_Result_Extraction_Supplementary_Rules*: Συλλέγει την πληροφορία για τον περιορισμό θέσης του φύλλου (δηλαδή του τελευταίου κόμβου) σε κάθε ένα από τα συζητούμενα τρία μονοπάτια, εφόσον ένας τέτοιος περιορισμός είναι απαραίτητος (για οποιοδήποτε από τα μονοπάτια).
7) Μέθοδος *getSearchEngineFeatures*: Μέσω της κλήσης της γίνεται διαθέσιμο (με την βοήθεια μίας απλής και μεστής δομής δεδομένων) το σύνολο των πληροφοριών της νεοεισηγμένης search engine, που μπορεί να την καταστήσει πλήρως λειτουργική.

- ***Κλάση SE_2_XML_Writer***
    - ✓ Καταγράφει για την εγκαθιστάμενη μηχανή αναζήτησης τις απαραίτητες τιμές των στοιχείων που την καθιστούν λειτουργική σε ένα μοναδικό XML αρχείο.

Η κλάση Search_Engine-to-XML-Writer, σε συνεργασία με την προηγούμενη, καταγράφει το ελάχιστο απαραίτητο σύνολο δεδομένων, το οποίο χρειάζεται και επαρκεί, προκειμένου η προς εγκατάσταση μηχανή αναζήτησης να μπορεί να γίνει λειτουργήσιμη και να ενσωματωθεί αρμονικά στην εφαρμογή μας. Το παραγόμενο XML αρχείο είναι λιτό και σύντομο, ενώ υπακούει σε ένα αυστηρό και σαφές DTD* αρχείο, το οποίο καθορίζει σε απόλυτο βαθμό την μορφή του. Πάνω σε τέτοια XML αρχεία, που αποθηκεύονται σε προκαθορισμένη τοποθεσία του συστήματός μας, βασίζεται η λειτουργία κάθε wrapped search engine. Είναι αυτά που χρησιμοποιεί το υποσύστημα διαδικτυακής αναζήτησης για να εντάξει τις user-defined search engines, με απώτερο σκοπό την αγαστή συνεργασία τους με τις API-embedded search engines και τελικά την προσφορά κοινής λίστας ταξινομημένων ως προς την σημαντικότητα αποτελεσμάτων.

* Σημείωση: DTD = *Document Type Definition*, γλώσσα ορισμού των αποδεκτών δομικών τμημάτων ενός XML αρχείου. Καθορίζει την δομή του εγγράφου μέσω μίας λίστας «νομίμων» στοιχείων (elements), χαρακτηριστικών (attributes) και τιμών (values) αυτών. Αποτελεί μία από τις αρκετές SGML και XML schema γλώσσες, που χρησιμεύουν για το εν λόγω έργο.

4.2.5 Πακέτο κλάσεων scrapemode

- Αφορά την διαδικασία του wrapping, στα πλαίσια εγκατάστασης μίας καινούριας μηχανής αναζήτησης στο σύστημά μας.

Το συγκεκριμένο πακέτο απαρτίζεται από το σύνολο των κλάσεων που αφορούν το screen-scraping τμήμα της διαδικασίας εγκατάστασης μίας νέας μηχανής αναζήτησης, καθώς και από την κλάση που ενσωματώνει έναν φυλλομετρητή ιστοσελίδων στην εφαρμογή μας.

- ***Κλάση WebRenderer***
  - ✓ Συνιστά έναν εσωτερικό φυλλομετρητή ιστοσελίδων για την εφαρμογή μας.

Είναι η εν λόγω κλάση εκείνη η οποία υλοποιεί την προσθήκη ενός Mozilla-based / Firefox-clone web-browser στο σύστημα. Πρόκειται για έναν πλήρως λειτουργικό browser, ο οποίος δεν έχει τίποτα να ζηλέψει από τους συνήθεις δημοφιλείς, όπως ο Microsoft Internet Explorer, ο Mozilla Firefox, ο Apple Safari, ο Google Chrome ή ο Opera Browser. Ο συζητούμενος φυλλομετρητής, παρέχει μέσω μίας address bar – text field, την δυνατότητα στον χρήστη να πληκτρολογήσει το URL που επιθυμεί να επισκεφθεί. Επίσης, του προσφέρει τις λειτουργικότητες "επίσκεψης της προηγούμενης (στο ιστορικό) σελίδας" (μέσω ενός *back-button*), "επίσκεψης της επόμενης (στο ιστορικό) σελίδας" (μέσω ενός *forward-button*), "διακοπής φόρτωσης της σελίδας" (μέσω ενός *stop-button*) και "ανανέωσης της τρέχουσας σελίδας" (μέσω ενός *refresh-button*). Ακόμα, έχουμε δημιουργήσει μία *progress bar* (που μας δείχνει προοδευτικά το φόρτωμα της εκάστοτε ιστοσελίδας, τόσο σχηματικά όσο και υπό την

μορφή ποσοστού), όπως και μία *status bar* (η οποία μας πληροφορεί για την τρέχουσα κατάσταση, π.χ. "Loading…", "Done" κλπ). Υπενθυμίζουμε, ότι ο χρησιμοποιούμενος browser υποστηρίζει όλα τα σύγχρονα πρωτόκολλα και τις τεχνολογίες που χρειάζονται και οι πλέον απαιτητικές ιστοσελίδες προκειμένου να απεικονιστούν απρόσκοπτα.

- ***Κλάση InstallationWebRenderer***
  - ✓ Πρόκειται για μία εξειδικευμένη παραλλαγή του χρησιμοποιούμενου web-browser, ο οποίος έχει πλέον και δυνατότητες πρόγνωσης search scripts και query parameters, ενώ επίσης είναι σε θέση να αξιοποιεί την τεχνική του screen-scraping για την μηχανική εκμάθηση ανάγνωσης αποτελεσμάτων και των δομικών στοιχείων τους.

Πρόκειται για μία αρκετά εξειδικευμένη έκδοση της προηγούμενης κλάσης, η οποία έχει εμπλουτιστεί με μία πληθώρα λειτουργικών χαρακτηριστικών, με απώτερο σκοπό να διαδραματίσει τον ρόλο του συντονιστή της wrapping process, καθώς και εκείνον της διαπροσωπείας για την αμφίδρομη επικοινωνία μεταξύ χρήστη και μηχανής. Η εν λόγω κλάση επικοινωνεί με τις ακόλουθες κλάσεις: 1) HTMLParser, 2) SearchScript_Query Parameter_Oracle και 3) WrapperSpecifications. Ουσιαστικά, κατασκευάζει έναν web-browser, ο οποίος συν τοις άλλοις επιτρέπει στον χρήστη να διαλέξει το search script και την query parameter από μία παρεχόμενη λίστα με αντίστοιχα ζεύγη που έχουν προβλεφθεί ή να δώσει το δικό του ζευγάρι, εκτελεί αυτομάτως μία συνήθη αναζήτηση με την εγκαθιστάμενη μηχανή αναζήτησης (βάσει προκαθορισμένης λέξης-κλειδί), εμφανίζει την επιστρεφόμενη σελίδα αποτελεσμάτων και καλεί τον χρήστη να επιλέξει (μέσω mouse-highlighting) και να «δηλώσει» έναν τίτλο αποτελέσματος (result title), ένα διαδικτυακό σύνδεσμο αποτελέσματος (result link / URL) και μία περίληψη / περιγραφή αποτελέσματος (result summary / description). Σημειώνουμε εδώ, ότι είναι δυνατόν να παρακαμφθεί το σκέλος που αφορά τα links των αποτελεσμάτων, καθώς αυτά μπορούν (εκτός εξαιρέσεων) να εκμαιευθούν και από τον υπερσύνδεσμο προς το εκάστοτε URL που κατά κανόνα φιλοξενείται στον τίτλο του αντίστοιχου αποτελέσματος. Επιπροσθέτως, τονίζουμε ότι σε οποιαδήποτε φάση της όλης διαδικασίας, αυτή μπορεί να ακυρωθεί, ενώ αφού έχουν ολοκληρωθεί όλα τα βήματα, ο χρήστης πρέπει να προβεί σε μία οριστική επικύρωσή της, σημαίνοντας έτσι την επιτυχή ενσωμάτωση της καινούριας search engine στην εφαρμογή, η οποία εφόσον όλα τα βήματα έχουν γίνει με τον σωστό τρόπο θα λειτουργεί πλέον πλήρως και σε αρμονία με τις υπόλοιπες ήδη υπάρχουσες μηχανές αναζήτησης.

- ***Κλάση HTMLParser***
  - ✓ Αποτελεί τον πυρήνα της wrapping process, όντας επιφορτισμένη με το parsing της HTML σελίδας των αποτελεσμάτων της εγκαθιστάμενης μηχανής αναζήτησης, την κατασκευή του σχετικού μοτίβου μονοπατιού επί του HTML δένδρου και τις απαραίτητες χρωματικές επισημάνσεις των ανάλογων τμημάτων της ιστοσελίδας

(είτε από την πλευρά του χρήστη, είτε από την πλευρά του συστήματος) για την επιτυχή διεκπεραίωση του interactive machine learning.

Η συγκεκριμένη κλάση είναι η ζωτικότερη για την διαδικασία του screen-scraping. Αυτή έχει στην διάθεσή της μονάχα τον HTML source code μία σελίδας αποτελεσμάτων της προς εγκατάσταση μηχανής αναζήτησης και την συμβολοακολουθία του επιλεχθέντος από τον χρήστη τμήματος της εν λόγω ιστοσελίδας, που προέρχονται από την αμέσως παραπάνω κλάση. Έτσι, πρέπει να οικοδομήσει την αντίστοιχη νοητή δενδρική δομή του HTML κώδικα, να εντοπίσει σε αυτήν το ενδιαφέρον string και να ανακαλύψει ολόκληρο το μονοπάτι (path) στο δένδρο (tree) από το φύλλο (leaf) αυτό ως την ρίζα (root) του. Η ίδια διαδικασία επαναλαμβάνεται τρεις φορές: 1) μία για την καταγραφή της διαδρομής στο δένδρο σχετικά με τίτλους αποτελεσμάτων της νεοεισαγόμενης μηχανής αναζήτησης, 2) μία για την εύρεση του μονοπατιού στην δενδρική δομή των URL των αποτελεσμάτων και 3) μία φορά για τον εντοπισμό της διαδρομής στο HTML δένδρο των περιγραφών των αποτελεσμάτων. Άξια προσοχής είναι η ικανότητα του συστήματός μας, να μπορεί να βρει τα απαραίτητα μονοπάτια, ακόμα και αν ο χρήστης δεν έχει επιλέξει ένα ολόκληρο στοιχείο (δηλαδή τίτλο / URL / περιγραφή), αλλά τμήμα αυτού. Η επιτυχία αυτή της εφαρμογής οφείλεται στην χρήση *κανονικών εκφράσεων* (*regular expressions*). Βέβαια, όπως γίνεται εύκολα κατανοητό, η εκάστοτε mouse-selection, πρέπει να έχει τουλάχιστον κάποιο ελάχιστο μήκος (ορισμένο όχι αυστηρά, αλλά από την κοινή λογική), καθώς διαφορετικά το σύστημα θα οδηγηθεί προφανώς σε λάθη, υπονομεύοντας την ορθή ολοκλήρωση της εγκατάστασης. Επίσης, πρέπει να αναφερθεί ο διαχωρισμός των ετικετών (tags) του HTML document σε δομικά (structural) και μη-δομικά (non-structural), καθώς και σε μορφοποίησης (stylish) και στα αντίθετά τους (non-stylish). Οι συγκεκριμένες ομαδοποιήσεις εξυπηρετούν στην κατασκευή του «πραγματικού» HTML δένδρου, καθώς και στην υπέρβαση προβλημάτων που έχουν να κάνουν με tags που εμφανίζονται εντός του επιλεχθέντος string. Στην υλοποίηση της συζητούμενης κλάσης, αξιοποιείται η κλάση FlagList (σε επίπεδο runtime), ενώ κατά την κατασκευή της χρησιμοποιήθηκε και η κλάση RegExTester (σε επίπεδο development δηλαδή).

- ***Κλάση FlagList***
  - ✓ Ορίζει έναν νέο τύπο δεδομένων, που απαρτίζεται από έναν δυναμικό πίνακα HTML στοιχείων και έναν απλό πίνακα λογικών μεταβλητών.

Ο ρόλος της παρούσας κλάσης είναι καθαρά επικουρικός προς την προηγούμενη, καθώς αρκείται στον ορισμό ενός νέου τύπου δεδομένων, εκείνου του FlagList. Πρόκειται για μία δομή δεδομένων που περιέχει έναν vector από HTML elements και έναν boolean array, οι μονάδες του οποίου λειτουργούν ως «σημαίες», για την διάκριση μεταξύ δύο καταστάσεων. Με την βοήθεια της δομής FlagList, καθίσταται δυνατός ο διαχωρισμός των HTML

στοιχείων σε δομικά και μη-δομικά, ενώ πλέον το κάθε στοιχείο συνοδεύεται από τον αντίστοιχο χαρακτηρισμό.

- ***Κλάση RegExTester***
  - ✓ Βοηθά στην κατασκευή και τον έλεγχο ορθής λειτουργίας της εκάστοτε επιθυμητής κανονικής έκφρασης.

Η κλάση αυτή δεν ενσωματώνεται άμεσα στην εφαρμογή μας, ωστόσο αποτελεί μία κλάση ελέγχου κανονικών εκφράσεων και αποδείχθηκε ιδιαίτερα χρήσιμη στην διασαφήνιση εκείνων των regular expressions που επιτελούν ακριβώς το εκάστοτε έργο για το οποίο προορίζονται ως φίλτρα συμβολοακολουθιών. Η λειτουργία της συνοψίζεται στα εξής: Δοθέντος ενός string pattern, επιχειρείται matching μεταξύ αυτού και ενός test string. Είτε σε περίπτωση θετικής απάντησης είτε αρνητικής, ο προγραμματιστής ενημερώνεται για το αποτέλεσμα με το αντίστοιχο κάθε φορά μήνυμα.

- ***Κλάση SearchScript_QueryParameter_Oracle***
  - ✓ Επιχειρεί την πρόβλεψη του search script και της query parameter, που σε συνδυασμό με το δοθέν base URL, μπορούν να συνθέσουν το ζητούμενο search URL.

Η εν λόγω κλάση είναι επιφορτισμένη με δύο σημαντικά έργα: 1) Την πρόγνωση του *search script*, που υλοποιείται μέσω της μεθόδου searchScriptGuess(), και 2) την πρόβλεψη της *query parameter*, κάτι το οποίο συντελείται με την βοήθεια της μεθόδου queryParameter Guess(). Υπενθυμίζουμε στο σημείο αυτό ότι το ζητούμενο *search URL* συνίσταται από το παρεχόμενο από τον ίδιο τον χρήστη *base URL*, σε συνδυασμό με τα προηγούμενες δύο παραμέτρους, δηλαδή: search_URL = base_URL / search_script & query_parameter = *keyword*. Η πρόγνωση του search script και της query parameter επιτυγχάνεται αφού το σύστημα επισκεφθεί το main URL της ενσωματούμενης μηχανής αναζήτησης που έδωσε ο χρήστης στην πρώτη καρτέλα του οδηγού εγκατάστασης, και στην συνέχεια, εφόσον λάβει τον HTML κώδικα που αντιστοιχεί στην εν λόγω ιστοσελίδα, εντοπίζοντας ως values συγκεκριμένων fields τις πιθανές τιμές των δύο ζητούμενων παραμέτρων. Σημειώνουμε εδώ, ότι οι πολλαπλές καταχωρήσεις των ίδιων ενδεχόμενων τιμών παραβλέπονται, κρατώντας τελικά μόνο την μία εξ αυτών.

- ***Κλάση SearchScript_QueryParameter_Chooser***
  - ✓ Παρουσιάζει τα ευρεθέντα ζεύγη search scripts – query parameters και ενδεχομένως συνιστά κάποιο, ενώ επίσης υπάρχει μέριμνα για την καταγραφή ενός άλλου ζευγαριού που πιθανώς να επιθυμεί να ορίσει ο ίδιος ο χρήστης.

Αφορά την δημιουργία και την εμφάνιση στον χρήστη της ανάλογης καρτέλας, αφού ολοκληρωθεί η πρόβλεψη των search script & query parameter. Όλα τα υποψήφια ζεύγη παρουσιάζονται ως στοιχεία δύο combo-boxes (drop-down lists), ενώ εφόσον κάποιο από

αυτά υπάρχει καταχωρημένο στο σύστημά μας ως σύνηθες, μαρκάρεται ως συνιστώμενο (recommended). Επίσης, δίνεται η δυνατότητα στον χρήστη να εισάγει το δικό του (custom) ζευγάρι search script – query parameter, εφόσον ο ίδιος διαθέτει τέτοιου επιπέδου γνώσεις και οι παρεχόμενες επιλογές είτε δεν τον καλύπτουν είτε δεν τις θεωρεί σωστές.

- ***Κλάση WrapperSpecifications***
  - ✓ Περιλαμβάνει ένα σύνολο προηγμένων τεχνικών βελτίωσης της ικανότητας εξόρυξης δεδομένων από τον wrapper, σε περίπτωση που η βασική μέθοδος δεν επαρκεί.

Η συγκεκριμένη κλάση δημιουργεί ένα μικρό frame, το οποίο εμφανίζεται σε κάθε ένα από τα τρία στάδια απόξεσης των αποτελεσμάτων από την οθόνη (τίτλος – URL – περιγραφή). Καλεί τον χρήστη να επικυρώσει την επιτυχία της προσπάθειας του συστήματος να «μαντέψει», βάσει μόνο ενός επιλεχθέντος τμήματος αποτελέσματος, τα υπόλοιπα αντίστοιχα τμήματα των άλλων αποτελεσμάτων. Βέβαια, δίνεται η δυνατότητα στον χρήστη να χαρακτηρίσει την απόπειρα της εφαρμογής αποτυχημένη, ακυρώνοντας παράλληλα έτσι το τελευταίο βήμα της διαδικασίας εγκατάστασης και συνεχίζοντας από την ακριβώς προηγούμενη φάση, ενδεχομένως προσπαθώντας ξανά. Εκτός της απόλυτης επιτυχίας ή της αποτυχίας, υπάρχει και η περίπτωση που το σύστημα «προβλέπει» με μερικώς σωστό τρόπο τα αποτελέσματα. Σε ένα τέτοιο ενδεχόμενο, μπορεί να απαιτείται μία μοναδική και μικρή (όμως σημαντική) τροποποίηση στον τρόπο διαχείρισης των αποτελεσμάτων. Αυτή η διαφοροποίηση αφορά την χρήση περιορισμού θέσης στο φύλλο του HTML δένδρου (κάτι το οποίο δεν αποτελεί την προεπιλογή), δηλαδή, σε περίπτωση που μαρκάρονται περισσότερα στοιχεία από τα επιθυμητά, μπορούμε να απαλείψουμε τα πρόσθετα στοιχεία, επιβάλλοντας στο σύστημα την επιλογή ενός συγκεκριμένου leaf (και όχι αυτού μαζί με τα υπόλοιπα siblings) ως τερματικού κόμβου στο μονοπάτι επί του HTML δένδρου, προκειμένου η εν λόγω διαδρομή να εξειδικεύεται κατά ένα βήμα περαιτέρω και να φιλτράρει με τον τρόπο αυτό τα τελικά στοιχεία. Άλλος τρόπος που μπορεί να γίνει η συζητούμενη τροποποίηση κατά την screen-scraping process είναι μέσω της επιλογής μίας regular expression, με την βοήθεια μίας λίστας από έτοιμες regexes που προσφέρονται σε ετούτο το διαδραστικό παράθυρο. Η κανονική έκφραση που θα διαλέξει ο χρήστης, θα εφαρμοστεί στα στοιχεία που χαρακτηρίζονται ως τελικά, προκειμένου να αποκόψει όσα θεωρούνται βάσει αυτής ακατάλληλα. Και στην περίπτωση του περιορισμού θέσης φύλλου, και σε εκείνη των κανονικών εκφράσεων, πρέπει να δοθεί εντολή στο σύστημα να προβεί σε re-wrapping (στα πλαίσια του εκάστοτε τμήματος αποτελεσμάτων), ο χρήστης στην συνέχεια να δει τα νέα μαρκαρισμένα στοιχεία επί της result page και ακολούθως είτε να αποφασίσει να επικυρώσει το συγκεκριμένο βήμα της διαδικασίας ενσωμάτωσης, είτε να το απορρίψει, είτε ενδεχομένως να εφαρμόσει και άλλον επιπρόσθετο περιορισμό. Τα προαναφερθέντα βήματα πρέπει να ακολουθηθούν για κάθε ένα από τα τρία τμήματα ενός αποτελέσματος αναζήτησης, με

απώτερο σκοπό η όλη διαδικασία να ολοκληρωθεί ομαλά και έτσι η νεοεγκατεστημένη μηχανή να λειτουργεί με απρόσκοπτο τρόπο. Υπενθυμίζουμε στο σημείο αυτό, ότι κυρίως λόγω της επιτρεπόμενης σχετικής χαλαρότητας από το HTML πρότυπο κατά την σύνταξη αντιστοίχων εγγράφων, καθώς και εξαιτίας της μεγάλης ποικιλίας εμφάνισης των σελίδων αποτελεσμάτων από τις διάφορες μηχανές αναζήτησης, δεν είναι δυνατή η κατασκευή ενός screen-scraping συστήματος, το οποίο να είναι ικανό να αντιμετωπίσει οποιαδήποτε περίπτωση. Έτσι, η ορθή λειτουργία της παρούσας εφαρμογής είναι εγγυημένη με την προϋπόθεση ότι η επιστρεφόμενη λίστα αποτελεσμάτων από την εκάστοτε μηχανή αναζήτησης είναι καλώς δομημένη, ακολουθεί το ίδιο μοτίβο για κάθε αποτέλεσμα και δεν αποκλίνει σημαντικά από το HTML πρότυπο.

4.3 Περιγραφή της Βάσης Δεδομένων του DBLP

Η Βάση Δεδομένων που συμπεριλαμβάνει η εφαρμογή μας αποτελεί ουσιαστικά έναν κλώνο της αντίστοιχης online του *DBLP* και αφορά τις επιστημονικές δημοσιεύσεις στον τομέα της Επιστήμης των Υπολογιστών. Στην ενότητα αυτή, θα επιχειρήσουμε να εξηγήσουμε συνοπτικά το σχήμα της ΒΔ που προκύπτει από το E-R μοντέλο του προηγούμενου κεφαλαίου.

Ένα paper μπορεί να είναι τύπου article, inproceedings, proceedings, book, incollection, phdthesis, mastersthesis ή www, και έτσι ομιλούμε περί *εξειδίκευσης* του εν λόγω αντικειμένου σε πιο συγκεκριμένες υποκατηγορίες. Επειδή το κάθε paper υπάγεται αναγκαστικά σε μία από τις παρεχόμενες ομαδοποιήσεις, πρόκειται για *συνολική* εξειδίκευση, ενώ εφόσον η κάθε μία επιστημονική δημοσίευση μπορεί να υπάγεται μόνο σε μία από τις κατηγορίες αυτές, οι τελευταίες είναι *ξένες* μεταξύ τους.

Τα papers, ανεξαρτήτως τύπου, διαθέτουν όλα κάποιες κοινές *ιδιότητες*, και εν προκειμένω τις εξής: 1) title (συνιστά το *κλειδί*, δηλαδή εκείνη την ιδιότητα που απαιτείται και συγχρόνως επαρκεί για την πλήρη διασαφήνιση και διάκριση των διαφόρων οντοτήτων της βάσης δεδομένων μας μεταξύ τους), 2) date added (*παραγόμενη ιδιότητα*, καθώς προκύπτει από την ίδια την ΒΔ και δεν απαιτείται εισαγωγή της τιμής της), 3) type (είναι η ιδιότητα, για την οποία έγινε λόγος παραπάνω και σχετίζεται με τα υπάρχοντα υποσύνολα των papers).

Ας ασχοληθούμε τώρα με τις οντότητες, *γενίκευση* των οποίων αποτελεί μία επιστημονική δημοσίευση. Ένα article περιέχει τις παρακάτω ιδιότητες: 1) page region, 2) year, 3) volume, 4) journal, 5) number, 6) web-site, 7) DBLP URL και 8) author (*πολλαπλή ιδιότητα*, καθώς λαμβάνει ενδεχομένως περισσότερες της μίας τιμές). Επίσης, μία οντότητα inproceedings περιλαμβάνει τις ακόλουθες ιδιότητες: 1) page region, 2) year, 3) web-site, 4) DBLP URL, 5) conference & 6) author (πολλαπλή ιδιότητα). Ακόμα, μία οντότητα proceedings διαθέτει τις

εξής ιδιότητες: 1) year, 2) volume, 3) DBLP URL, 4) conference, 5) ISBN, 6) series, 7) series URL, 8) publisher, 9) editor (πολλαπλή ιδιότητα). Επιπροσθέτως, ένα book έχει τις κάτωθι ιδιότητες: 1) year, 2) DBLP URL, 3) ISBN, 4) publisher, 5) publisher URL, 6) editor (πολλαπλή ιδιότητα). Επιπλέον, μία οντότητα τύπου incollection χαρακτηρίζεται από τις εξής ιδιότητες: 1) DBLP URL, 2) conference, 3) page region, 4) year, 5) web-site, 6) author (πολλαπλή ιδιότητα). Εκτός αυτών, μία phdthesis διακρίνεται από τις ακόλουθες ιδιότητες: 1) year, 2) institution, 3) author (πολλαπλή ιδιότητα). Επίσης, μία mastersthesis περιλαμβάνει τις εξής ιδιότητες: 1) year, 2) institution, 3) author (πολλαπλή ιδιότητα). Τέλος, μία οντότητα τύπου www περιέχει τις παρακάτω ιδιότητες: 1) year, 2) DBLP URL και 3) author (πολλαπλή ιδιότητα).

4.4 Κωδικοποίηση αρχείων

Οι ενσωματούμενες από τον χρήστη μηχανές αναζήτησης καταγράφουν την απαραίτητη για την λειτουργία τους πληροφορία σε ένα σύντομο και περιεκτικό XML αρχείο η κάθε μία, το οποίο υπακούει στους γενικούς κανόνες που υπαγορεύει ένα λιτό, σαφές και αυστηρό DTD έγγραφο. Έτσι, κάθε wrapped search engine υλοποιείται μέσω του αντίστοιχου XML document, την ανάγνωση και πλήρη εκμετάλλευση του οποίου γνωρίζει να επιτελεί η εφαρμογή μας. Παραθέτουμε ακολούθως ένα παράδειγμα μίας τέτοιου είδους μηχανής αναζήτησης (πιο συγκεκριμένα της Gigablast Web Search Engine), καθώς και το εν λόγω αρχείο προδιαγραφών στην συνέχεια.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="Validator.xslt"?>
<search_engine>
<identification name="Gigablast" url="http://www.gigablast.com">
</identification>
<service>
<web_page rank="2">
<date_based_search_support script_expansion="&dt=" filter="date:" query_expansion=
"date">
</date_based_search_support>
<domain_based_search_support script_expansion="&dm=" filter="domain:" query_expansion=
"domain">
</domain_based_search_support>
<file_type_based_search_support script_expansion="&fltp=" filter="filetype:" query_
expansion="file type">
</file_type_based_search_support>
<region_based_search_support script_expansion="&rg=" filter="region:" query_expansion=
"region">
```

```
</region_based_search_support>
</web_page>
</service>
<implementation>
<search script="/index.php" parameter="q" url="http://www.gigablast.com/index.php?q=">
</search>
<title_extraction path="html=>body=>div=>table=>tbody=>tr=>td=>table=>tbody=>tr=>td=>
table=>tbody=>tr=>td=>a" position="-1">
</title_extraction>
<link_extraction path="html=>body=>div=>table=>tbody=>tr=>td=>table=>tbody=>tr=>td=>
table=>tbody=>tr=>td=>span" position="0">
</link_extraction>
<summary_extraction
path="html=>body=>div=>table=>tbody=>tr=>td=>table=>tbody=>tr=>td=>table=>tbody=>tr=>
td=>div" position="-1">
</summary_extraction>
</implementation>
</search_engine>
```

Παράθεση 4.1 – Παράδειγμα του αντίστοιχου XML αρχείου μίας wrapped search engine

```
<?xml version="1.0" encoding="UTF-8"?>
<!-edited with Altova XMLSpy →
<!ENTITY % STRING "CDATA">
<!ENTITY % URI "CDATA">
<!ENTITY % EVAL_INT "CDATA">
<!ELEMENT Search_Engine (Identification, Service)>
<!ELEMENT Service (web_page, blog, document, paper, map, image, audio, video)>
<!ELEMENT web_page (date_support?, domain_support?, file_type_support?,
region_support?)>
<!ELEMENT blog (blog_search_support?, tag_search_support?, post_search_support?,
url_search_support?)>
<!ELEMENT document (date_support?, domain_support?, file_type_support?,
size_support?)>
<!ELEMENT paper (author_support?, date_support?, publication_support?,
subject_support?)>
<!ELEMENT image (coloration_support?, domain_support?, file_type_support?,
size_support?)>
<!ELEMENT audio (bitrate_support?, duration_support?, file_type_support?,
genre_support?)>
<!ELEMENT video (date_support?, duration_support?, file_type_support?, size_support?)>
<!ATTLIST Identification name %STRING; #REQUIRED url %URI; #REQUIRED>
<!ATTLIST web_page rank %EVAL_INT; #REQUIRED>
```

```
<!ATTLIST blog rank %EVAL_INT; #REQUIRED>
<!ATTLIST document rank %EVAL_INT; #REQUIRED>
<!ATTLIST paper rank %EVAL_INT; #REQUIRED>
<!ATTLIST map rank %EVAL_INT; #REQUIRED>
<!ATTLIST image rank %EVAL_INT; #REQUIRED>
<!ATTLIST audio rank %EVAL_INT; #REQUIRED>
<!ATTLIST video rank %EVAL_INT; #REQUIRED>
<!ATTLIST date_support query %STRING; "null" expansion %STRING; "null" keyword
%STRING; "null">
<!ATTLIST domain_support query %STRING; "null" expansion %STRING; "null" keyword
%STRING; "null">
<!ATTLIST file_type_support query %STRING; "null" expansion %STRING; "null" keyword
%STRING; "null">
<!ATTLIST region_support query %STRING; "null" expansion %STRING; "null" keyword
%STRING; "null">
<!ATTLIST blog_search_support query %STRING; "null" expansion %STRING; "null" keyword
%STRING; "null">
<!ATTLIST tag_search_support query %STRING; "null" expansion %STRING; "null" keyword
%STRING; "null">
<!ATTLIST post_search_support query %STRING; "null" expansion %STRING; "null" keyword
%STRING; "null">
<!ATTLIST url_search_support query %STRING; "null" expansion %STRING; "null" keyword
%STRING; "null">
<!ATTLIST size_support query %STRING; "null" expansion %STRING; "null" keyword
%STRING; "null">
<!ATTLIST author_support query %STRING; "null" expansion %STRING; "null" keyword
%STRING; "null">
<!ATTLIST publication_support query %STRING; "null" expansion %STRING; "null" keyword
%STRING; "null">
<!ATTLIST subject_support query %STRING; "null" expansion %STRING; "null" keyword
%STRING; "null">
<!ATTLIST coloration_support query %STRING; "null" expansion %STRING; "null" keyword
%STRING; "null">
<!ATTLIST bitrate_support query %STRING; "null" expansion %STRING; "null" keyword
%STRING; "null">
<!ATTLIST duration_support query %STRING; "null" expansion %STRING; "null" keyword
%STRING; "null">
<!ATTLIST genre_support query %STRING; "null" expansion %STRING; "null" keyword
%STRING; "null">
```

Παράθεση 4.2 – DTD αρχείο για XML έγγραφα των wrapped search engines

5

Ειδικά Θέματα Υλοποίησης

Στο παρόν κεφάλαιο θα συζητηθούν ενδελεχώς θέματα υλοποίησης του συστήματος, τα οποία είτε χρίζουν ιδιαίτερης προσοχής είτε εμφανίζουν αξιοσημείωτο ενδιαφέρον. Πιο συγκεκριμένα, θα παρουσιάσουμε τις σημαντικότερες λεπτομέρειες των εν λόγω θεμάτων, ενώ επίσης θα γίνει αναφορά στις χρησιμοποιηθείσες πλατφόρμες ανάπτυξης και εκτέλεσης της εφαρμογής, καθώς και στα προγραμματιστικά εργαλεία που αξιοποιήθηκαν.

5.1 Λεπτομέρειες υλοποίησης

Στην ενότητα αυτή περιγράφονται λεπτομερώς ζητήματα της διπλωματικής εργασίας που μπορούν να χαρακτηριστούν ως ενδιαφέροντα από τεχνικής ή αλγοριθμικής πλευράς. Για κάθε ένα τέτοιο θέμα, αναδεικνύεται το προκύψαν πρόβλημα, η επιλεχθείσα μέθοδος αντιμετώπισής του και ο αντίστοιχος τρόπος υλοποίησης, συνοδευόμενος από τα σχετικά τμήματα ψευδοκώδικα ή / και σχήματα.

5.1.1 Εύρεση μοτίβου πλήρους μονοπατιού σε HTML δένδρο, με γνώση μόνο ενός τελικού κόμβου μίας τέτοιας διαδρομής

Η wrapping process, στην οποία έχουμε προαναφερθεί ήδη πολλάκις και συνιστά την καρδιά του υποσυστήματος ενσωμάτωσης μίας νέας μηχανής αναζήτησης στην εφαρμογή μας, βασίζει την λειτουργία της στο κατάλληλο HTML parsing [Fre98]. Εκείνο που επιθυμούμε

είναι να μπορεί το σύστημα να «μάθει» να διαβάζει τα αποτελέσματα από την εκάστοτε σχετική λίστα που επιστρέφει μία search engine, με τον άνθρωπο να επεμβαίνει μόνο κατά την εγκατάσταση, υποδεικνύοντας ένα result και τα τρία βασικά συστατικά του μέρη (title – URL – description). Όταν λοιπόν ο χρήστης επιλέξει (μέσω mouse-highlighting) ένα τέτοιο result subelement, πρέπει να αναλογιστούμε, ότι αυτό αντιστοιχεί σε κάποιο τμήμα του HTML κώδικα της εν λόγω ιστοσελίδας. Λαμβάνοντας το HTML source και εντοπίζοντας το μαρκαρισμένο κείμενο μέσα σε αυτό, μπορούμε, οικοδομώντας την αντίστοιχη δενδρική δομή, να βρούμε το πλήρες μονοπάτι του συγκεκριμένου στοιχείου επί της τελευταίας. Η λογική της συζητούμενης μεθοδολογίας συνοψίζεται στο γεγονός ότι όλα τα όμοια υποστοιχεία αποτελεσμάτων, θα βρίσκονται σε paths της ίδιας ακριβώς μορφής, ανάγοντας έτσι σε τελικό μας στόχο την εύρεση του μοτίβου (pattern) του ενδιαφέροντος μονοπατιού [AGM03]. Ετούτη η διαδικασία μπορεί εν συνεχεία να ακολουθηθεί για κάθε ένα από τα τρία result subelements, ενώ το όφελος εν τέλει είναι πως θα έχουμε καταφέρει να «διδάξουμε» πλέον στο σύστημά μας να αναγιγνώσκει ολόκληρη την σελίδα αποτελεσμάτων της εγκαθιστάμενης μηχανής αναζήτησης, καθιστώντας την έτσι λειτουργική για οποτεδήποτε αυτή κληθεί να αποζητήσει πληροφορίες στο Διαδίκτυο.

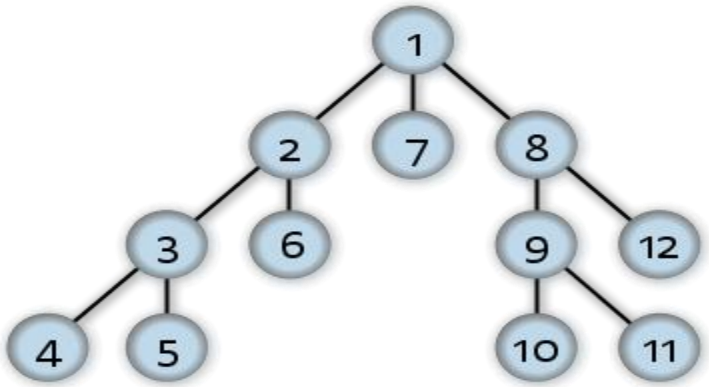
5.1.1.1 Κατασκευή της εικονικής δενδρικής δομής μίας HTML σελίδας

Για την δημιουργία του HTML δένδρου έγινε χρήση ενός από τα πιο διάσημα open source tools στον τομέα αυτό, του *Jericho HTML Parser*. Το συγκεκριμένο εργαλείο δεν βασίζει την λειτουργία του ούτε αποκλειστικά στο *DOM* (Document – Object Model) [WP6], ούτε στο *SAX* (Simple API for XML) [WP7], δηλαδή δεν αποτελεί ούτε έναν αμιγώς tree-based parser, ούτε έναν event-based parser, αλλά είναι ένα υβρίδιο και των δύο μορφών, συνδυάζοντας τα πλεονεκτήματά τους, και επιχειρώντας μέσα από τον εν λόγω συγκερασμό να υπερβεί τις αδυναμίες των δύο μοντέλων.

5.1.1.2 Διάσχιση και αναζήτηση στο νοητό HTML δένδρο

Η HTML δενδρική δομή δεν παύει να συνιστά και η ίδια ένα δένδρο, όπως αυτό ορίζεται στην Θεωρία Γράφων (Graph Theory). Επομένως, αλγοριθμικά μπορεί και πρέπει να αντιμετωπιστεί ως τέτοιο. Η διάσχιση (traversing) του HTML δένδρου γίνεται βάσει της πασίγνωστης μεθόδου που υπακούει στον κανόνα «βάθος-πρώτα». Ο αλγόριθμος που χρησιμοποιείται για την αναζήτηση επί του HTML tree είναι ο DFS (depth-first search algorithm), ο οποίος βασίζεται στην προαναφερθείσα τεχνική [WP8]. Έτσι, η αναζήτηση για την εύρεση του ζητούμενου στο HTML δένδρο στηρίζεται στην εν λόγω μέθοδο διάσχισης, με την μόνη διαφοροποίηση ότι η «σάρωση» των nodes του δένδρου παύει, μόλις βρεθεί ο ενδιαφέρων κόμβος. Στον κόμβο αυτόν απαντάται το επιλεχθέν από τον χρήστη κείμενο και

είναι ουσιαστικά ο τελευταίος στο μονοπάτι που προσπαθούμε να προσδιορίσουμε. Αμέσως παρακάτω δείχνουμε σχηματικά την depth-first διάσχιση:



Σχήμα 5.1 – DFS tree traversing

5.1.1.3 Σχετικός αλγόριθμος υλοποίησης του DFS

Στην συνέχεια παραθέτουμε τον ψευδοκώδικα που αναδεικνύει την βασική ιδέα πραγματοποίησης της όλης ζητούμενης διαδικασίας. Σημειώνουμε ότι το τμήμα που αφορά τον αλγόριθμο DFS υλοποιήθηκε με την βοήθεια μίας δομής δεδομένων τύπου στοίβας (stack).

```
PSEUDOCODE()
   HTML_Tree T := get tree from (get HTML from 'result web page');
   String y := 'text marked by user';
   HTML_Path(T,y)
      Vector Q := {};
      HTML_Element r = get element from T at 0;
      DFS(T,r)
         Stack S := {};
         for each vertex u, set visited[u] := false;
         push r in S;
         while (S is not empty) do
            v := pop S;
            if (not visited[u]) then
               visited[v] := true;
               if (text[v] = y)
                  add v to Q;
                  for each parent node y of v, add y to Q at i;
                  break;
               end if
               for each unvisited sibling w of v, push w in S;
            end if
         end while
      END DFS();
      Reverse(Q)
         Vector P := {};
         for each element z of Q, add z to P at (size[Q]-j-1);
      END Reverse();
   END HTML_Path();
END PSEUDOCODE();
```

Παράθεση 5.1 – Αλγόριθμος εύρεσης μοτίβου πλήρους μονοπατιού σε HTML δένδρο

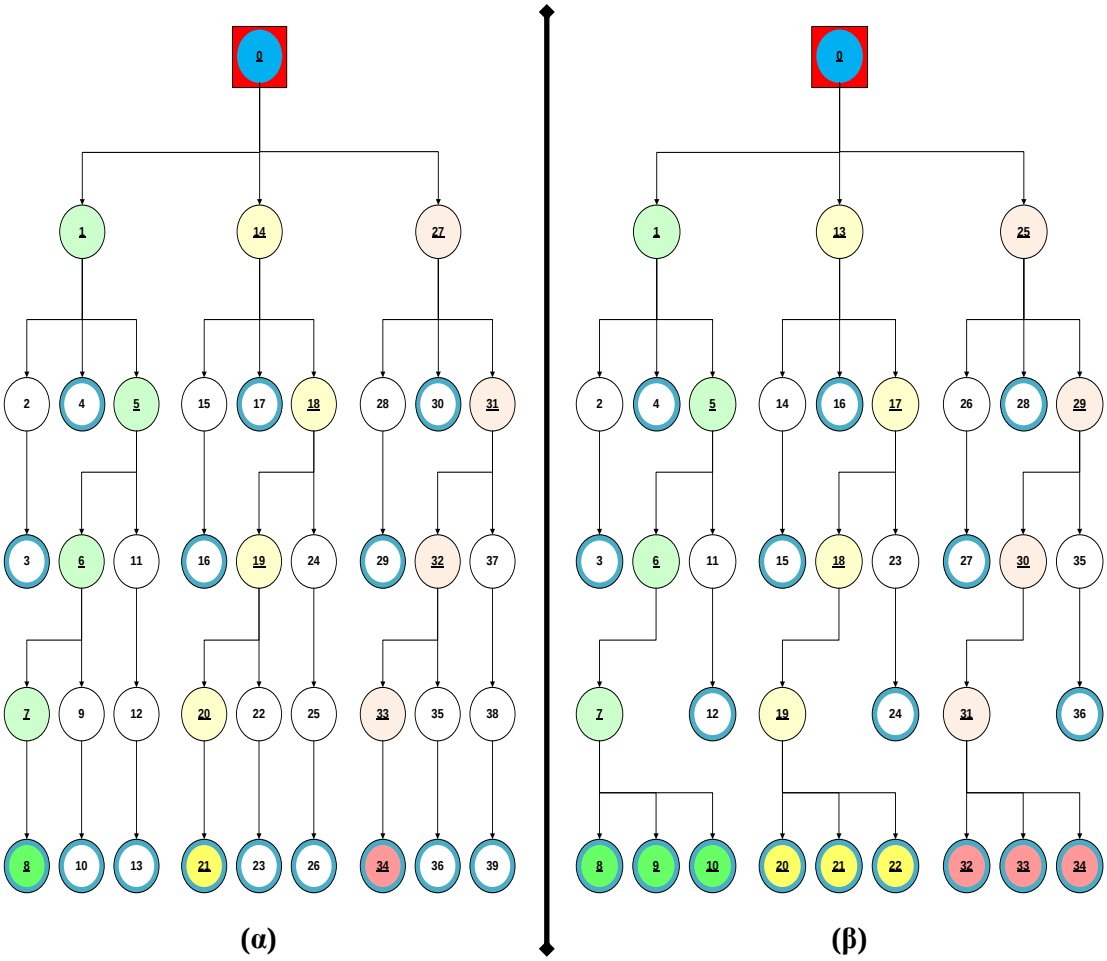
5.1.1.4 *Ειδική περίπτωση με χρήση περιορισμού θέσης του φύλλου στο προσδιορισθέν μοτίβο μονοπατιού*

Υπάρχουν περιπτώσεις όπου ο καθορισμός των τριών διαδρομών επί του HTML δένδρου δεν επαρκεί για τον σαφή προσδιορισμό της θέσης των αντίστοιχων υποστοιχείων των αποτελεσμάτων. Συγκεκριμένα, αναφέρουμε το ακόλουθο παράδειγμα, όπου η μέχρι στιγμής επιλεχθείσα μέθοδος υλοποίησης αδυνατεί να δώσει λύση:

```
<HTML>                                  <HTML>
  <HEAD>                                  <HEAD>
    …                                       …
  </HEAD>                                 </HEAD>
  <BODY>                                  <BODY>
     …                                       …
    <DIV>                                   <UL>
      <H1>title1</H1>                         <LI>title1</LI>
      <P>summary1</P>                         <LI>summary1</LI>
      <CITE>link1</CITE>                      <LI>link1</LI>
    </DIV>                                  </UL>
    <DIV>                                   <UL>
      <H1>title2</H1>                         <LI>title2</LI>
      <P>summary2</P>                         <LI>summary2</LI>
      <CITE>link2</CITE>                      <LI>link2</LI>
    </DIV>                                  </UL>
    <DIV>                                   <UL>
      <H1>title3</H1>                         <LI>title3</LI>
      <P>summary3</P>                         <LI>summary3</LI>
      <CITE>link3</CITE>                      <LI>link3</LI>
    </DIV>                                  </UL>
     …                                       …
  </BODY>                                 </BODY>
</HTML>                                 </HTML>
```

Παράθεση 5.2 – Χαρακτηριστικό τμήμα του HTML κώδικα δύο ιστοσελίδων αποτελεσμάτων, προερχόμενων από διαφορετικές μηχανές αναζήτησης

Εκτελούμε μία αναζήτηση με την ίδια λέξη-κλειδί σε δύο διαφορετικές μηχανές, με το ενδιαφέρον μέρος του HTML κώδικα των αντίστοιχων επιστρεφόμενων ιστοσελίδων αποτελεσμάτων να φαίνεται στην ακριβώς προηγούμενη σελίδα. Έστω ότι θέλουμε να εκμαιεύσουμε τους τίτλους των αποτελεσμάτων και από τις δύο λίστες. Το πρώτο μοτίβο μονοπατιού, θα είναι το εξής: HTML → BODY → … → DIV → H1 και όλα βαίνουν καλώς. Το δεύτερο HTML path pattern, θα είναι το ακόλουθο: HTML → BODY → … → UL → LI, στο οποίο όμως δεν ταιριάζουν μόνο οι τίτλοι των αποτελεσμάτων, αλλά και οι περιλήψεις και τα URLs. Επομένως, εντοπίσαμε ένα πρόβλημα, το οποίο μπορεί να αντιμετωπιστεί μόνο εφόσον εκτός του μονοπατιού, τηρούμε και την σχετική θέση του τελικού κόμβου. Έτσι, εάν στο παράδειγμα αυτό ακολουθούσαμε μεν την υποδεικνυόμενη διαδρομή, αλλά επίσης στο τέλος δεν λαμβάναμε τα περιεχόμενα όλων των κόμβων παρά μόνο του εκάστοτε πρώτου, θα ήμασταν σε θέση να εξαγάγουμε τις ζητούμενες πληροφορίες. Για λόγους καλύτερης κατανόησης, ας καταφύγουμε σε ένα παρόμοιο παράδειγμα, στο οποίο όμως θα εργαστούμε με τις δενδρικές δομές αντί του HTML κώδικα.



Σχήμα 5.2 – Διάσχιση δύο HTML δένδρων και εύρεση των μοτίβων μονοπατιών

Στα παραπάνω δύο σχήματα απεικονίζονται δύο συνήθεις περιπτώσεις HTML δένδρων που αντιστοιχούν σε ιστοσελίδες αποτελεσμάτων από μηχανές διαδικτυακής αναζήτησης. Για

λόγους απλούστευσης και δίχως βλάβη της γενικότητας, έχουν παραληφθεί τα ονόματα των HTML tags που κανονικά οριοθετούν ένα element και έχουν αντικατασταθεί από αριθμούς. Όπως είναι φανερό, έχει εφαρμοστεί η τεχνική αναζήτησης DFS. Σημειώνουμε, ότι με 0 έχει ονοματιστεί ο κόμβος που αντιστοιχεί στην ρίζα (root) του δένδρου, ενώ με διπλούς κύκλους εμφανίζονται στα ανωτέρω γραφήματα οι τελικοί κόμβοι – φύλλα (leafs). Επιπροσθέτως, πρέπει να τονιστεί ότι έχουν υπογραμμιστεί οι κόμβοι που συνθέτουν ένα πλήρες μονοπάτι σε κάθε ένα από τα υπόδενδρα που εν προκειμένω έχουμε θεωρήσει για το παράδειγμά μας. Υποθέτουμε ακόμα ότι οι δύο ιστοσελίδες που αντιστοιχούν στους γράφους αυτούς περιλαμβάνουν τρία αποτελέσματα η κάθε μία, με τίτλο – URL – περίληψη τα υποστοιχεία του εκάστοτε αποτελέσματος. Έστω ακόμα ότι αναζητούμε τον τίτλο των αποτελεσμάτων και γνωρίζουμε μόνο ότι ο κόμβος 8 και των δύο περιπτώσεων συνιστά ένα τέτοιου είδος result subelement.

- Στην πρώτη περίπτωση (σχήμα 5.2.α), διασχίζοντας το δένδρο βρίσκουμε την διαδρομή 0 → 1 → 5 → 6 → 7 → 8. Ομοίως, εφαρμόζοντας το μοτίβο που υποδεικνύει το εν λόγω μονοπάτι, οδηγούμαστε στην εύρεση και των υπολοίπων title paths: 0 → 14 → 18 → 19 → 20 → 21 και 0 → 27 → 31 → 32 → 33 → 34. Επομένως, εδώ η μεθοδολογία μας λειτουργεί απρόσκοπτα και με επιτυχία.
- Ωστόσο στην δεύτερη περίπτωση (σχήμα 5.2.β), είμαστε σε θέση να βρούμε την διαδρομή – πρότυπο 0 → 1 → 5 → 6 → 7 → 8 και βάσει αυτής και τα μονοπάτια 0 → 13 → 17 → 18 → 19 → 20 & 0 → 25 → 29 → 30 → 31 → 32, όμως παράλληλα θεωρούνται ως ορθά και τα paths 0 → 1 → 5 → 6 → 7 → 9 (αντιστοιχεί στο URL του 1^{ου} result), 0 → 1 → 5 → 6 → 7 → 10 (αντιστοιχεί στην description του 1^{ου} result), 0 → 13 → 17 → 18 → 19 → 21 (αντιστοιχεί στο weblink του 2^{ου} αποτελέσματος), 0 → 13 → 17 → 18 → 19 → 22 (αντιστοιχεί στην summary του 2^{ου} αποτελέσματος), 0 → 25 → 29 → 30 → 31 → 33 (αντιστοιχεί στον διαδικτυακό σύνδεσμο του 3^{ου} result) και 0 → 25 → 29 → 30 → 31 → 34 (αντιστοιχεί στην περίληψη του 3^{ου} result).

Έχει πλέον γίνει κατανοητό το πρόβλημα που προκύπτει στην μία από τις δύο πιο συνηθισμένες απαντώμενες περιπτώσεις. Για την αντιμετώπιση του συγκεκριμένου προβλήματος, οδηγηθήκαμε στην ανάπτυξη της ακόλουθης τεχνικής: Αρχικά επιχειρείται η εφαρμογή της πρώτης μεθόδου για το HTML parsing και εφόσον ο χρήστης κρίνει ότι τα αντίστοιχα σημειωμένα αποτελέσματα στην οθόνη του δεν είναι ικανοποιητικά, δίνει την εντολή στο υποσύστημα εγκατάστασης να προβεί σε re-wrapping, κάνοντας χρήση της δεύτερης μεθόδου, η οποία αναλύεται ακολούθως. Σύμφωνα με αυτήν, λαμβάνεται υπόψη ο αποκαλούμενος *περιορισμός θέσης φύλλου* (leaf position constraint), δηλαδή κατά την απόπειρα εύρεσης του path pattern, καταγράφεται επίσης η σχετική θέση του τελικού κόμβου

της διαδρομής ως προς τον κόμβο-γονέα και έτσι το μοτίβο συνοδεύεται από έναν πρόσθετο ακέραιο αριθμό, ενώ κατά την εφαρμογή του πλέον δεν επιλέγονται όλα τα υπάρχοντα φύλλα, αλλά μόνο το φύλλο που εμφανίζεται με την σειρά που υποδεικνύει ο συγκεκριμένος αριθμός. Με τον τρόπο αυτό επιτυγχάνεται η σωστή λειτουργία του συστήματός μας και στις δύο περιπτώσεις τις οποίες αναδείξαμε.

- Τώρα πια στην δεύτερη περίπτωση (σχήμα 5.2.β), πέρα από την πρότυπη διαδρομή 0 → 1 → 5 → 6 → 7 → 8, λαμβάνουμε υπόψη μας και τον περιορισμό θέσης φύλλου, ο οποίος εδώ είναι 1. Έχοντας ως βάση αυτές τις πληροφορίες εντοπίζουμε μόνο τα μονοπάτια 0 → 13 → 17 → 18 → 19 → 20 & 0 → 25 → 29 → 30 → 31 → 32 (τα οποία και στην διαδρομή – οδηγό ταιριάζουν, αλλά και ο τελευταίος κόμβος που περιλαμβάνουν, αποτελεί το πρώτο παιδί του προτελευταίου κόμβου). Πλέον τα paths 0 → 1 → 5 → 6 → 7 → 9, 0 → 1 → 5 → 6 → 7 → 10, 0 → 13 → 17 → 18 → 19 → 21, 0 → 13 → 17 → 18 → 19 → 22, 0 → 25 → 29 → 30 → 31 → 33 και 0 → 25 → 29 → 30 → 31 → 34 υπακούουν στο μοτίβο του μονοπατιού, όμως ο τελευταίος κόμβος τους δεν συνιστά το πρώτο παιδί του κόμβου – γονέα (αλλά είναι είτε το δεύτερο είτε το τρίτο). Έτσι, επιτυγχάνεται η επιδιωκόμενη διάκριση και ορθά απορρίπτονται τα τελευταία τέσσερα μονοπάτια.

5.1.1.5 Διάκριση μεταξύ δομικών HTML στοιχείων και στοιχείων μορφοποίησης

Για την πραγματοποίηση των προαναφερθέντων εργασιών, αλλά και κάποιων άλλων μικρότερης σημασίας (όπως εκείνης που αναδεικνύεται στην αμέσως επόμενη ενότητα), κατέστη επιτακτική η ανάγκη του διαχωρισμού των διαφόρων HTML ετικετών σε δομικές και μη- δομικές / μορφοποίησης. Ως *structural tags* θεωρούμε εκείνες που διαδραματίζουν καθοριστικό ρόλο στην διαμόρφωση της τελικής όψης μίας HTML σελίδας (όπως οι <a>, <h2>, <td> κλπ). Ως *style tags* ονομάζουμε τις υπόλοιπες, οι οποίες σχετίζονται μόνο με την οπτική μορφοποίηση μίας σελίδας και δεν επηρεάζουν την δομή της (χαρακτηριστικά αναφέρουμε τις <i>, , κ.ά.). Έτσι, η δενδρική δομή που κατασκευάζουμε βασιζόμενοι σε ένα HTML έγγραφο αποτελείται μονάχα από δομικά στοιχεία, παραβλέποντας τα υπόλοιπα [GA05]. Αυτό το τέχνασμα οδηγεί και στην χρήση απλούστερων δομών δεδομένων κατά το HTML parsing, όμως ωφέλησε και με άλλους τρόπους την ανάπτυξη της εφαρμογής μας. Αξιοσημείωτη είναι η περίπτωση που το επιλεχθέν από τον χρήστη μέρος κειμένου συμπεριλαμβάνει style tags (π.χ. ένας τίτλος αποτελέσματος, στον οποίο μία λέξη είναι bold). Εάν δεν είχαμε μεριμνήσει για την συζητούμενη διάκριση, η μεθοδολογία με το path pattern που αναπτύξαμε στις παραπάνω ενότητες θα αποτύγχανε, καθώς το επίμαχο κείμενο δεν θα περικλειόταν από ένα μονάχα στοιχείο, αλλά το αριστερό και το δεξιό υποτμήμα του θα βρίσκονταν ως περιεχόμενα διαφορετικού element από ότι το

μεσαίο υποτμήμα. Με την εκμετάλλευση όμως αυτής της τεχνικής που βελτιώνει την συνολική μέθοδο, τέτοιου είδους προβλήματα αποφεύγονται. Τα προηγούμενα γίνονται ακόμα πιο κατανοητά αν αναλογιστούμε το εξής παράδειγμα:

```
<HTML>
  <HEAD>
    …
  </HEAD>
  <BODY>
     …
    <TABLE>
      <TR>
        <TD>This is <B>title</B> number 1</TD>
        <TD>This is link number 1</TD>
      </TR>
      <TR>
        <TD>This is title number 2</TD>
        <TD>This is link number 2</TD>
      </TR>
      <TR>
        <TD>This is <EM>title</EM> number 3</TD>
        <TD>This is link number 3</TD>
      </TR>
       …
    </TABLE>
     …
  </BODY>
</HTML>
```

Παράθεση 5.3 – Χαρακτηριστικός HTML κώδικας ιστοσελίδας αποτελεσμάτων αναζήτησης, με στοιχεία μορφοποίησης εντός της ενδιαφέρουσας πληροφορίας

Έστω ότι εκτελούμε μία αναζήτηση κάνοντας χρήση κάποιας μηχανής και ο HTML κώδικας της επιστρεφόμενης σελίδας αποτελεσμάτων είναι ο παραπάνω. Κάνουμε ακόμα την παραδοχή, ότι ενδιαφερόμαστε να εξαγάγουμε τους τίτλους των αποτελεσμάτων. Το μοτίβο μονοπατιού που θα προκύψει, εφόσον θεωρήσουμε όλα τα HTML στοιχεία ως δομικά είναι

το ακόλουθο: HTML → BODY → … → TABLE → TR → TD → (B / EM), με περιορισμό θέσης φύλλου 1. Ανακύπτει πρόβλημα δηλαδή με την κατάληξη της διαδρομής, καθώς σε ενδιαφέροντα δεδομένα καταλήγουν τα εξής τρία paths (για leaf constraint 1): HTML → BODY → … → TABLE → TR → TD, HTML → BODY → … → TABLE → TR → TD → B, HTML → BODY → … → TABLE → TR → TD → EM. Μπορούμε όμως να υπερβούμε την συζητούμενη δυσκολία, αν επιχειρήσουμε μία «σημασιολογική» προσέγγιση των διαφόρων HTML elements και δούμε, ποια από αυτά προορίζονται να διαδραματίσουν έναν δομικό ρόλο και ποια έναν απλό ρόλο μορφοποίησης. Έτσι εδώ, τα HTML, HEAD, BODY, TABLE, TR, TD υπάγονται στην πρώτη κατηγορία, ενώ τα B και EM στην δεύτερη. Επομένως, καταλήγουμε στο εξής (δομικό) μοτίβο μονοπατιού: HTML → BODY → … → TABLE → TR → TD (με περιορισμό θέσης τελικού κόμβου: 1), ενώ τα άλλα στοιχεία (τα επονομαζόμενα μορφοποίησης) αγνοούνται. Παρατηρούμε, ότι αξιοποιώντας την συγκεκριμένη τεχνική καταφέρνουμε να αντιμετωπίσουμε το πρόβλημα που παρουσιάστηκε.

5.1.1.6 Χρήση κανονικών εκφράσεων στην διαδικασία του HTML parsing

Οι *κανονικές εκφράσεις* (regular expressions, εν συντομία regexes) παρέχουν έναν σαφή και έξυπνο τρόπο για την ταυτοποίηση συμβολοακολουθιών που συνιστούν κομμάτια ενός κειμένου ενδιαφέροντος, όπως χαρακτήρες, λέξεις ή μοτίβα χαρακτήρων [WP9]. Στο σύστημά μας αξιοποιούνται κυρίως για τον ορισμό των δομικών στοιχείων και των στοιχείων μορφοποίησης, καθώς και για την προσθήκη ευελιξίας στις επιλογές κειμένου, στις οποίες προβαίνει ο χρήστης κατά την διαδικασία της μηχανικής εκμάθησης ανάγνωσης αποτελεσμάτων (μέσω της τεχνικής του screen-scraping). Όσον αφορά την πρώτη χρησιμότητά τους, παραθέτουμε ακολούθως τα μοτίβα που οικοδομήσαμε με την βοήθεια regular expressions για τις structural και τις style tags:

```
structural_tag_pattern = "(?s)</?+(?!b(?= |>)|i(?= |>)|u(?= |>)|em
(?= |>)|strong(?= |>)|big(?= |>)|small(?= |>)|style(?= |>)|font
(?= |>)|\\p{Punct}|br(?= |>)){1}.*?>";
```

Παράθεση 5.4 – Μοτίβο δομικών στοιχείων, ορισμένο με χρήση κανονικών εκφράσεων

```
style_tag_pattern = "(?s)</?+(?=b(?= |>)|i(?= |>)|u(?= |>)|em
(?= |>)|strong(?= |>)|big(?= |>)|small(?= |>)|style(?= |>)|font
(?= |>)|\\p{Punct}|br(?= |>)){1}.*?>";
```

Παράθεση 5.5 – Μοτίβο στοιχείων μορφοποίησης, ορισμένο με χρήση κανονικών εκφράσεων

Σχετικά με την δεύτερη χρήση των regexes στην εφαρμογή μας, αυτές προσδίδουν μία ελευθερία (στο μέτρο του δυνατού) στις επιλογές υποστοιχείων αποτελεσμάτων που κάνει ο χρήστης, προκειμένου να υποδείξει στο σύστημα τον τρόπο ανάγνωσης μίας result webpage από την εγκαθιστάμενη search engine. Εάν το υποσύστημα ενσωμάτωσης απαιτούσε την αυστηρή ανάδειξη καθενός από τα τρία subelements ενός result, τότε η εγκατάσταση θα αποτύγχανε εάν ο χρήστης έκανε λάθος ακόμα και έναν χαρακτήρα κατά το mouse-highlighting. Επί παραδείγματι, έστω ότι θέλουμε να «διδάξουμε» στην μηχανή την ανάγνωση των URLs από μία σελίδα αποτελεσμάτων και ας υποθέσουμε ότι ο διαδικτυακός σύνδεσμος που θέλουμε να χρησιμοποιήσουμε είναι ο εξής: *http://en.wikipedia.org/wiki/Alan_Turing*. Για να λειτουργήσει σωστά η μηχανική εκμάθηση, πρέπει να επιλέξουμε ολόκληρο το στοιχείο, δηλαδή: *http://en.wikipedia.org/wiki/Alan_Turing*. Με την χρήση κανονικών εκφράσεων, κάτι τέτοιο δεν είναι απαραίτητο και αρκεί η επιλογή ενός (σημαντικού) τμήματος για την απρόσκοπτη λειτουργία: http://en.wikipedia.org/wiki/*Alan_Turing*. Πρέπει βεβαίως να σημειωθεί, ότι το επιλεχθέν μέρος του εκάστοτε υποστοιχείου οφείλει να είναι επαρκώς μεγάλο για να μην προκύψουν άλλου είδους προβλήματα, τα οποία σχετίζονται με το ταίριασμα του χρησιμοποιούμενου regular expression pattern και σε άλλα ανεπιθύμητα μέρη της ιστοσελίδας.

5.1.2 Ταξινόμηση και συγχώνευση αποτελεσμάτων αναζήτησης στο Internet

Το υποσύστημα διαδικτυακής αναζήτησης που χρησιμοποιείται στο παρόν έργο αξιοποιεί μία πληθώρα μηχανών αναζήτησης (είτε wrapped, είτε API-embedded) και εξαιτίας του γεγονότος αυτού οδηγεί στην υιοθέτηση του χαρακτηρισμού *μετα-αναζήτηση* (meta-search). Ο προσδιορισμός του αριθμού αποτελεσμάτων που θα ζητούνται από την κάθε μία μηχανή αναζήτησης, βάσει προκαθορισμένου συνολικού πλήθους είναι η πρώτη από τις εμφανιζόμενες προκλήσεις.

Μία άλλη είναι η σχετική ταξινόμηση των αποτελεσμάτων μεταξύ τους, προκειμένου το κάθε ένα να λάβει την αντίστοιχη θέση στην τελική λίστα που θα παρουσιαστεί στον χρήστη. Επίσης, πρέπει να αντιμετωπιστεί η περίπτωση επιστροφής του ίδιου αποτελέσματος από μία ή περισσότερες μηχανές αναζήτησης.

Ένα ακόμα θέμα είναι το υπαρκτό ενδεχόμενο δύο διαφορετικά αποτελέσματα να είναι ακριβώς το ίδιο σημαντικά. Τέλος, η διαμόρφωση του οριστικού καταλόγου των αποτελεσμάτων που θα δοθεί στον χρήστη πρέπει να απορρέει μέσω της ομαλής συγχώνευσης των διαφόρων στοιχείων.

Κεντρικό ρόλο στην υλοποίηση των παραπάνω διαδραματίζει ο αλγόριθμος *weighted Borda-Fuse*, ο οποίος σε συνδυασμό με δικές μας επιπρόσθετες λειτουργίες και βελτιώσεις, κατορθώνει να ικανοποιήσει τις απαιτήσεις του συζητούμενου υποσυστήματος πλήρως.

Σημειώνουμε ακόμα, ότι μέριμνα λαμβάνεται και για την παράθεση έξτρα λέξεων-κλειδιών από τον χρήστη, την επιλογή επιθυμητού τύπου δεδομένων (π.χ. general web pages ή papers), τον τρόπο κατάταξης (αναμεμειγμένα ή κατά μηχανή αναζήτησης) και το μέγεθος ανάπτυξης των αποτελεσμάτων (σύντομη ή εκτενής μορφή).

5.1.2.1 Διαμοιρασμός των ζητούμενων αποτελεσμάτων ανά μηχανή αναζήτησης

Όπως έχει ήδη προαναφερθεί, κάθε μηχανή αναζήτησης στο σύστημά μας λαμβάνει μία συγκεκριμένη βαθμολογία, με κλίμακα από 1 έως 5 (1 = άριστα, 5 = κακώς), για τον εκάστοτε τύπο δεδομένων (0 εφόσον η εν λόγω μηχανή δεν υποστηρίζει τον τρέχοντα τύπο). Ο χρήστης είναι εκείνος που προσδιορίζει το συνολικό επιθυμητό πλήθος αποτελεσμάτων, ενώ το υποσύστημα αναζήτησης καλείται βάσει του αριθμού αυτού και των βαθμολογιών των μηχανών να καθορίσει τα επιμέρους πλήθη αποτελεσμάτων, τα οποία θα κληθεί να φέρει η κάθε μία search engine. Οι χρησιμοποιούμενες μηχανές είναι όσες έχουν ενσωματωθεί με την βοήθεια APIs, καθώς και όσες έχουν εγκατασταθεί με εκμετάλλευση της τεχνικής του screen-scraping. Ο απλός μαθηματικός τύπος που αξιοποιείται για την συζητούμενη διαδικασία είναι ο ακόλουθος:

`Res_Num[SE(i)] = floor{[SE_Rank(i)·N] / Sum[SE_Rank(1..M)]}`

όπου `Res_Num[SE(i)]`: αριθμός αποτελεσμάτων που αναλογεί στην τρέχουσα μηχανή αναζήτησης `i`, `N`: συνολικό πλήθος αποτελεσμάτων, `SE_Rank(i)`: ανηγμένη βαθμολογία της τρέχουσας μηχανής αναζήτησης `i` για τον συγκεκριμένο τύπο δεδομένων (είναι το δεκαπλάσιο αντίστροφο της δοθείσας βαθμολογίας αν η τελευταία είναι διάφορη του μηδενός, αλλιώς και αυτή μηδενική), `M`: πλήθος όλων των μηχανών αναζήτησης, `Sum[SE_Rank(1..M)]`: άθροισμα των ανηγμένων βαθμών όλων των μηχανών αναζήτησης, `floor()`: συνάρτηση στρογγυλοποίησης με αποκοπή.

5.1.2.2 Ο αλγόριθμος weighted Borda-Fuse

Γνωρίζοντας το πλήθος αποτελεσμάτων και την βαθμολογία που αντιστοιχεί σε κάθε μηχανή αναζήτησης είναι δυνατή η εφαρμογή του δημοκρατικού αλγόριθμου Borda-Fuse, ο οποίος ταξινομεί τα στοιχεία σε μία ενιαία λίστα. Σύμφωνα με τον συγκεκριμένο αλγόριθμο, κάθε ένα αποτέλεσμα βαθμολογείται κατά φθίνουσα σειρά, όσο η θέση στην οποία βρίσκεται εντός του καταλόγου αποτελεσμάτων της μηχανής προέλευσης είναι χαμηλότερη. Το βήμα βαθμολόγησης μεταξύ δύο διαδοχικών αποτελεσμάτων είναι η μονάδα, ενώ το πρώτο αποτέλεσμα έχει ως βαθμό το πλήθος των αποτελεσμάτων `Res_Num[SE(i)]` που αντιστοιχούν στην εν λόγω μηχανή και το τελευταίο το `1`. Εν συνεχεία κάθε result οποιασδήποτε search engine πολλαπλασιάζεται επί τον βαθμό αξιολόγησης που φέρει η

εκάστοτε μηχανή και τελικώς όλα τα αποτελέσματα διατάσσονται με φθίνοντα τρόπο, κάνοντας απλώς χρήση του συνήθους αλγορίθμου ταξινόμησης BubbleSort (εναλλακτικά θα μπορούσε να χρησιμοποιηθεί QuickSort ή InsertionSort ή MergeSort).

Έχει πλέον επιτευχθεί το ζητούμενο, ενώ αξιοπρόσεκτο είναι το γεγονός ότι ο χαρακτηρισμός «δημοκρατικός», προκύπτει από τον τρόπο που τα αποτελέσματα συλλέγουν τις βαθμολογίες που τους αντιστοιχούν, υπό την μορφή ψήφων, με την όλη διαδικασία να θυμίζει εκλογή των «δημοφιλέστερων» και γενικότερα μία ψηφοφορία.

Σημειώνουμε επιπλέον, ότι σε περίπτωση απόλυτης ισοβαθμίας δύο αποτελεσμάτων, πρώτα τοποθετείται εκείνο του οποίου η μηχανή προέλευσης προηγείται αλφαβητικά (όσον αφορά την ονομασία της). Ακόμα, για το ενδεχόμενο ταύτισης δύο αποτελεσμάτων, έχουμε φροντίσει έτσι ώστε να κρατείται μόνο το ένα εκ των δύο (αυτό που συγκεντρώνει την μεγαλύτερη βαθμολογία), ενώ το δίδυμό του αφαιρείται από την αντίστοιχη sublist, η οποία εμπλουτίζεται με ένα πρόσθετο αποτέλεσμα στο τέλος, που υπό κανονικές συνθήκες δεν θα εισαγόταν στην λίστα. Επίσης, λόγω της αποκοπής στην οποία καταφεύγουμε, το συνολικό πλήθος των αποτελεσμάτων που διαμοιράζονται είναι μικρότερο του επιθυμητού, και έτσι η διαφορά συμπληρώνεται διανέμοντας και τα υπόλοιπα results με «δημοκρατικό» τρόπο.

Τέλος, για λόγους πληρότητας, παραθέτουμε αμέσως παρακάτω ένα παράδειγμα εφαρμογής του αλγορίθμου Borda-Fuse σε ένα συνηθισμένο περιστατικό διαδικτυακής μετα-αναζήτησης:

- Επιθυμητό πλήθος αποτελεσμάτων: N = 20
- Επιλεχθείς τύπος δεδομένων: general web pages
- Διαθέσιμες μηχανές αναζήτησης (με βαθμολογίες): α) Google (1), β) Yahoo (1), γ) MS Live (2), δ) Gigablast (3), ε) Technorati (4 – αν και blog search engine, κάνουμε την παραδοχή ότι επιστρέφει και ιστοσελίδες γενικού περιεχομένου), στ) YouTube (0 – θεωρείται video search engine)
- Πλήθος διαθέσιμων μηχανών αναζήτησης: M = 6
- Ανηγμένες βαθμολογίες των μηχανών αναζήτησης: α) Google: 10 / 1 = 10, β) Yahoo: 10 / 1 = 10, γ) MS Live: 10 / 2 = 5, δ) Gigablast: 10 / 3 ≈ 3.333, ε) Technorati: 10 / 4 = 2.5, στ) YouTube: 0
- Διαμοιρασμός αποτελεσμάτων:

  Αρχικά:

  - Στο Google: $N_1' = 10 \cdot 20 / (10 + 10 + 5 + 3.333 + 2.5 + 0) = 200 / 30.833 \approx$ $\approx [6.487] = 6$
  - Στο Yahoo: $N_2' = 10 \cdot 20 / 30.833 = 200 / 30.833 \approx [6.487] = 6$

- Στο MS Live: $N_3' = 5 \cdot 20 / 30.833 = 100 / 30.833 \approx [3.243] = 3$
- Στο Gigablast: $N_4' = 3.333 \cdot 20 / 30.833 = 66.66 / 30.833 \approx [2.162] = 2$
- Στο Technorati: $N_5' = 2.5 \cdot 20 / 30.833 = 50 / 30.833 \approx [1.621] = 1$
- Στο YouTube: $N_6' = 0 \cdot 20 / 30.833 = 0 / 30.833 = 0$

Τελικά:

- Επειδή: $N' = N_1' + N_2' + N_3' + N_4' + N_5' + N_6' = 6 + 6 + 3 + 2 + 1 + 0 = 18 < 20 = N$, όπου $N - N' = 20 - 18 = 2$, μοιράζουμε και αυτά τα αποτελέσματα:

  Έτσι: $N_1 = N_1' + 1 = 6 + 1 = 7$ και $N_2 = N_2' + 1 = 6 + 1 = 7$, ενώ: $N_3 = N_3' = 3$, $N_4 = N_4' = 2$, $N_5 = N_5' = 1$, $N_6 = N_6' = 0$

- Βαθμολόγηση αποτελεσμάτων ανά μηχανή αναζήτησης:
  - του Google: $G_1 = 7$, $G_2 = 6$, $G_3 = 5$, $G_4 = 4$, $G_5 = 3$, $G_6 = 2$, $G_7 = 1$
  - του Yahoo: $Y_1 = 7$, $Y_2 = 6$, $Y_3 = 5$, $Y_4 = 4$, $Y_5 = 3$, $Y_6 = 2$, $Y_7 = 1$
  - του MS Live: $L_1 = 3$, $L_2 = 2$, $L_3 = 1$
  - του Gigablast: $B_1 = 2$, $B_2 = 1$
  - του Technorati: $T_1 = 1$
  - του YouTube: –
- Βαθμολόγηση αποτελεσμάτων συγκεντρωτικά:
  - του Google (ανηγμένη βαθμολογία = 10): $SG_1 = 10 \cdot 7 = 70$, $SG_2 = 60$, $SG_3 = 50$, $SG_4 = 40$, $SG_5 = 30$, $SG_6 = 20$, $SG_7 = 10$
  - του Yahoo (ανηγμένη βαθμολογία = 10): $SY_1 = 10 \cdot 7 = 70$, $SY_2 = 60$, $SY_3 = 50$, $SY_4 = 40$, $SY_5 = 30$, $SY_6 = 20$, $SY_7 = 10$
  - του MS Live (ανηγμένη βαθμολογία = 5): $SL_1 = 5 \cdot 3 = 15$, $SL_2 = 10$, $SL_3 = 5$
  - του Gigablast (ανηγμένη βαθμολογία = 3.333): $SB_1 = 3.333 \cdot 2 = 6.666 \approx 7$, $SB_2 = 3.333 \approx 3$
  - του Technorati (ανηγμένη βαθμολογία = 2.5): $ST_1 = 2.5 \cdot 1 = 2.5 \approx 3$
  - του YouTube (ανηγμένη βαθμολογία = 0): –
- Ζητούμενη ενιαία τελική λίστα των ταξινομημένων και συγχωνευμένων αποτελεσμάτων:

  $\{ SG_1, SY_1, SG_2, SY_2, SG_3, SY_3, SG_4, SY_4, SG_5, SY_5, SG_6, SY_6, SL_1, SG_7, SL_2, SY_7, SB_1, SL_3, SB_2, ST_1 \}$

Παράδειγμα 5.1 – Κατανομή, διάταξη και συγχώνευση αποτελεσμάτων διαδικτυακής μετα-αναζήτησης

5.1.3 Αυτοματοποιημένη πρόγνωση του πλήρους URL διαδικτυακής αναζήτησης

Η εφαρμογή μας υποστηρίζει την αυτόματη πρόβλεψη του διαδικτυακού συνδέσμου της εκάστοτε εισαγόμενης μηχανής αναζήτησης. Όπως έχει ήδη προαναφερθεί, το επονομαζόμενο *full search URL* (π.χ. *http://search.yahoo.com/search?p=einstein*) αποτελείται από τρία μέρη:

a) το *base URL* (*http://search.yahoo.com*),

b) το *search script* (*/search*) και

c) γ) την *query parameter* (*?p=*).

Σημειώνουμε ότι το base (ή *main*) URL παρέχεται από τον ίδιο τον χρήστη στην πρώτη καρτέλα του οδηγού εγκατάστασης. Εκτός αυτού όμως, τόσο το search script όσο και η query parameter συμπληρώνονται αυτομάτως από το ίδιο το σύστημα. Εκείνο που ουσιαστικά επιτελείται είναι κατ' αρχάς η επίσκεψη της κύριας ιστοσελίδας της προς ενσωμάτωση μηχανής αναζήτησης.

Σε αυτό το σημείο το υποσύστημα εγκατάστασης λαμβάνει τον αντίστοιχο HTML κώδικα και επιχειρεί να εκμαιεύσει από αυτόν τις ενδιαφέρουσες πληροφορίες.

1. Η όλη ιδέα βασίζεται στο γεγονός ότι κατά κανόνα η μορφή των εν λόγω webpages είναι ιδιαίτερα απλή. Έτσι, συνήθως περιλαμβάνουν ένα πεδίο προς εισαγωγή κειμένου (textfield για τα keywords), ένα κομβίο αναζήτησης (το λεγόμενο search button) και ίσως στοιχεία που υποστηρίζουν κάποιες πρόσθετες λειτουργικότητες.
2. Κατέχοντας γνώσεις σύνταξης HTML εγγράφων, κανείς μπορεί από τα παραπάνω να εξάγει το συμπέρασμα ότι το ζητούμενο search script βρίσκεται πιθανότατα στην τιμή του field "action", το οποίο συνοδεύει στοιχεία τύπου "form", ενώ η ζητούμενη query parameter μπορεί ενδεχομένως να εντοπιστεί στην τιμή του field "name" σε στοιχείο τύπου "input".
3. Για διασφάλιση ακόμα υψηλότερης πιθανότητας επιτυχίας, στην πρώτη περίπτωση μεριμνούμε έτσι ώστε να αγνοούνται τα form elements με null action field (δηλαδή εκείνα που δεν εμφανίζουν field με ονομασία action), και στην δεύτερη περίπτωση φροντίζουμε την συμπερίληψη στο σύνολο-στόχο μόνο των input elements που είτε παρουσιάζουν null type field είτε εμφανίζουν ως value του type field την συμβολοακολουθία "text".

Οι προηγούμενες επιλογές και περιορισμοί επιβάλλονται από την σύνταξη του HTML προτύπου και μπορούν να γίνουν περισσότερο κατανοητοί, λαμβάνοντας υπόψη τα κάτωθι. Εκεί δίνουμε αρχικά το γενικό περίγραμμα κατασκευής μίας submit form και στην συνέχεια παραθέτουμε το αντίστοιχο παράδειγμα που σχετίζεται με την διάσημη μηχανή αναζήτησης του Yahoo.

```
<form name="input" action="html_form_submit.asp" method="get">
Type here:
        <input type="text" name="inserted_data">
        <input type="submit" value="Submit">
</form>
```

Παράθεση 5.6 – Στοιχειώδης HTML κώδικας κατασκευής ενός textfield & ενός submit button, με χρήση form element

```
<form id="sf" action="/search" accept-charset="utf-8">
        <div>
                <label for="yschsp" class="off-left">Web Search</label>
                <input type="text" id="yschsp" name="p" value="">
                <input type="submit" class="ygbt" value="Search">
                <input type="hidden" name="fr" value="sfp" />
                <input type="hidden" name="fr2" value="" />
                <input type="hidden" name="iscqry" id="iscqry">
                        <div id="atg" class="panel hidden">
                                <ul id="atgl"></ul>
                        </div>
        </div>
</form>
```

Παράδειγμα 5.2 – Τμήμα του HTML source με το αντίστοιχο form element της Yahoo Web Search Engine, όπου έχουν τονιστεί το search script και η query parameter

Μετά την εφαρμογή της περιγραφείσας διαδικασίας επαναληπτικά, καταλήγουμε σε δύο λίστες, η πρώτη εκ των οποίων περιλαμβάνει όλα τα ευρεθέντα πιθανά search scripts, ενώ η δεύτερη περιέχει όλες τις ευρεθείσες υποψήφιες query parameters. Αξιοσημείωτο είναι το γεγονός ότι έχουμε μεριμνήσει έτσι ώστε να μην καταχωρούνται πολλαπλά οι ίδιες τιμές. Επιπλέον, πρέπει να τονίσουμε πως το σύστημά μας υποδεικνύει δύο στοιχεία (ένα για το search script και ένα για την query parameter) ως τα συνιστώμενα (recommended), εφόσον αυτά ταυτίζονται με τα συνήθως απαντώμενα (όπως τα "/search", "/index.php", "/web" για scripts και οι "q", "p", "query" για parameters). Επίσης, οφείλουμε να αναφέρουμε ότι σε περίπτωση που το σύστημά μας αδυνατεί να βρει τις ζητούμενες σωστές τιμές ή ο ίδιος ο χρήστης θέλει να εισάγει τις δικές του, παρέχεται η δυνατότητα της πληκτρολόγησής τους (custom values). Βέβαια, κάτι τέτοιο απαιτεί πιο προηγμένες γνώσεις, που ξεφεύγουν από το πλαίσιο της απλής χρήσης, δίχως όμως να αποτελούν το οτιδήποτε ιδιαιτέρως εξειδικευμένο. Τέλος, άξιο λόγου είναι το γεγονός ότι η συγκεκριμένη μεθοδολογία που αναδείχθηκε στην παρούσα ενότητα λειτουργεί απροβλημάτιστα στην συντριπτική πλειοψηφία των κοινών μηχανών αναζήτησης.

5.1.4 Συγκέντρωση των δεδομένων μίας εγκαθιστάμενης μηχανής αναζήτησης

Προκειμένου μία μηχανή αναζήτησης να καταστεί πλήρως λειτουργική, απαιτείται ένα σύνολο ουσιαστικών πληροφοριών που την ορίζουν. Αυτή η συλλογή δεδομένων αναπαρίσταται στην εφαρμογή μας από μία ομάδα συμβολοακολουθιών, οι οποίες κατά την εγκατάσταση καταγράφονται σε ένα XML αρχείο, ενώ διαβάζονται από το ίδιο αρχείο κατά την χρήση της συγκεκριμένης μηχανής. Τονίζουμε στο σημείο αυτό ότι είναι εντελώς διαφορετικός ο τρόπος διαχείρισης στο σύστημά μας μίας API-embedded search engine και μίας wrapped search engine. Εδώ γίνεται λόγος για τις μηχανές αναζήτησης που εγκαθιστά ο ίδιος ο χρήστης με την βοήθεια του σχετικού installation wizard και της τεχνικής του screen scraping, δηλαδή τις wrapped search engines.

1. Η ονομασία (name – π.χ. Google Search), το main URL (όπως *http://www.live.com/*), ο χρησιμοποιούμενος τύπος δεδομένων της καινούριας μηχανής (data type – επί παραδείγματι general web pages ή paper κλπ) και ο βαθμός αξιολόγησής της (evaluation integer – σε κλίμακα από 1 έως 5) εισάγονται από τον ίδιο τον χρήστη.
2. Επιπλέον, μπορούν να πληκτρολογηθούν οι πρόσθετες λειτουργικότητες (advanced search features) που υποστηρίζει ενδεχομένως η εν λόγω μηχανή. Αυτό το στάδιο είναι προαιρετικό και αφορά τέσσερα στοιχεία προηγμένης αναζήτησης, ανάλογα με τον επιλεχθέντα τύπο δεδομένων (π.χ. date- / domain- / file type- / region-based search για web pages). Πρέπει να αναφερθεί ότι σε περίπτωση που επιλέξουμε να καταστήσουμε τη νέα μηχανή λειτουργική ως προς ένα τέτοιου είδους στοιχείο, οφείλουμε να γνωστοποιήσουμε στο σύστημα τα απαραίτητα συνοδευτικά δεδομένα, δηλαδή την σχετική script expansion, το αντίστοιχο filter και την κατάλληλη query expansion.
3. Στην συνέχεια αναλαμβάνει η ίδια η εφαρμογή να «μαντέψει» τόσο το search script για την συγκεκριμένη μηχανή αναζήτησης, όσο και την query parameter, προκειμένου να οικοδομήσει το search URL (αφού ήδη γνωρίζει το base URL).
4. Τέλος, σε μία διαδραστική διαδικασία που βασίζεται στην μεθοδολογία της απόξεσης δεδομένων από την οθόνη (screen-scraping) και υποβοηθάται από την χρήση κανονικών εκφράσεων (regular expressions), ο χρήστης καλείται να υποδείξει τα τρία συστατικά στοιχεία ενός αποτελέσματος (το οποίο βρίσκεται στην ιστοσελίδα αποτελεσμάτων που εμφανίζεται στον χρήστη, αφού το ίδιο το υποσύστημα εγκατάστασης έχει πραγματοποιήσει μία αναζήτηση με την εγκαθιστάμενη μηχανή βάσει προκαθορισμένης λέξης-κλειδιού) και μέσω τυχόν επιβολής πρόσθετων περιορισμών (όπως ο leaf position constraint για τίτλους ή / και URLs ή / και περιγραφές) να βοηθήσει στην εκμάθηση ανάγνωσης σελίδων αποτελεσμάτων

(κατασκευή HTML node routes για titles – links – summaries) από τη νεοεισαγόμενη search engine.

Αμέσως παρακάτω, παραθέτουμε την βασική δομή του συζητούμενου XML αρχείου, προσδιορίζοντας κάθε φορά τον τρόπο με τον οποίο εισάγονται τα δεδομένα (σημείωση: με κίτρινο έχουν μαρκαριστεί τα εισαγόμενα από τον χρήστη / προβλεπόμενα από το σύστημα στοιχεία που ορίζουν την μηχανή αναζήτησης και την διαφοροποιούν από τις υπόλοιπες, ενώ με γαλάζιο έχουν μαρκαριστεί τα προαιρετικά στοιχεία).

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="Validator.xslt"?>
<search_engine>
   <identification name="search_engine_name" url="main_search_engine_URL">
   </identification>
   <service>
      <data_type rank="evaluation_integer">
         <advanced_feature_1_based_search_support script_expansion="date1" filter=
         "date2" query_expansion="date3">
         </advanced_feature_1_based_search_support>
         <advanced_feature_2_based_search_support script_expansion="domain1" filter=
         "domain2" query_expansion="domain3">
         </advanced_feature_2_based_search_support>
         <advanced_feature_3_based_search_support script_expansion="file1" filter=
         "file2" query_expansion="file3">
         </advanced_feature_3_based_search_support>
         <advanced_feature_4_based_search_support script_expansion="region1" filter=
         "region2" query_expansion="region3">
         </advanced_feature_4_based_search_support>
      </data_type>
   </service>
   <implementation>
      <search script="search script" parameter="query parameter" url="search_URL">
      </search>
      <title_extraction path="HTML_node_route_for_titles" position=
      "leaf_position_constraint_for titles">
      </title_extraction>
      <link_extraction path="HTML_node_route_for_links" position=
      "leaf_position_constraint_for links">
      </link_extraction>
      <summary_extraction path="leaf_position_constraint_for_summaries" position=
      "leaf_position_constraint_for_summaries">
      </summary_extraction>
```

```
    </implementation>
</search_engine>
```

Παράθεση 5.7 – Installed (via wrapping) search engine XML template

Αφού η εγκατάσταση ολοκληρωθεί επιτυχώς, η καινούρια μηχανή αναζήτησης είναι έτοιμη προς χρήση. Το υποσύστημα διαδικτυακής αναζήτησης πλέον αναλαμβάνει την διαχείρισή της οποτεδήποτε υφίσταται σχετικό αίτημα από την πλευρά του χρήστη. Έτσι, όταν επιχειρείται μία αναζήτηση βάσει κάποιου keyword, ερωτώνται τόσο οι API-embedded search engines, όσο και οι wrapped search engines. Στην περίπτωση μίας μηχανής αναζήτησης από το δεύτερο σύνολο, προσπελαύνεται το αντίστοιχο XML αρχείο και απλώς στέλνοντας υπό την μορφή HTTP request το search URL με την λέξη κλειδί, επιστρέφεται από τον σχετικό server η κατάλληλη λίστα αποτελεσμάτων.

Προφανώς, για τον προσδιορισμό του επιθυμητού πλήθους των αποτελεσμάτων αξιοποιείται ο evaluation integer και λαμβάνεται υπ' όψιν ο χρησιμοποιούμενος data type. Επίσης, αν η εν λόγω μηχανή υποστηρίζει στοιχεία προηγμένης λειτουργικότητας και εφόσον κάτι τέτοιο υπάρχει στις απαιτήσεις της επιχειρούμενης αναζήτησης, εκμεταλλευόμαστε το παρεχόμενο script expansion (παραθέτοντάς το δίπλα στο search URL με την αντίστοιχη τιμή για την παράμετρο που περιέχει) ή το υπάρχον φίλτρο (επισυνάπτοντάς το μαζί με το keyword) ή την διαθέσιμη query expansion (που αφορά ουσιαστικά πρόσθετες κατατοπιστικές λέξεις-κλειδιά). Αξιοσημείωτο είναι το γεγονός ότι οι τρεις διαφορετικοί τρόποι υλοποίησης ενός advanced search feature προτιμώνται σύμφωνα με την σειρά με την οποία δόθηκαν αμέσως προηγουμένως.

Τέλος, για την ανάγνωση των επιστρεφόμενων αποτελεσμάτων από την εφαρμογή μας, παίρνουμε τον αντίστοιχο HTML κώδικα της σελίδας αποτελεσμάτων και οικοδομώντας την σχετική δενδρική δομή, εφαρμόζουμε διαδοχικά τα τρία μοτίβα μονοπατιών (HTML node routes – ένα για τους τίτλους, ένα για τα URLs και ένα για τις περιλήψεις των αποτελεσμάτων) πάνω σε αυτήν. Αναφέρουμε συμπληρωματικά, ότι και ο περιορισμός θέσης φύλλου (leaf position constraint) διαδραματίζει καθοριστικό ρόλο στον τελικό προσδιορισμό των συζητούμενων διαδρομών, εφόσον αυτός παρέχεται (για ένα ή δύο ή και τα τρία μονοπάτια), διαφορετικά συμπεραίνουμε πως θα έχει κριθεί (κατά την εγκατάσταση) ως μη απαραίτητος.

5.1.5 Επισήμανση HTML κειμένου στην διαδικασία εκμάθησης απόξεσης ιστο-σελίδων αποτελεσμάτων

Κατά το τελευταίο στάδιο της διαδικασίας εγκατάστασης μίας νέας μηχανής αναζήτησης, ο χρήστης καλείται σε τρεις διαδοχικές φάσεις να υποδείξει στο σύστημα τα υποστοιχεία

(τίτλος – URL – περίληψη) που απαρτίζουν ένα αποτέλεσμα. Η εν λόγω υπόδειξη γίνεται μέσω της απλής τεχνικής του mouse-highlighting, δηλαδή ο χρήστης απλώς κάνοντας click με το ποντίκι του και κρατώντας πατημένο το αριστερό πλήκτρο επιλέγει το κείμενο που θεωρεί πως αντιστοιχεί σε ένα result subelement κάθε φορά. Ο HTML κώδικας του επιλεχθέντος κειμένου λαμβάνεται με την βοήθεια του κάτωθι τμήματος προγράμματος (χρήση JavaScript inline σε Java):

```
String javascript = "var selectedString = document.getSelection();
                     selectedString;" ;
browser.executeScript(javascript);
String selected_text = browser.executeScriptWithReturn(javascript);
```

Επιπλέον, για το μαρκάρισμα των θεωρούμενων από το σύστημα λοιπών (ίδιου τύπου) υποστοιχείων των αποτελεσμάτων, καταφεύγουμε στην προσθήκη CSS tags στον ήδη υπάρχοντα HTML κώδικα (προφανώς μόνο στα κατάλληλα σημεία – Σημείωση: CSS = Cascading Style Sheets):

```
if (current_element.getName().equals("span"))
        HTML_source = HTML_source.replace(current_start_tag + result_subelement +
        current_end_tag,current_start_tag.substring(0,current_start_tag.length()-1) +
        " style=\"background-color:subelement_color\">" + result_subelement +
        current_end_tag);
else
        HTML_source = HTML_source.replace(current_start_tag + result_subelement +
        current_end_tag, current_start_tag + "<span style=\"background-color:
        subelement_color\">" + result_subelement + "</span>" + current_end_tag);
```

Αναφέρουμε ακόμα ότι αν `result_subelement ≡ result_title`, τότε `subelement_color ≡ tomato` (hexadecimal color code: #FF6347), ενώ αν `result_subelement ≡ result_link`, τότε `subelement_color ≡ gold` (hexadecimal color code: #FFD700), και αν `result_subelement ≡ result_summary`, τότε `subelement_color ≡ lightgreen` (hexadecimal color code: #90EE90).

5.1.6 Αναζήτηση πληροφοριών στην Βάση Δεδομένων του DBLP

Έχοντας οργανώσει αρχικά την Βάση Δεδομένων (η οποία συνιστά κλώνο εκείνης που παρέχει το *DBLP*) και έχοντας προβεί στην συνέχεια σε ευρετηριοποίησή της (με την βοήθεια του εργαλείου *Apache Lucene*), είμαστε πλέον σε θέση να υποβάλλουμε ερωτήματα προς αυτήν και να λαμβάνουμε τις αντίστοιχες απαντήσεις. Εξαιτίας του υλοποιημένου indexing, η όλη διαδικασία αναζήτησης διαρκεί κάθε φορά μερικές δεκάδες milliseconds.

Μπορούμε να εκτελούμε query requests προς την ΒΔ, δηλώνοντας προαιρετικά το σχετικό εισαγόμενο field (σημ.: ως προεπιλογή θεωρείται ο τίτλος μίας επιστημονικής δημοσίευσης), ενώ ως basic query reply παίρνουμε ένα από τα ακόλουθα τρία strings:

1) “No matches were found!”,
2) “Matches found! No perfect match was found!”,
3) “Matches found! Perfect match also found!”.

Στην τρίτη περίπτωση, επιστρέφεται και το λεγόμενο extended query reply, το οποίο περιλαμβάνει τα 18 πιθανά πεδία που απαρτίζουν ένα paper μαζί με τις αντίστοιχες τιμές τους (σημ.: ειδική τιμή "empty" σε κάθε πεδίο που δεν υφίσταται για την ευρεθείσα εγγραφή).

Πρέπει επίσης να αναφερθεί, ότι και για την διαδικασία του searching αξιοποιείται το Lucene [HG04]. Πιο συγκεκριμένα, ένας *IndexReader* καλείται να ανοίξει το υπάρχον ευρετήριο, ενώ ένας *IndexSearcher* επιφορτίζεται με το έργο της αναζήτησης, χρησιμοποιώντας τον προηγούμενο reader. Επιπλέον, χρειάζεται ένας *Analyzer* για να επιτελέσει την διαμέριση (tokenization) της συμβολοακολουθίας, στην οποία θα βασιστεί η αναζήτηση, καθώς και την διαχείριση ειδικών χαρακτήρων και την ερμηνεία τελεστών. Επιπροσθέτως, απαιτείται ένας *QueryParser*, που αναλαμβάνει, βάσει του οριζόμενου field και του προσδιορισθέντος analyzer, την «μεταγλώττιση» της ερώτησης προς την ΒΔ.

Επισημαίνουμε ακόμα, ότι ως αποτέλεσμα της αναζήτησης επιστρέφεται μία λίστα από εγγραφές, που μπορούν να χαρακτηριστούν ως επιτυχείς ευρέσεις (*hits*).

1. Εάν το μέγεθος αυτής της λίστας είναι μηδενικό (δηλαδή *hit counter* = 0), βρισκόμαστε στην πρώτη περίπτωση, όπου δεν βρέθηκε καμία επιστημονική δημοσίευση που να μοιάζει (έστω και σχετικά λίγο) με την δοθείσα ως προς το επιλεγμένο πεδίο.
2. Αντιθέτως, για μη μηδενικό μέγεθος της εν λόγω λίστας, συμπεραίνουμε πως υπάρχουν ταιριάσματα μεταξύ του search string και της τιμής του (default / user-defined) field κάποιων papers.
3. Εάν μάλιστα βρεθεί και το επονομαζόμενο «τέλειο ταίριασμα» (*perfect match*), τότε κατανοούμε πως το αρχικό αποτέλεσμα (που αποτέλεσε και την βάση της αναζήτησης) απαντάται και στην βάση δεδομένων μας, επομένως μπορούμε να ανακτήσουμε από αυτήν όλες τις πρόσθετες πληροφορίες των τιμών των πεδίων που είναι διαθέσιμα. Η ύπαρξη τέλειου ταιριάσματος αντιστοιχεί στην προαναφερθείσα τρίτη περίπτωση, ενώ η απουσία του στην δεύτερη.
4. Εκτός αυτών, άξιο προσοχής είναι το γεγονός ότι το Lucene ορίζει εσωτερικά τον δείκτη ομοιότητας (*similarity index*) για κάθε ένα hit. Αυτός λαμβάνει τιμές από 0.01 (ελάχιστη εμφανιζόμενη ομοιότητα) έως 1.00 (απόλυτη ομοιότητα / ταύτιση). Η hit list, για την οποία έγινε λόγος προηγουμένως, περιλαμβάνει τα διάφορα hits ταξινομημένα κατά φθίνουσα similarity. Οπότε συνεπάγεται, ότι ένα perfect match, εφόσον υπάρχει, θα βρίσκεται στην πρώτη θέση της συγκεκριμένης λίστας.

Τέλος, σημειώνουμε πως εμείς θεωρούμε ότι ένα αποτέλεσμα διαδικτυακής αναζήτησης βρίσκεται και στην ΒΔ του DBLP, αν και μόνο αν παρουσιάζεται τέλειο ταίριασμα μεταξύ των δύο, συγκρίνοντάς τα ως προς τον τίτλο τους. Ακολούθως παραθέτουμε το πιο σημαντικό τμήμα κώδικα, που σχετίζεται με την υλοποίηση των όσων ελέχθησαν παραπάνω:

```
public class XML_full_search {

   public String query_full_reply(String question) throws Exception {

        Date start = new Date();
        String full_reply = "empty";
        String index = "C:\\DBLP\\full index";
        String field = "title";

        IndexReader reader = IndexReader.open(index);

        Searcher searcher = new IndexSearcher(reader);
        Analyzer analyzer = new StandardAnalyzer();
        QueryParser parser = new QueryParser(field, analyzer);

        Query query = parser.parse(question);

        String[] query_tokens = query.toString(field).split(" ");

        System.out.println("Searching for: " + query.toString(field));

        System.out.println();

        Hits hits = searcher.search(query);

        int hitCount = hits.length();

        if (hitCount == 0) {

                System.out.println("No matches were found!");

        }

        else {

                System.out.println("Matches found!");
                Document best_doc = hits.doc(0);
                boolean exact_matching = true;
                for (int i = 0; i < query_tokens.length; i++){
                        if (!best_doc.get("title").toLowerCase()
                            .contains(query_tokens[i])) {
                                exact_matching = false;
                                System.out.println("No perfect match was found!");
                                break;
                        }
                }
                if (exact_matching == true) {
                        System.out.println("\nPerfect match also found!");
                        full_reply = best_doc.get("type") + "splitchar" +
                                     best_doc.get("writers") + "splitchar" +
                                     best_doc.get("title") + "splitchar" +
                                     best_doc.get("pages") + "splitchar" +
                                     best_doc.get("year") + "splitchar" +
                                     best_doc.get("volume") + "splitchar" +
                                     best_doc.get("journal") + "splitchar" +
                                     best_doc.get("number") + "splitchar" +
                                     best_doc.get("link") + "splitchar" +
                                     best_doc.get("url") + "splitchar" +
                                     best_doc.get("date") + "splitchar" +
                                     best_doc.get("book") + "splitchar" +
                                     best_doc.get("isbn") + "splitchar" +
                                     best_doc.get("series") + "splitchar" +
                                     best_doc.get("series_url") + "splitchar" +
                                     best_doc.get("publisher") + "splitchar" +
                                     best_doc.get("publisher_url") + "splitchar" +
                                     best_doc.get("school");
                }

        }

        searcher.close();
        reader.close();

        System.out.println();

        Date end = new Date();
        long execution_time = end.getTime() - start.getTime();
```

```
        Time_Teller time_teller = new Time_Teller();
        String literal_time = time_teller.msecs_to_time(execution_time);
        System.out.println("Total searching time: " + literal_time);

        return full_reply;

    }

}
```

Παράθεση 5.8 – Σημαντικότερο τμήμα της κλάσης υλοποίησης της αναζήτησης στην ΒΔ του DBLP με την βοήθεια του Lucene

```
XML_full_search xml_full_search = new XML_full_search();

String full_reply = xml_full_search.query_full_reply("Transient Region Coverage in the
Propulsion IVHM Technology Experiment.");

String[] replies = full_reply.split("splitchar");
```

Παράδειγμα 5.3 – Αναζήτηση στην ΒΔ του DBLP βάσει ενός paper title

5.2 Πλατφόρμες και προγραμματιστικά εργαλεία

Στην παρούσα ενότητα αναφέρονται τα χαρακτηριστικά της συγκεκριμένης υλοποίησης (πλατφόρμα ανάπτυξης και εκτέλεσης της εφαρμογής, αξιοποιηθέντα προγραμματιστικά εργαλεία, συνιστώμενες απαιτήσεις σε hardware, κ.ά.), ενώ επίσης περιγράφεται λεπτομερώς η διαδικασία εγκατάστασης του συστήματος που πραγματεύεται η διπλωματική εργασία σε έναν υπολογιστή (απαραίτητο λογισμικό, αναγκαίες ρυθμίσεις, κλπ).

5.2.1 Τεχνικά χαρακτηριστικά της υλοποίησης του συστήματος

Ο κώδικας της εφαρμογής που μελετάται στην παρούσα διπλωματική εργασία γράφτηκε στην γλώσσα **Java**, η οποία υπάγεται στην αντικειμενοστρεφή φιλοσοφία προγραμματισμού (object-oriented programming language) [Dar01], [Fla05]. Ως πλατφόρμα για την ανάπτυξη (development) και την εκτέλεση (execution) της εν λόγω εφαρμογής χρησιμοποιήθηκε το Eclipse SDK version 3.2.2, γνωστότερο με την κωδική του ονομασία Eclipse Callisto (σημ.: SDK = Software Development Kit) [Hol04], [GBM03]. Αξιομνημόνευτο είναι το γεγονός, ότι το Eclipse μαζί με τα NetBeans συνιστούν τα δύο δημοφιλέστερα IDEs σε παγκόσμια κλίμακα (σημ.: IDE = Integrated Development Environment) για την δημιουργία προγραμμάτων / συστημάτων σε Java.

Άλλες γλώσσες των οποίων κάναμε χρήση (είτε επρόκειτο για programming, είτε για scripting languages) είναι οι κάτωθι: 1) HTML (HyperText Markup Language) [RF05], [W3SCH1], 2) CSS (Cascading Style Sheets) [W3SCH2], 3) XML (eXtensible Markup

Language) [W3SCH3], 4) DTD (Document Type Definition) [W3SCH4], και 5) JavaScript [W3SCH5].

Όσον αφορά τα προγραμματιστικά εργαλεία που αξιοποιήσαμε, αυτά συνοψίζονται στα εξής:

1) Eclipse Visual Editor version 1.2.1,
2) Jericho HTML Parser version 2.6,
3) Apache Lucene version 2.4.0,
4) JadeLiquid Software WebRenderer version 4.2 – Swing edition.

- Το πρώτο από αυτά (Visual Editor) αποτελεί ένα plug-in του Eclipse, που βασίζεται στο AWT (Abstract Windowing Toolkit) και το Swing, και λειτουργεί στα πλαίσια των GEF (Graphical Editing Framework) και EMF (Eclipse Modeling Framework), με απώτερο σκοπό την ευκολότερη και ταχύτερη κατασκευή γραφικών παραθύρων (όπως frames, dialogues, pop-up windows, information / warning / error messages κλπ).
- Ο Jericho HTML Parser είναι ουσιαστικά μία open-source Java βιβλιοθήκη, η οποία επιτρέπει την ανάλυση και την διαχείριση τμημάτων ενός HTML εγγράφου (συμπεριλαμβανομένων των server-side tags), ενώ επίσης συντελεί στην αναπαραγωγή τυχόν μη-αναγνωρίσιμων ή μη-έγκυρων κομματιών του HTML κώδικα. Επίσης, παρέχει συναρτήσεις επεξεργασίας HTML αρχείων σε high-level (ως προς την «απόσταση» μεταξύ προγραμματιστή και μηχανής).
- Το Apache Lucene συνιστά μία Java library, που υλοποιεί μία υψηλής απόδοσης (παρέχοντας υπηρεσίες ευρετηριοποίησης) και πλήρη (όσον αφορά τις δυνατότητες) μηχανή αναζήτησης κειμένου.
- Ο WebRenderer είναι ένα μοναδικό στοιχείο φυλλομετρητή ιστοσελίδων (browser component – γραφτεί σε Java), το οποίο επιτρέπει την κατάλληλη μεταγλώττιση και αναπαράσταση διαδικτυακού περιεχομένου (υποστηρίζοντας απρόσκοπτα όλες τις σχετικές σύγχρονες τεχνολογίες και πρότυπα, που απαντώνται στο Internet).

Επίσης, ως εργαλεία προβολής και επεξεργασίας του εκάστοτε κώδικα, πέρα από τον Java Editor του Eclipse, ή ως χρήσιμες προσθήκες χρησιμοποιήθηκαν και τα ακόλουθα:

1) Notepad++,
2) Altova XML Spy,
3) Altova Semantic Works,
4) Mozilla Firefox Web Developer Add-On,
5) Mozilla Firefox Piggy Bank Extension (version 3.1),
6) Mozilla Firefox Solvent Extension (version 2.0).

Επιπλέον, ο υπολογιστής στον οποίο αναπτύχθηκε και εκτελείται η εφαρμογή μας είναι ο HP Pavilion dv7-1070ev (φορητός), ενώ ως προς τα τεχνικά του χαρακτηριστικά διαθέτει επεξεργαστή Intel Core 2 Duo T9400 2.53GHz και μνήμη RAM 4096MB DDR2.

Τελικά, παραθέτουμε τις σχετικές ιστοσελίδες για τις προαναφερθείσες γλώσσες & εργαλεία:

- Java: *http://java.sun.com/*
- Eclipse: *http://www.eclipse.org/*
- HTML: *http://www.w3.org/TR/html401/*
- CSS: *http://www.w3.org/Style/CSS/*
- XML: *http://www.w3.org/TR/REC-xml/*
- DTD: *http://www.w3.org/TR/REC-xml/#dt-doctype*
- JavaScript: *http://javascript.internet.com/*
- Jericho HTML Parser: *http://jerichohtml.sourceforge.net/doc/index.html*
- Apache Lucene: *http://lucene.apache.org/java/docs/index.html*
- WebRenderer: *http://www.webrenderer.com/products/swing/product/*
- Notepad++: *http://notepad-plus.sourceforge.net/uk/site.htm*
- Altova XML Spy: *http://www.altova.com/products/xmlspy/xml_editor.html*
- Altova Semantic Works: *http://www.altova.com/products/semanticworks/semantic_web_rdf_owl_editor.html*
- Web Developer Firefox add-on: *https://addons.mozilla.org/en-US/firefox/addon/60*
- Piggy Bank Firefox extension: *http://simile.mit.edu/wiki/Piggy_Bank*
- Solvent Firefox extension: *http://simile.mit.edu/wiki/Solvent*

5.2.2 Διαδικασία εγκατάστασης της εφαρμογής

Για την απρόσκοπτη εγκατάσταση και την εύρωστη λειτουργία της εν λόγω εφαρμογής, υφίστανται ορισμένα προαπαιτούμενα σχετικά με την ύπαρξη του κατάλληλου λογισμικού στον υπολογιστή μας. Πρώτα από όλα απαιτείται η παρουσία της Java στο σύστημα, δηλαδή ενός JRE (Java Runtime Environment). Η προτιμώμενη έκδοση συνίσταται να είναι η πιο πρόσφατη και σε κάθε περίπτωση όχι προηγούμενη της version 6-update 12, μπορεί δε να βρεθεί στην διαδικτυακή τοποθεσία: *http://java.sun.com/javase/downloads/index.jsp*. Ακόμα, χρειάζεται το πρόγραμμα αποσυμπίεσης αρχείων WinRAR, το οποίο κανείς μπορεί να κατεβάσει από τον ιστότοπο: *http://www.win-rar.com/download.html*. Επιπροσθέτως, χρήσιμη είναι η ύπαρξη ενός XML editor (όπως ο Oxygen XML Editor, ο οποίος βρίσκεται στο URL: *http://www.oxygenxml.com/download_oxygenxml_editor.html*).

Αφού εγκαταστήσουμε το προαναφερθέν λογισμικό, προβαίνουμε στην διαδικασία της εξαγωγής των αρχείων της εφαρμογής, που βρίσκονται στο παρεχόμενο συμπιεσμένο αρχείο WebFreeMind.rar (το οποίο είναι διαθέσιμο για downloading στην ακόλουθη ιστοσελίδα: *http://www.dbnet.ece.ntua.gr/downloads/WebFreeMind.htm*). Σε καμία περίπτωση δεν πρέπει να λησμονήσουμε την επιλογή της ιδιότητας του "full path extraction", προτού προχωρήσουμε με την αποσυμπίεση. Κάνοντας διπλό κλικ στο νέο εικονίδιο WebFreeMind (που μόλις εμφανίστηκε στην Επιφάνεια εργασίας / Desktop), εκτελείται η εφαρμογή και εμείς είμαστε πλέον σε θέση να εκμεταλλευτούμε τις δυνατότητες που αυτή μας παρέχει, για να ικανοποιήσουμε τις ανάγκες μας που αφορούν την ανάπτυξη εμπλουτιζόμενων από διαδικτυακή πληροφορία mindmaps.

Οφείλουμε τέλος να τονίσουμε, ότι πρέπει να κάνουμε τις κατάλληλες ρυθμίσεις στο τείχος προστασίας (firewall) του υπολογιστή μας, ώστε αυτό να επιτρέπει την πραγματοποίηση τόσο εισερχόμενων όσο και εξερχόμενων συνδέσεων (grant full access – inbound / outbound connections) της εφαρμογής WebFreeMind με το Διαδίκτυο.

6

Έλεγχος

Το συγκεκριμένο κεφάλαιο ασχολείται με την αξιολόγηση του κατασκευασθέντος συστήματος. Αρχικά, παρουσιάζουμε με συνοπτικό τρόπο την προτιμηθείσα μεθοδολογία ελέγχου, ενώ στην συνέχεια παραθέτουμε αναλυτικά τον διεξαχθέντα έλεγχο της εφαρμογής μας.

6.1 Μεθοδολογία ελέγχου

Ο έλεγχος του συστήματος πραγματοποιήθηκε βασιζόμενος σε ένα σενάριο λειτουργίας. Ας περιγράψουμε εδώ σύντομα το εν λόγω σενάριο, προκειμένου να προβάλουμε στον μελετητή της παρούσας διπλωματικής εργασίας ένα αληθινό περιστατικό χρήσης. Σημειώνουμε, πως χάριν της ευκολίας κατανόησης, το παράδειγμα αυτό θα είναι σχετικά απλό.

Υποθέτουμε, ότι επιθυμούμε την δημιουργία ενός mindmap που έχει να κάνει με την *Επιστήμη των Υπολογιστών* ως αντικείμενο. Για τον σκοπό αυτό, θα αξιοποιήσουμε αρχικά την εφαρμογή μας ως εργαλείο χαρτογράφησης εννοιών (-σκέψεων-ιδεών), καθότι αυτή έχει οικοδομηθεί πάνω στο FreeMind. Έτσι, τοποθετούμε ως κεντρική ιδέα τον όρο “Computer Science”, και ως κύρια σημεία τις υποκατηγορίες “Theoretical”, “Practical”, “Technical” & “Applied”. Επιπλέον, ως σημεία εξειδίκευσης του τομέα “Theoretical (Computer Science)” συμπεριλαμβάνουμε τα εξής: “Mathematical Logic”, “Automata Theory”, “Number Theory”, “Graph Theory”, “Type Theory”, “Category Theory”, “Computational Geometry”& “Quantum Computing Theory”. Επίσης, για τον τομέα “Practical (Computer Science)”

επιλέγουμε τα ακόλουθα υποσημεία: “Algorithms”, “Data Structures”, “Programming Languages” & “Software Engineering”, όπου ο κόμβος “Algorithms” έχει ως παιδί τον πρόσθετο κόμβο “Analysis of Algorithms” και εκείνος με ονομασία “Programming Languages” τον κόμβο “Compilers”. Ακόμα, ο τομέας “Technical (Computer Science)” θεωρούμε πως διαθέτει ως child-nodes τους παρακάτω: “Digital Logic”, “Microarchitecture” & “Multiprocessing”. Εκτός αυτών, ο τομέας “Applied (Computer Science)” διακρίνεται στις παρατιθέμενες υποκατηγορίες: “Databases”, “Internet”, “Bioinformatics”, “Cognitive Science”, “Computational Chemistry”, “Computational Neuroscience”, “Computational Physics”, “Numerical Algorithms” & “Symbolic Mathematics”.

Έστω ότι μετέπειτα θέλουμε να πλοηγηθούμε στο Διαδίκτυο, προκειμένου να επισκεφθούμε οποιονδήποτε σχετικό ενδιαφέροντα ιστότοπο. Για τον σκοπό αυτό, ανοίγουμε τον ενσωματωμένο web browser που περικλείεται στο σύστημά μας και μεταβαίνουμε π.χ. στην τοποθεσία *http://www.computer.org/portal/site/ieeecs/index.jsp*, η οποία συνιστά την official IEEE Computer Society webpage. Είμαστε έτσι σε θέση, άμεσα και γρήγορα, να περιηγηθούμε στην συγκεκριμένη ιστοσελίδα και να πληροφορηθούμε για το ζήτημα που μας απασχολεί.

Επιπροσθέτως, θεωρούμε ότι στην συνέχεια επιθυμούμε να εμπλουτίσουμε τον χάρτη μας με γενικού τύπου πληροφορίες από το Internet, οι οποίες σχετίζονται με το θέμα “Computational Neuroscience”. Το μόνο που απαιτείται να κάνουμε, είναι να ενεργοποιήσουμε το παράθυρο της προηγμένης διαδικτυακής μετα-αναζήτησης, και αφού έχουμε επιλέξει το αντίστοιχο topic στον mindmap, καθώς και το αιτούμενο πλήθος αποτελεσμάτων, πυροδοτούμε την διαδικασία αναζήτησης, λαμβάνοντας τελικά μία λίστα με τα διάφορα σχετικά αποτελέσματα (τα οποία έχουν ταξινομηθεί και συγχωνευθεί καταλλήλως). Κανείς μπορεί να παρατηρήσει, ότι τα εν λόγω αποτελέσματα προέρχονται από ποικίλες και δημοφιλείς μηχανές αναζήτησης. Σημειώνουμε ότι η παρουσίασή τους δύναται να γίνει είτε μικτά, είτε ανά μηχανή προέλευσης, ενώ επίσης παρέχεται η δυνατότητα να τα εμφανίσουμε σε σύντομη ή εκτενή μορφή. Τα παρατιθέμενα checkboxes (ένα δίπλα σε κάθε result) επιτρέπουν την επιλογή όλων ή μίας ομάδας αυτών, ενώ τα result links είναι ενεργά, υπό την έννοια ότι κάνοντας κλικ σε οποιοδήποτε από αυτά, η αντίστοιχη ιστοσελίδα ανοίγει στον ενσωματωμένο browser. Τελευταίο βήμα αυτού του σταδίου συνιστά η μεταφορά των διαλεγμένων αποτελεσμάτων στον χάρτη (με την μορφή: result title ως node text – result URL ως node hyperlink).

Εκτός των παραπάνω, έστω ότι ακολούθως θέλουμε την λήψη επιστημονικών δημοσιεύσεων (από το Διαδίκτυο, αλλά και διασταυρωμένων – εμπλουτισμένων από το DBLP, όπου αυτό είναι δυνατό), οι οποίες σχετίζονται με το topic “Automata Theory”. Ενεργοποιούμε και πάλι το παράθυρο της προηγμένης διαδικτυακής μετα-αναζήτησης, μόνο που πέρα από την επιλογή του αντίστοιχου node στο mindmap, του επιθυμητού πλήθους αποτελεσμάτων και

την καταγραφή τυχόν πρόσθετων keywords, πρέπει να διευκρινίσουμε και το ότι η αναζήτηση αφορά papers και όχι (general) web pages. Τα αποτελέσματα που λαμβάνουμε μπορούν, όπως και πριν, να παρουσιαστούν mixed / per search engine και σε short / extended form. Εκείνο το οποίο πρέπει να τονίσουμε είναι, ότι όσα εξ αυτών εμφανίζονται με κόκκινο τίτλο, έχουν επίσης εντοπιστεί στην Βάση Δεδομένων του DBLP, επομένως διαθέτουν το ανάλογο κύρος, ενώ η ανάπτυξή τους στην πλήρη μορφή τους μας παρέχει περαιτέρω πρόσθετες πληροφορίες, οι οποίες έχουν ληφθεί από την συγκεκριμένη βάση δεδομένων. Προφανώς, και εδώ υπάρχει η δυνατότητα της επίσκεψης των αντίστοιχων ιστοσελίδων, της διαλογής των αποτελεσμάτων και της μεταφοράς ορισμένων από αυτά στον χάρτη μας.

Η ενσωμάτωση μίας καινούριας μηχανής αναζήτησης (εν προκειμένω της Yahoo Web Search) είναι το τελικό στάδιο του σεναρίου χρήσης που αναπτύσσουμε. Για να πραγματοποιήσουμε κάτι τέτοιο, αξιοποιούμε τον οδηγό εγκατάστασης που υπάρχει στην εφαρμογή. Σε αυτόν τον οδηγό εισάγουμε την ονομασία (Yahoo), το main URL (*http://search.yahoo.com/*), τον χρησιμοποιούμενο τύπο δεδομένων (web page) και τον βαθμό αξιολόγησης της εν λόγω μηχανής (5/5 αστέρια). Προαιρετικά, μπορούμε να πληκτρολογήσουμε τα κατάλληλα δεδομένα (script expansion, search filter & query expansion) που αντιστοιχούν σε τέσσερα υποψήφια στοιχεία (π.χ. date-based search support, domain-based search support, filetype-based search support, region-based search support, δηλαδή οι πιθανές επεκτάσεις για webpage search engines), προσδίδοντας προηγμένη λειτουργικότητα στη νέα υπηρεσία αναζήτησης. Αναφέρουμε, ότι μπορούμε να εισάγουμε από ένα έως τέσσερα τέτοια σχετικά στοιχεία, ή ακόμα και κανένα, σημειώνουμε δε, ότι για λόγους απλούστευσης (και δίχως βλάβη της γενικότητας) στο παράδειγμα αυτό αποφεύγουμε κάποια τέτοιου είδους εισαγωγή.

Εν συνεχεία, αφού επικυρώσουμε τις προηγούμενες επιλογές μας στην εμφανιζόμενη καρτέλα του installation wizard, ανοίγει ένα παράθυρο ιδιότυπου web browser, ο οποίος επισκέπτεται το παρασχεθέν URL της εγκαθιστάμενης μηχανής. Πατώντας το κομβίο που αφορά στην πρόγνωση του search script και της query parameter, παρουσιάζεται στον χρήστη ένα frame, το οποίο περιλαμβάνει μία λίστα με τις αντίστοιχες πιθανές τιμές, το συνιστώμενο ζεύγος τιμών, καθώς και δύο πεδία για πληκτρολόγηση των δικών μας τιμών των παραμέτρων. Στην συγκεκριμένη περίπτωση, το recommended pair (/search και p=) είναι ταυτόχρονα και το κατάλληλο, οπότε μεταβαίνουμε στην επόμενη φάση της εγκατάστασης, όπου ο installation browser εκτελεί μία πειραματική αναζήτηση, βάσει του κατασκευασθέντος search URL (*http://search.yahoo.com/search?p=keyword*) και μίας προκαθορισμένης keyword (*Alan Turing*).

Στην σελίδα αποτελεσμάτων που εμφανίζεται, ο χρήστης καλείται να μαρκάρει ένα προς ένα τα στοιχεία που απαρτίζουν ένα πλήρες αποτέλεσμα (τίτλος – διαδικτυακός σύνδεσμος –

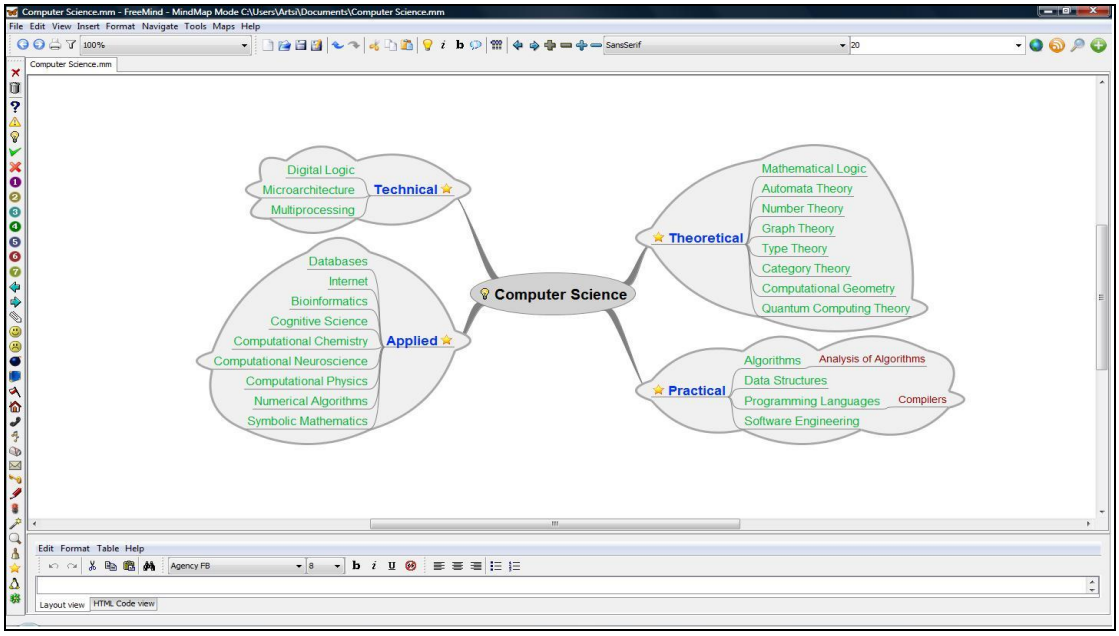
περιγραφή) και κάθε φορά να επικυρώσει την πρόβλεψη των υπόλοιπων αντίστοιχων result subelements από το σύστημα ή να την βελτιώσει, επιβάλλοντας την εφαρμογή κάποιων από τους πρόσθετους διαθέσιμους περιορισμούς. Η συγκεκριμένη φάση, η οποία εμπεριέχει την τεχνική του screen-scraping και βασίζεται σε HTML parsing, συνιστά και την τελευταία στην διαδικασία ενσωμάτωσης, ενώ εφόσον όλα έχουν εξελιχθεί ομαλά (δίχως κάποιο σφάλμα από πλευράς χρήστη ή κάποιο πρόβλημα τεχνικής φύσεως από την πλευρά του παροχέα της εν λόγω υπηρεσίας αναζήτησης) η καινούρια μηχανή έχει εγκατασταθεί και είναι πλέον έτοιμη προς εκμετάλλευση.

Για τον έλεγχο της ορθής & ομαλής λειτουργίας και της αρμονικής ενσωμάτωσης της νεοεισαχθείσας μηχανής αναζήτησης, εκτελούμε κατά τα γνωστά μία αναζήτηση, έχοντας επιλέξει τον κόμβο, ο οποίος φέρει το κείμενο “Software Engineering”. Στην επιστρεφόμενη λίστα αποτελεσμάτων, παρατηρούμε ότι υπάρχουν αναμεμειγμένα και αποτελέσματα από αυτή την μηχανή (Yahoo), επομένως, η εγκατάσταση κρίνεται ως επιτυχής.

6.2 Αναλυτική παρουσίαση ελέγχου

Στην παρούσα ενότητα παρουσιάζεται αναλυτικά (με χρήση screenshots) ο έλεγχος του συστήματος, ο οποίος ακολουθεί το σενάριο που περιγράφθηκε προηγουμένως. Πρέπει να τονίσουμε, ότι η συγκεκριμένη ενότητα μπορεί να διαδραματίσει και τον ρόλο του εγχειριδίου χρήσης της εφαρμογής μας.

Αρχικά κατασκευάζουμε τον χάρτη μας, έχοντας ως κεντρική έννοια την “Computer Science”, ενώ οι υπόλοιποι κόμβοι είναι εκείνοι που έχουν προαναφερθεί κατά την ανάπτυξη του usage scenario:

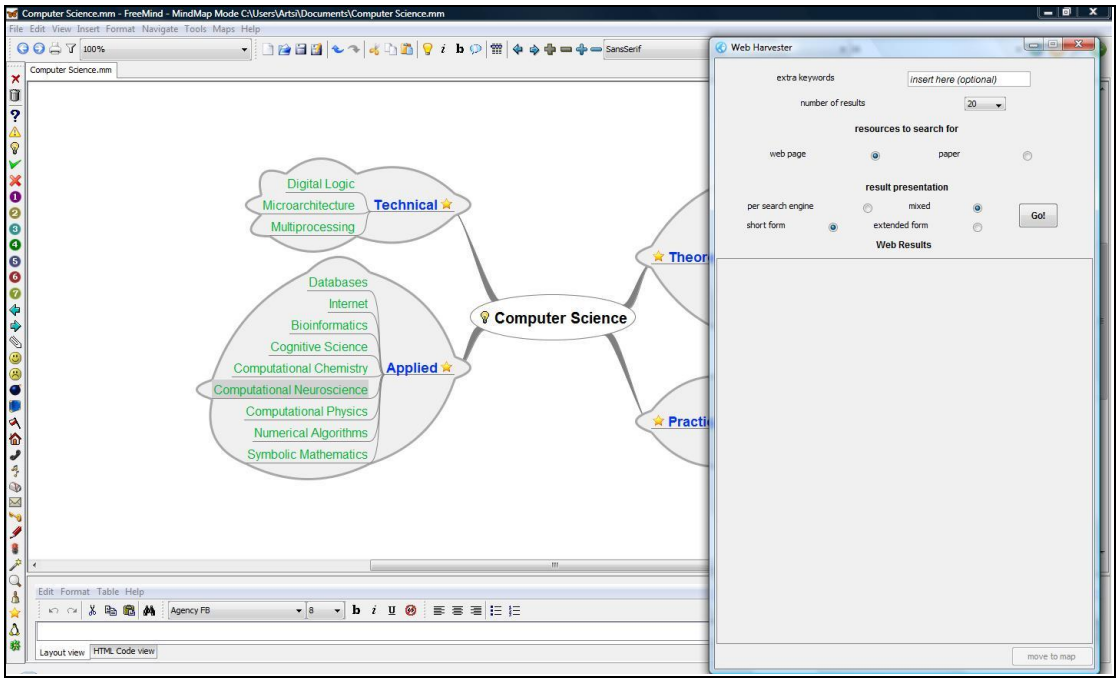


Στιγμιότυπο οθόνης 6.1 – Δημιουργία ενός mindmap

Στην συνέχεια ανοίγουμε τον ενσωματωμένο φυλλομετρητή ιστοσελίδων και πλοηγούμαστε στο Διαδίκτυο, επισκεπτόμενοι τον ιστότοπο της IEEE Computer Society:

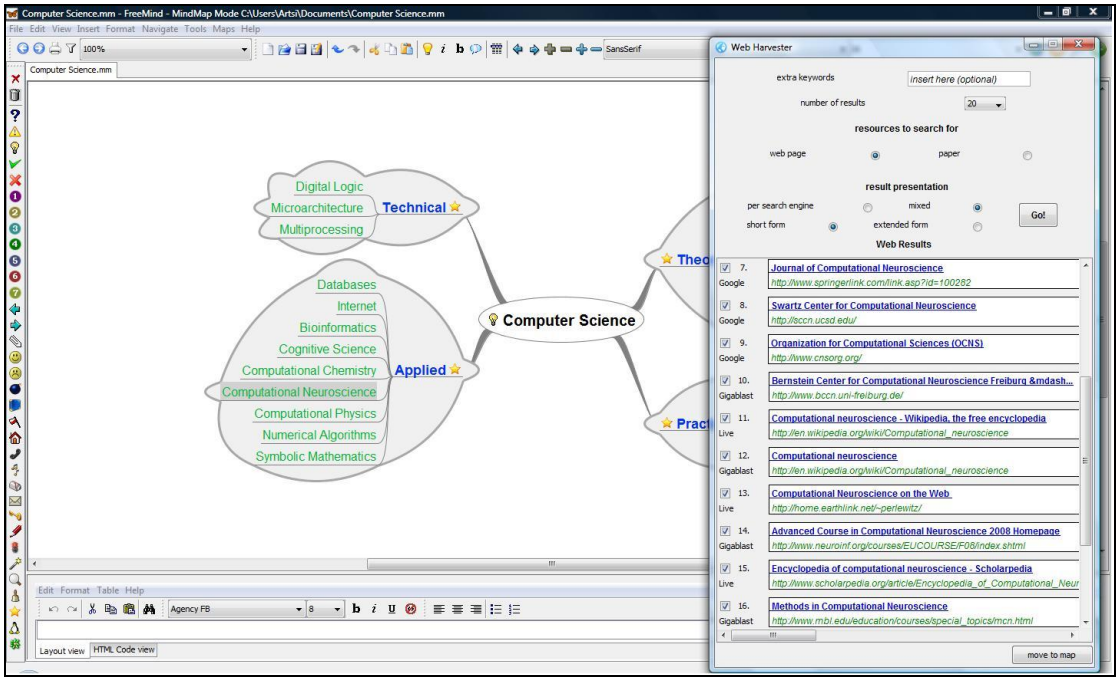
Στιγμιότυπο οθόνης 6.2 – Επίσκεψη μίας ιστοσελίδας με τον εσωτερικό Web Browser

Στην συνέχεια, προετοιμαζόμαστε για την χρήση της υπηρεσίας προηγμένης διαδικτυακής μετα-αναζήτησης (όσον αφορά τον όρο "Computational Neuroscience"), ενεργοποιώντας το αντίστοιχο παράθυρο και κάνοντας τις επιθυμητές επιλογές:



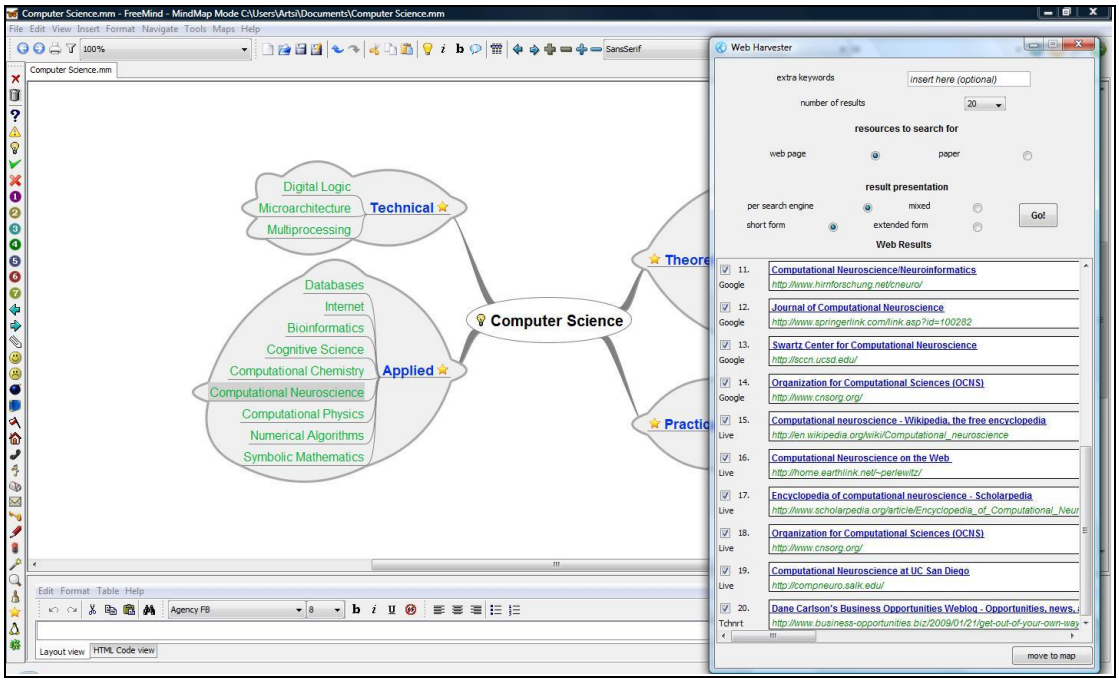
Στιγμιότυπο οθόνης 6.3 – Προετοιμασία εκτέλεσης μίας αναζήτησης στο Internet

Ακολούθως, πυροδοτούμε την αναζήτηση και λαμβάνουμε την λίστα των αποτελεσμάτων, τα οποία προέρχονται από διάφορες μηχανές αναζήτησης:



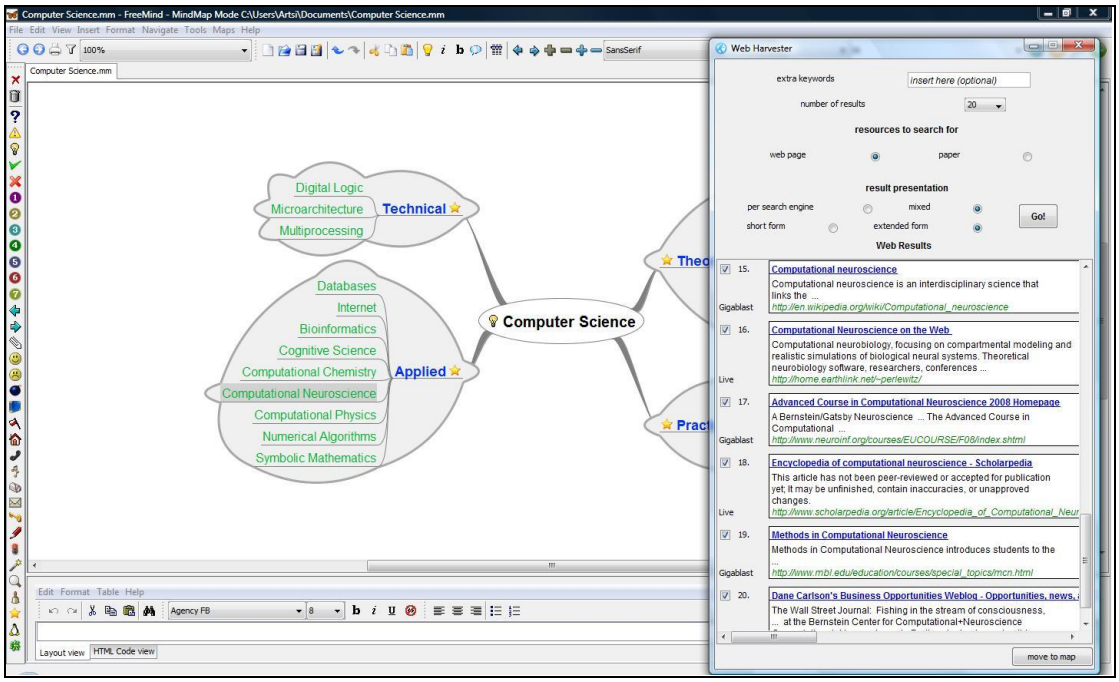
Στιγμιότυπο οθόνης 6.4 – Αποτελέσματα αναζήτησης (μεικτά – σύντομη μορφή)

Τα αποτελέσματα μπορούν επίσης να εμφανιστούν και ανά μηχανή προέλευσης (αλφαβητική ταξινόμηση των μηχανών) αντί του ανάμεικτου τρόπου παρουσίασης. Έτσι μπορούμε να δούμε συγκεντρωτικά τα αποτελέσματα που επέστρεψε η κάθε μία μηχανή αναζήτησης:



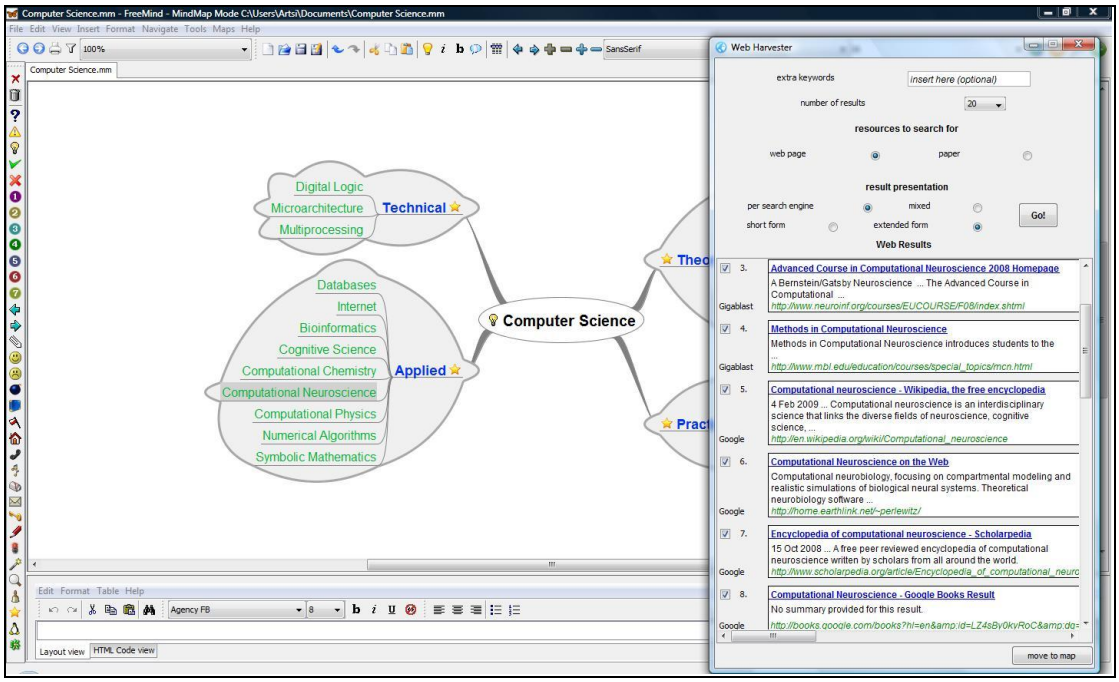
Στιγμιότυπο οθόνης 6.5 – Αποτελέσματα αναζήτησης (ανά μηχανή – σύντομη μορφή)

Μία άλλη δυνατή όψη των αποτελεσμάτων, όπως έχει προειπωθεί, είναι η ανεπτυγμένη (δηλαδή εκείνη, όπου εκτός του τίτλου και του διαδικτυακού συνδέσμου του κάθε αποτελέσματος, προβάλλεται και η εκάστοτε περίληψη):



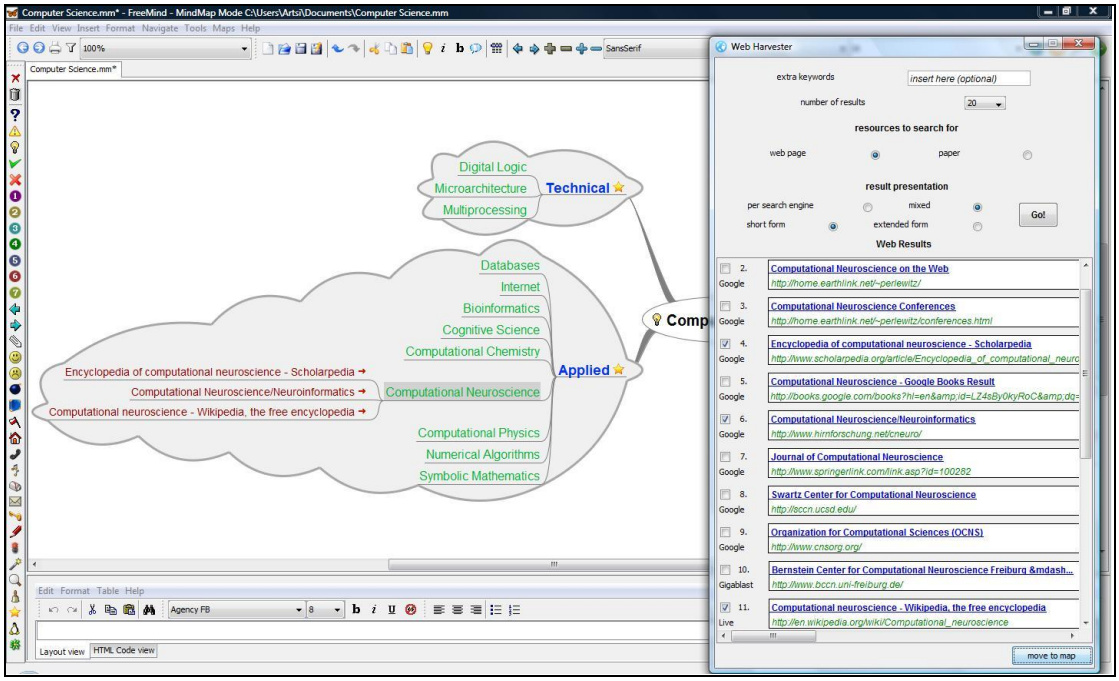
Στιγμιότυπο οθόνης 6.6 – Αποτελέσματα αναζήτησης (μεικτά – εκτενής μορφή)

Και στην περίπτωση των πλήρως ανεπτυγμένων αποτελεσμάτων, αυτά μπορούν να δίνονται στον χρήστη είτε αναμεμειγμένα (όπως παραπάνω), είτε ανά μηχανή προέλευσης:



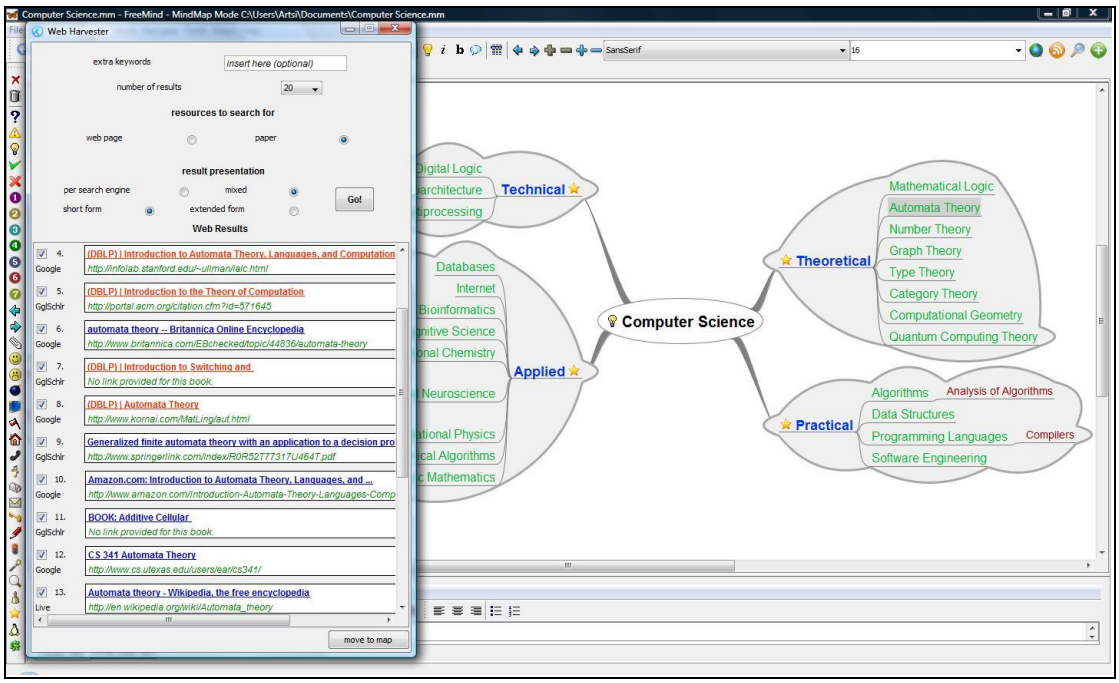
Στιγμιότυπο οθόνης 6.7 – Αποτελέσματα αναζήτησης (ανά μηχανή – εκτενής μορφή)

Τώρα, επιλέγουμε τα αποτελέσματα που θεωρούμε ως ενδιαφέροντα (με την βοήθεια των παρακείμενων checkboxes) και τα μεταφέρουμε στον χάρτη μας (με το αντίστοιχο button):



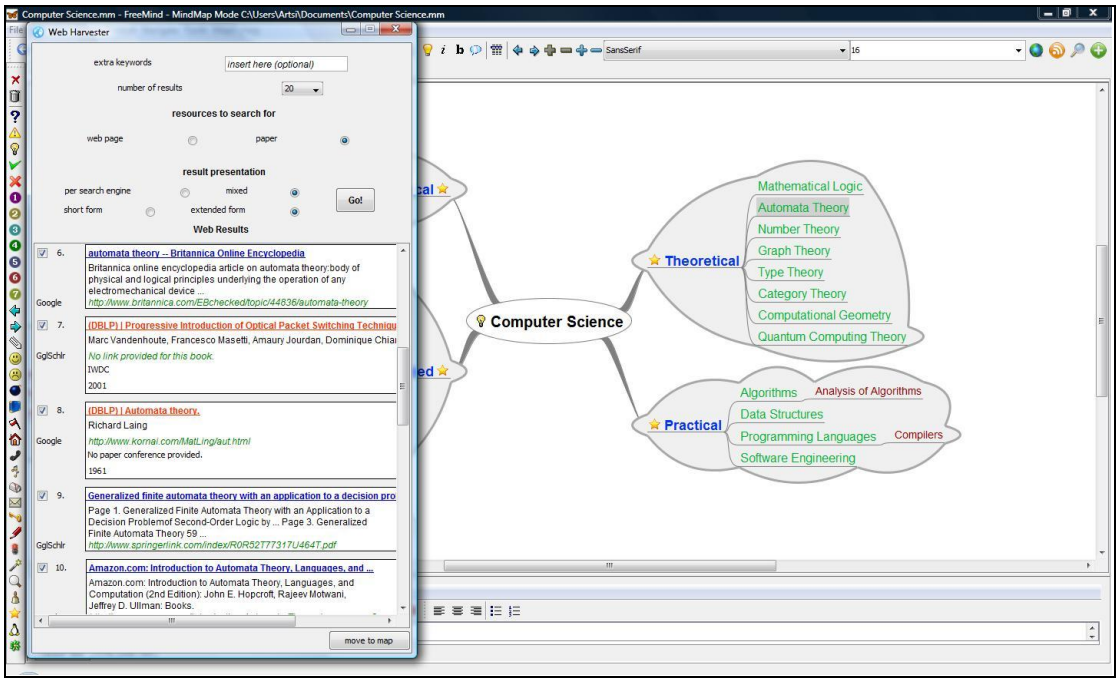
Στιγμιότυπο οθόνης 6.8 – Διαλογή και μετακίνηση επιλεγμένων results στον mindmap

Ακολούθως, διεξάγουμε μία αναζήτηση στο Διαδίκτυο με θέμα “Automata Theory”, στην οποία αυτή την φορά όμως, ορίζουμε ως επιθυμητό τύπο δεδομένων τις επιστημονικές δημοσιεύσεις. Τα διασταυρωμένα με το DBLP αποτελέσματα εμφανίζονται με κόκκινο τίτλο, ενώ οι υπόλοιπες λειτουργικότητες της αναζήτησης παραμένουν οι ίδιες:



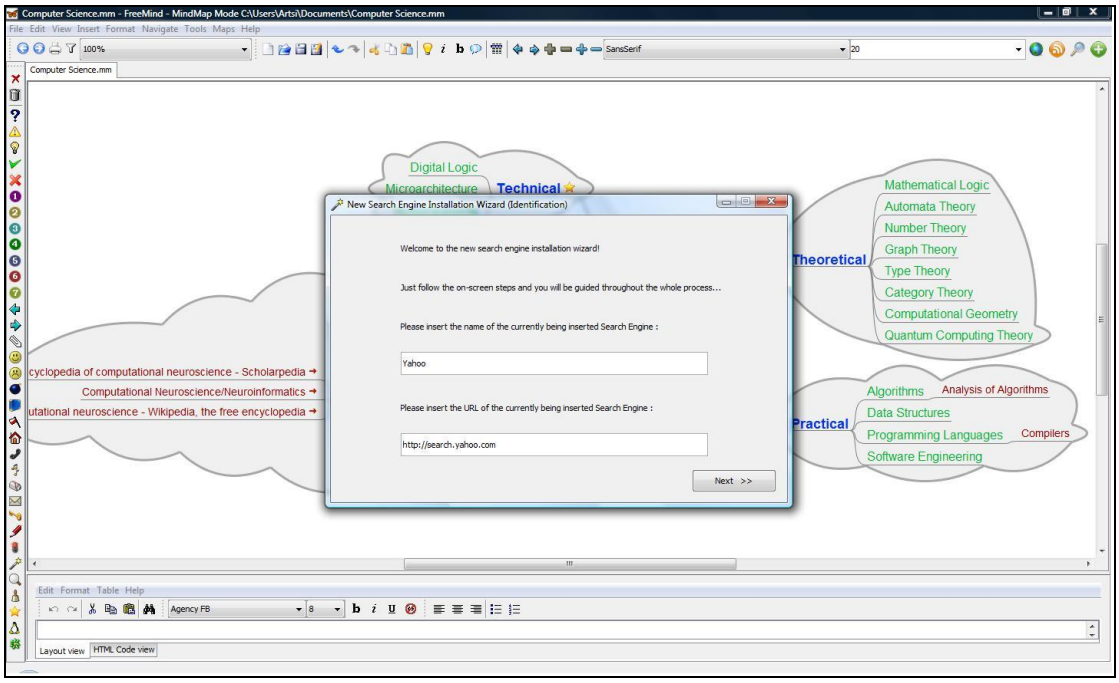
Στιγμιότυπο οθόνης 6.9 – Αναζήτηση για papers και επισήμανση DBLP results

Στην συνέχεια, αναπτύσσουμε τα αποτελέσματα, προκειμένου να φανεί η πρόσθετη πληροφορία που περικλείουν. Παρατηρούμε, ότι όσα έχουν βρεθεί και στο DBLP είναι εμπλουτισμένα και με άλλα δεδομένα:



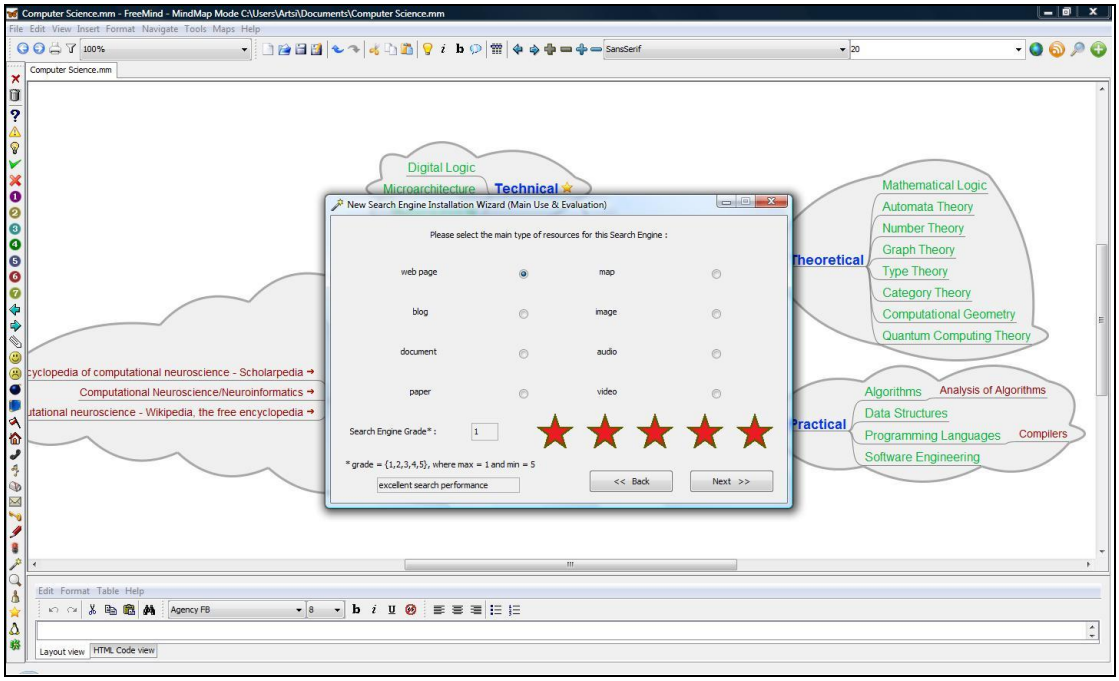
Στιγμιότυπο οθόνης 6.10 – Αποτελέσματα papers (χρήση DBLP – εκτενής μορφή)

Τώρα, θα ασχοληθούμε με την διαδικασία εγκατάστασης μίας καινούριας μηχανής αναζήτησης στο σύστημά μας. Εκκινούμε τον installation wizard και εισάγουμε αρχικά την ονομασία και τον βασικό διαδικτυακό σύνδεσμο στην πρώτη καρτέλα:



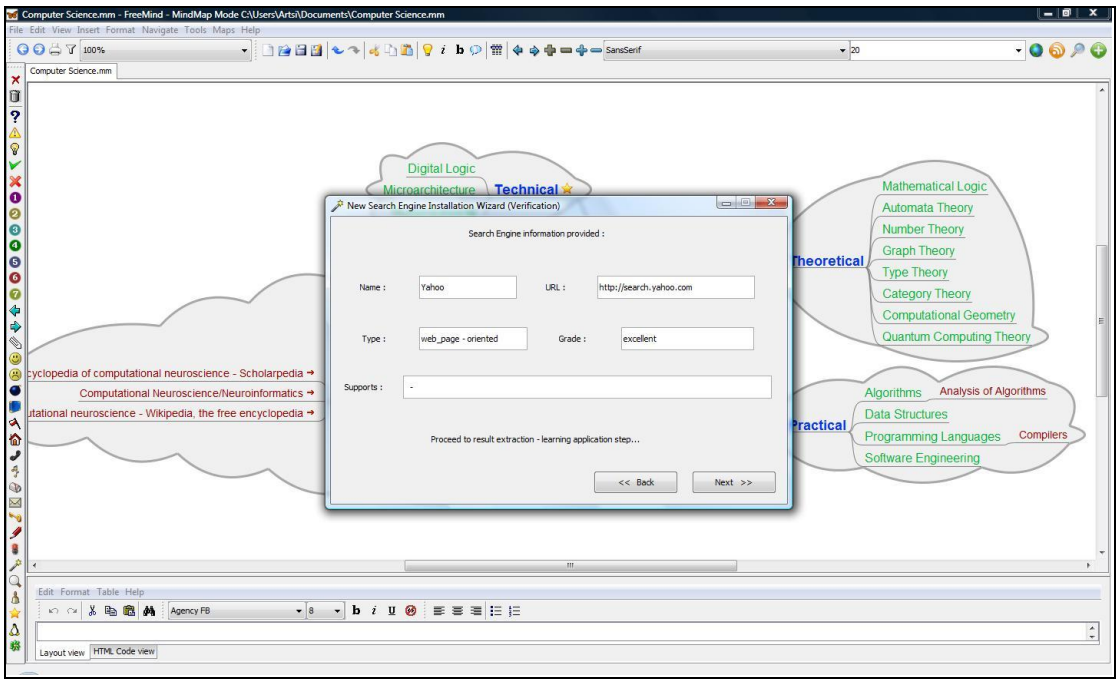
Στιγμιότυπο οθόνης 6.11 – Εγκατάσταση νέας μηχανής αναζήτησης (όνομα – κύριο URL)

Μετά, προσδιορίζουμε τον χρησιμοποιούμενο τύπο δεδομένων για την ενσωματούμενη μηχανή αναζήτησης, καθώς και τον βαθμό αξιολόγησής της:



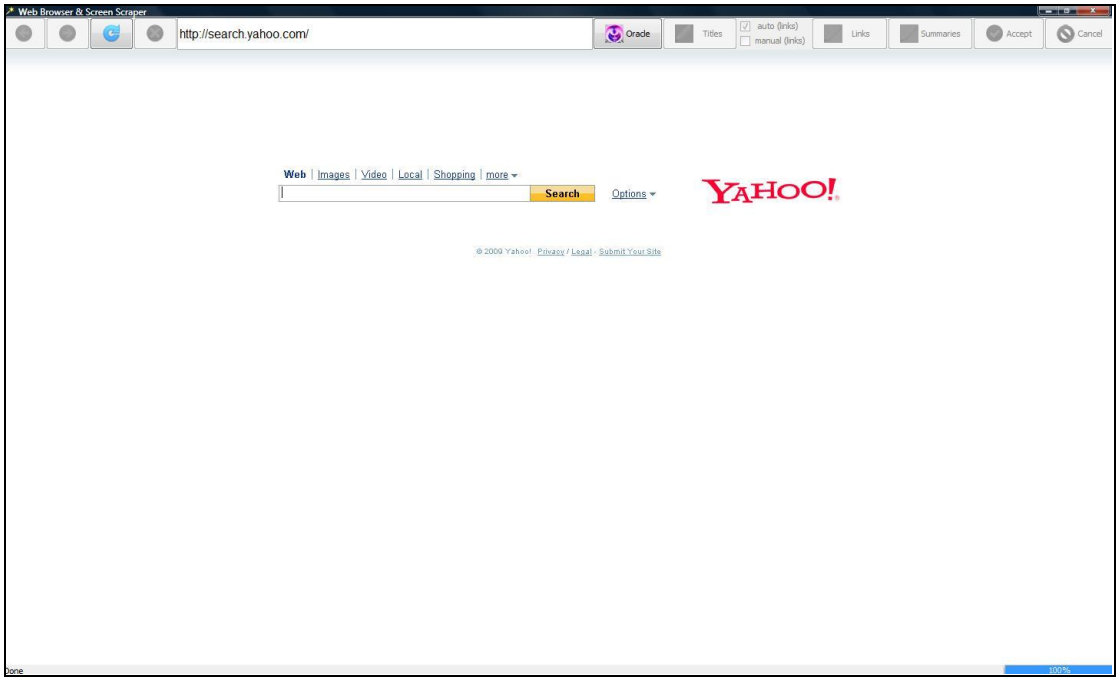
Στιγμιότυπο οθόνης 6.12 – Εγκατάσταση νέας μηχανής αναζήτησης (χρήση – βαθμολογία)

Στα πλαίσια αυτού του παραδείγματος, παραλείπουμε την καρτέλα που αφορά την προσθήκη χαρακτηριστικών προηγμένης αναζήτησης και προχωρούμε αμέσως στην επόμενη, η οποία δείχνει στον χρήστη τα μέχρι στιγμής καταχωρημένα στοιχεία και τον καλεί να τα επιβεβαιώσει:



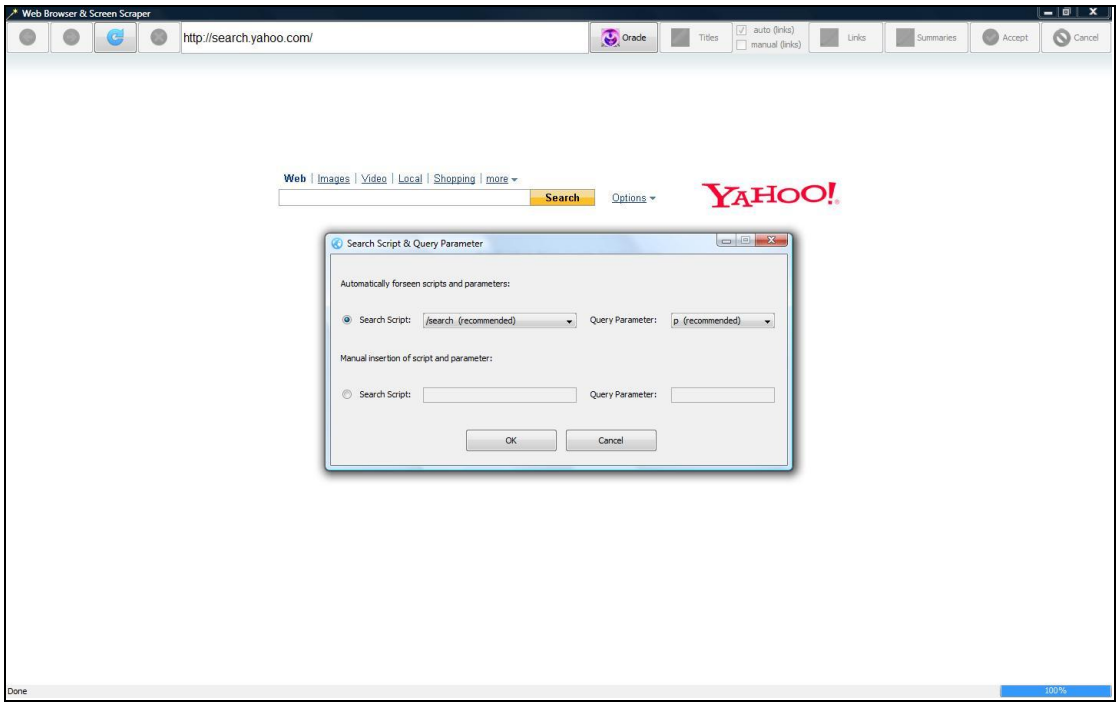
Στιγμιότυπο οθόνης 6.13 – Εγκατάσταση νέας μηχανής αναζήτησης (επικύρωση στοιχείων)

Έπειτα, ανοίγει ο ειδικός φυλλομετρητής ιστοσελίδων, ο οποίος αξιοποιείται αποκλειστικά στην διαδικασία της εγκατάστασης, και επισκέπτεται το URL που δώσαμε λίγο νωρίτερα:



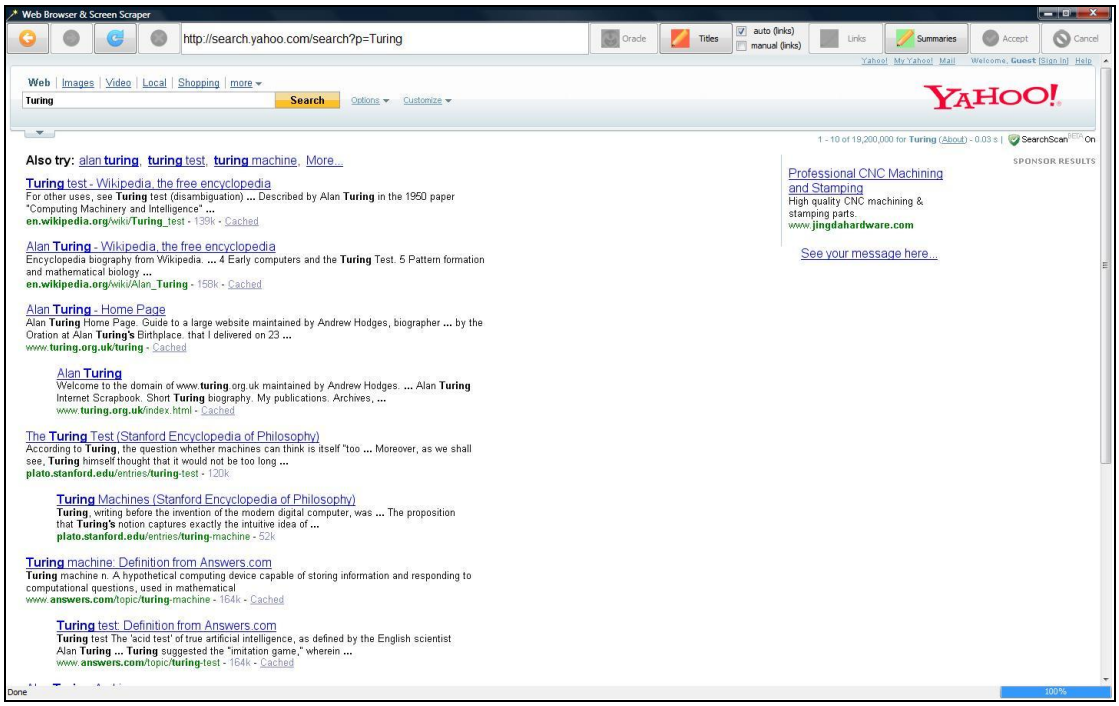
Στιγμιότυπο οθόνης 6.14 – Installation browser και μετάβαση στην ιστοσελίδα της μηχανής

Ύστερα, συνεχίζουμε με το στάδιο πρόβλεψης του search script και της query parameter, ζεύγος το οποίο στην προκειμένη περίπτωση μαντεύεται αυτομάτως ορθά από το ίδιο το σύστημα, επομένως δεν απαιτείται η παροχή κάποιων δεδομένων από πλευράς μας:



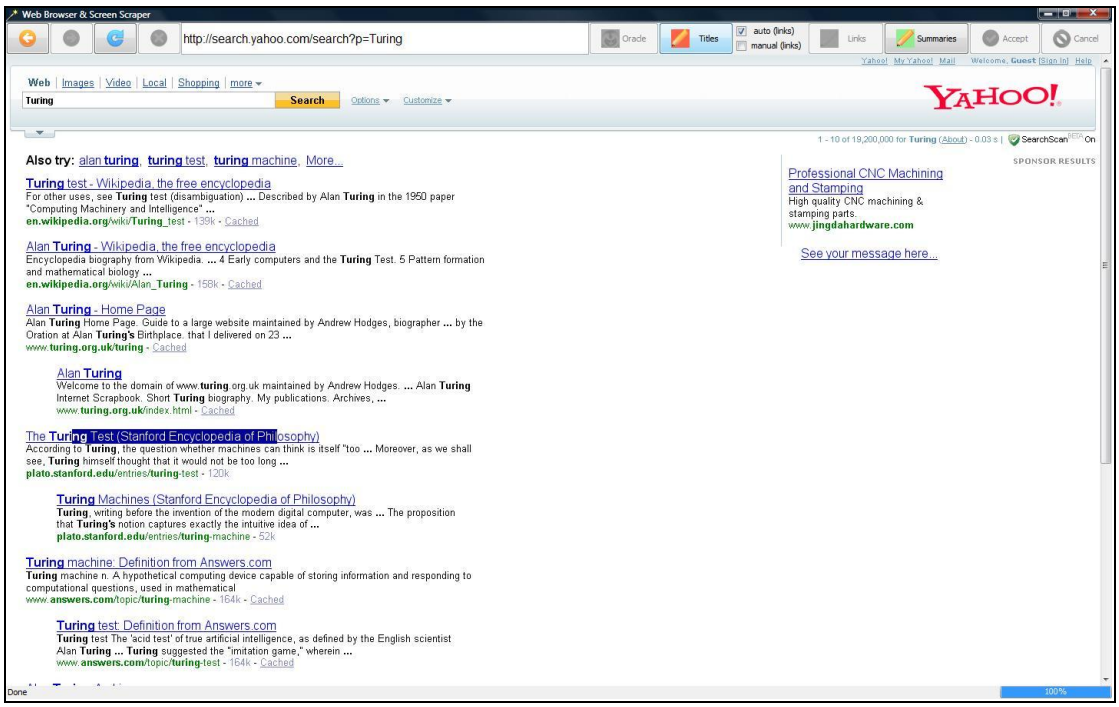
Στιγμιότυπο οθόνης 6.15 – Αυτοματοποιημένη πρόγνωση των search script & query parameter

Ακολούθως, ο browser φορτώνει αυτόματα την σελίδα αποτελεσμάτων που επιστρέφει η εν λόγω μηχανή αναζήτησης, αφού υποβάλλει προς αυτήν μία query με το προκαθορισμένο keyword “Turing”:



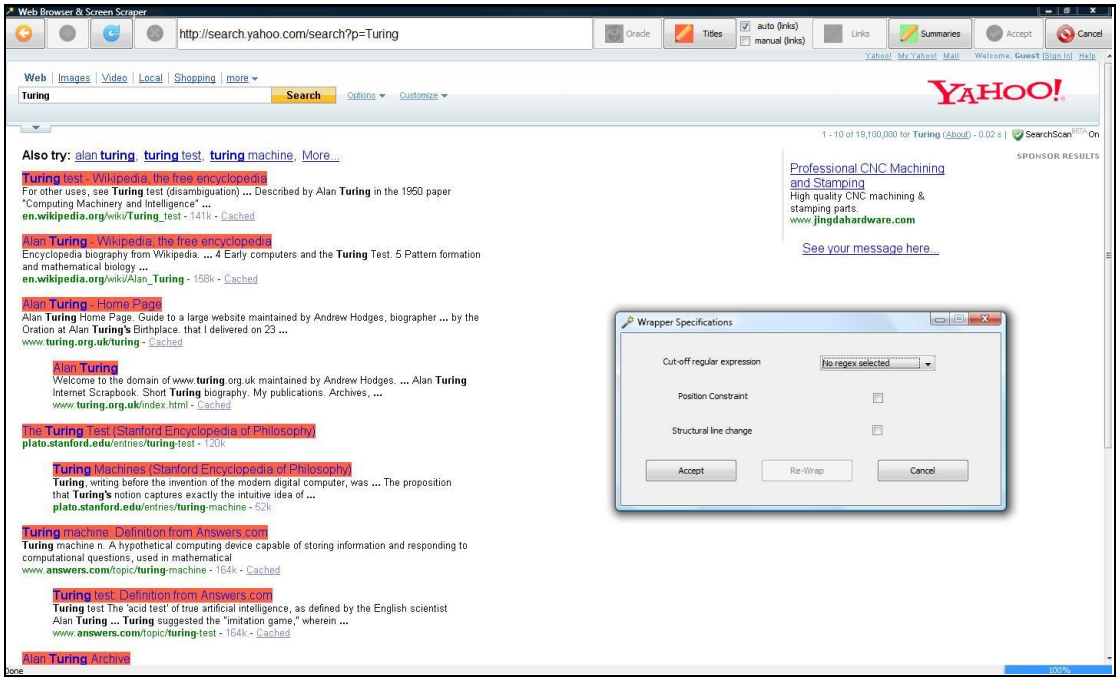
Στιγμιότυπο οθόνης 6.16 – Αυτόματη υποβολή search request & λήψη result list

Πλέον, εκκινούμε την διαδικασία του screen-scraping, το πρώτο μέρος της οποίας αφορά την εκμάθηση ανάγνωσης των τίτλων των αποτελεσμάτων από την μηχανή. Για τον σκοπό αυτό, επιλέγουμε ένα αξιόλογο τμήμα ενός result title και μετά πατάμε το αντίστοιχο button:



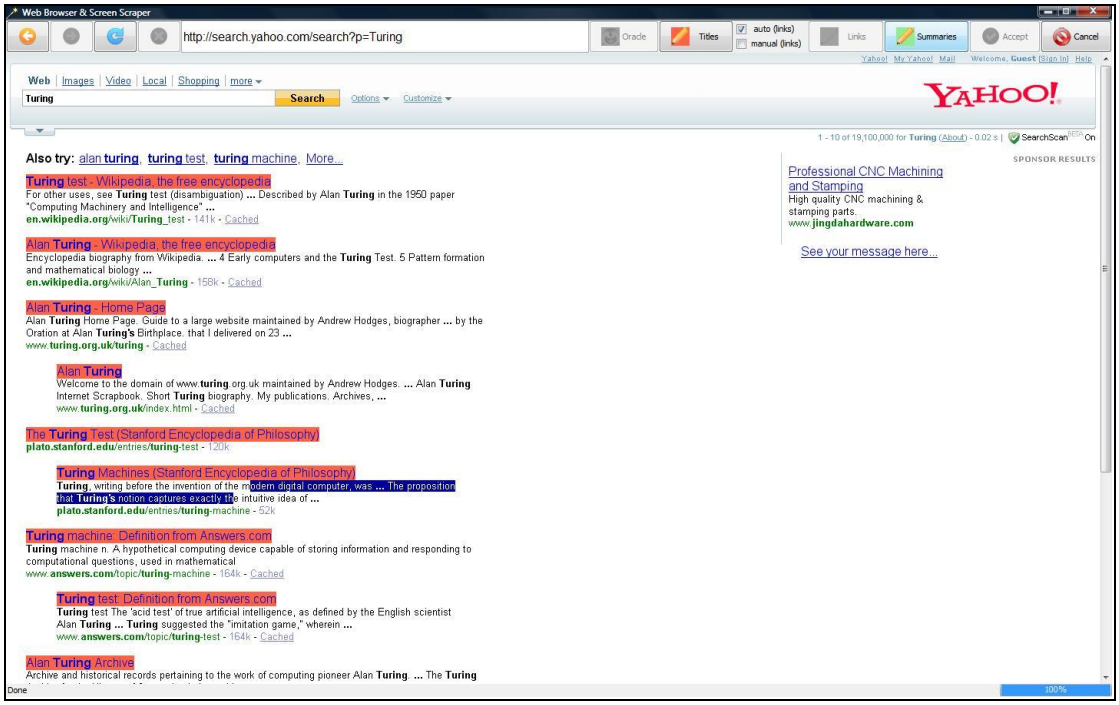
Στιγμιότυπο οθόνης 6.17 – Μαρκάρισμα και δήλωση του τίτλου ενός αποτελέσματος

Το σύστημα επιχειρεί να «διαβάσει» όλους τους τίτλους των αποτελεσμάτων της συγκεκριμένης ιστοσελίδας και τους επισημαίνει (με κόκκινο χρώμα), ενώ εμείς καλούμαστε να επικυρώσουμε, να διορθώσουμε (με την βοήθεια πρόσθετων περιορισμών) ή να ακυρώσουμε την προσπάθειά του:



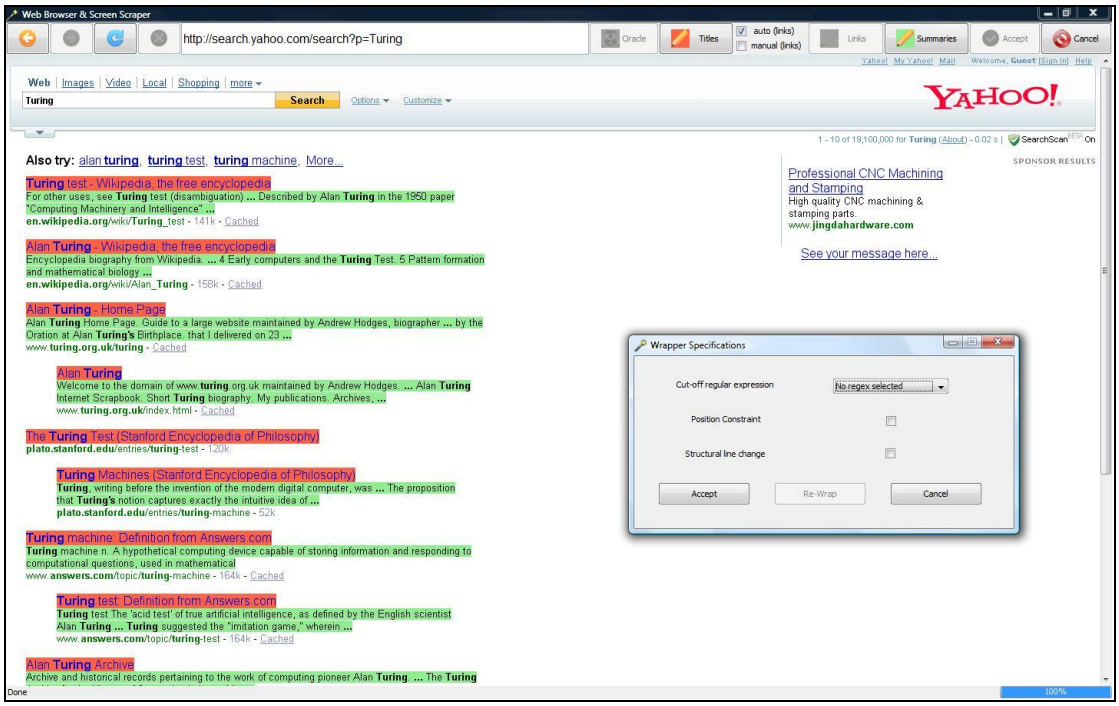
Στιγμιότυπο οθόνης 6.18 – Εντοπισμός όλων των result titles από το σύστημα

Επαναλαμβάνουμε τα δύο τελευταία βήματα με όμοιο τρόπο, προκειμένου να «διδάξουμε» στην εφαρμογή την ανάγνωση και των περιλήψεων των αποτελεσμάτων. Έτσι, αρχικά επιλέγουμε ένα σημαντικό μέρος ενός τέτοιου στοιχείου και έπειτα πατάμε το σχετικό button:



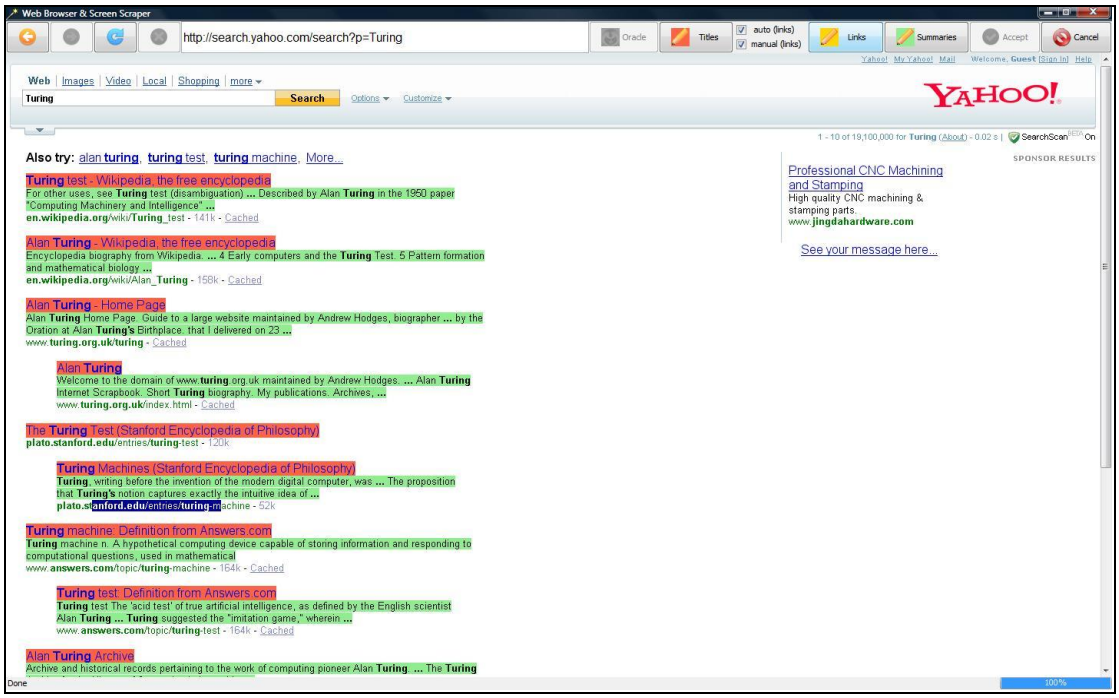
Στιγμιότυπο οθόνης 6.19 – Μαρκάρισμα και δήλωση της περίληψης ενός αποτελέσματος

Στον installation browser τώρα φορτώνεται η προηγούμενη σελίδα, με την διαφορά ότι έχουν αυτομάτως σημειωθεί (με πράσινο χρώμα) όλα τα αντίστοιχα στοιχεία που το σύστημα εκλαμβάνει ως περιλήψεις αποτελεσμάτων. Σε περίπτωση λάθους, μας δίνεται η δυνατότητα να προβούμε σε τροποποιήσεις, αξιοποιώντας τα παρεχόμενα εργαλεία:



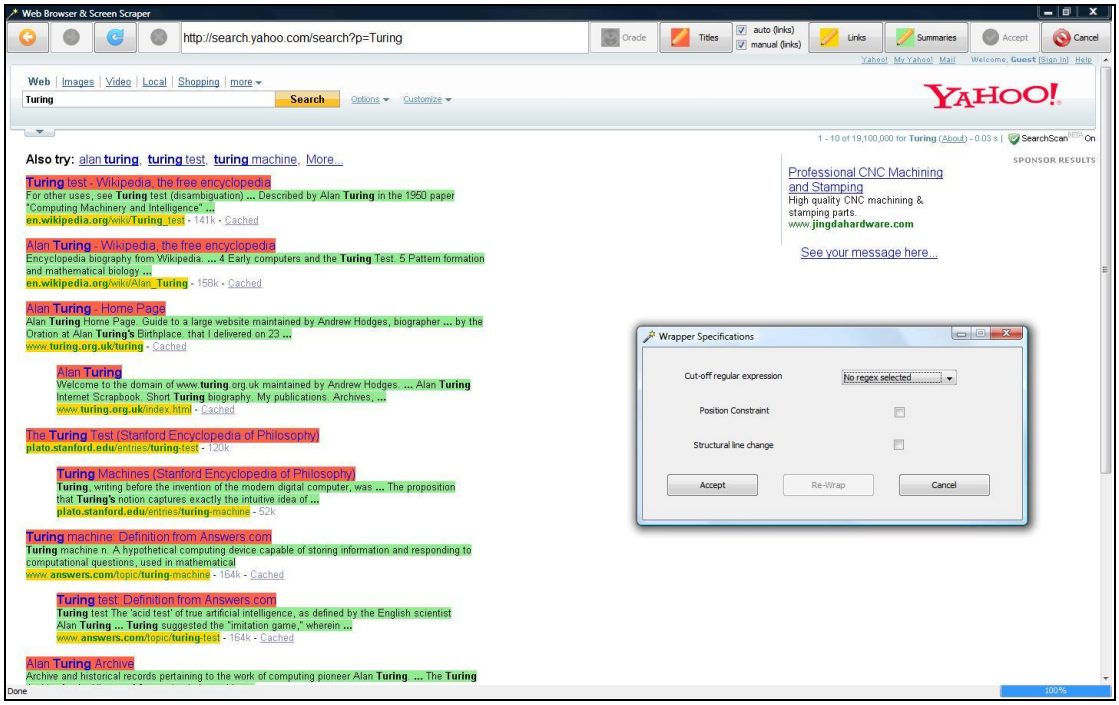
Στιγμιότυπο οθόνης 6.20 – Εντοπισμός όλων των result summaries από το σύστημα

Εργαζόμενοι ομοίως, κι έχοντας ως σκοπό την υπόδειξη της ανάγνωσης και των συνδέσμων των αποτελεσμάτων, επιλέγουμε ένα επαρκές τμήμα ενός σχετικού στοιχείου και μετά το «υποβάλλουμε»:



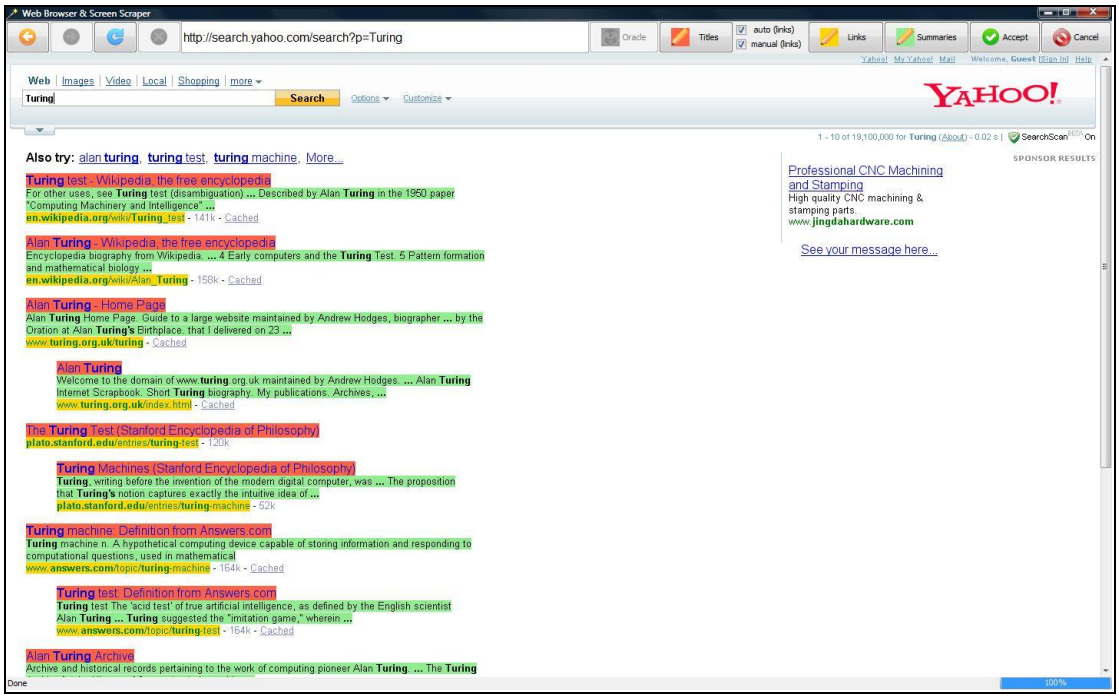
Στιγμιότυπο οθόνης 6.21 – Μαρκάρισμα και δήλωση του URL ενός αποτελέσματος

Στο σημείο αυτό, λαμβάνουμε την αρχική ιστοσελίδα με τα αποτελέσματα, μόνο που τώρα εκτός των τίτλων και των περιγραφών, έχουν τονισθεί (με κίτρινο χρώμα) και τα αντίστοιχα URLs. Σε κάθε στάδιο, μπορούμε βέβαια να κάνουμε τις ενδεχόμενες αναγκαίες διορθώσεις:



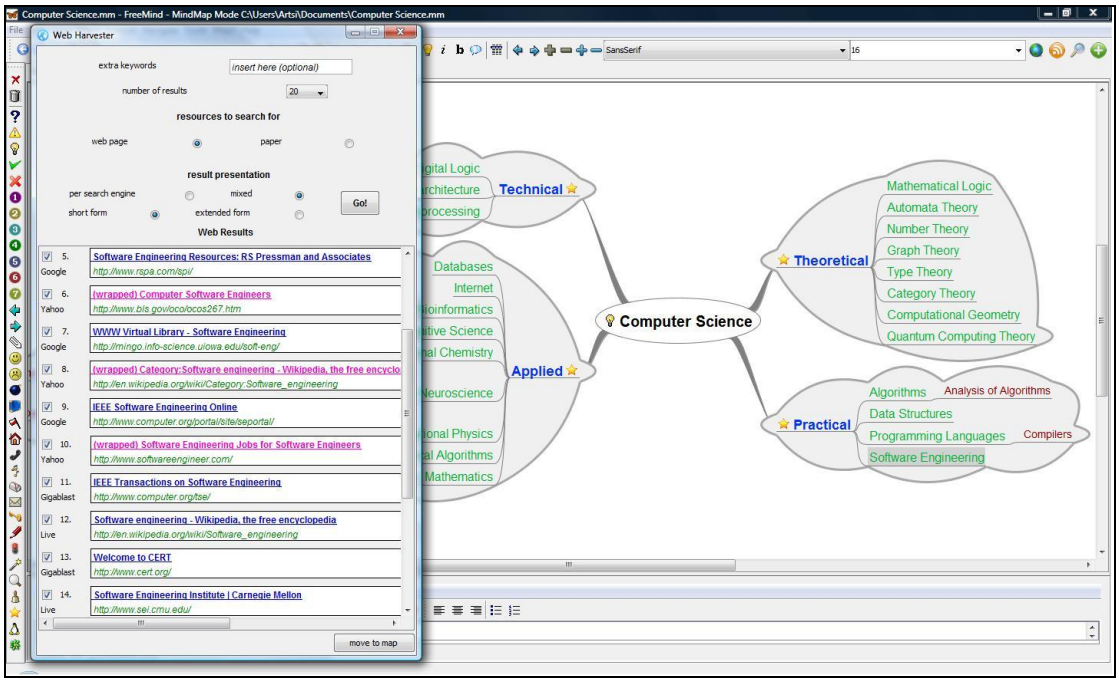
Στιγμιότυπο οθόνης 6.22 – Εντοπισμός όλων των result links από το σύστημα

Η καινούρια μηχανή αναζήτησης είναι πλέον έτοιμη να συνεισφέρει στην εφαρμογή μας. Η διαδικασία εγκατάστασης της εν λόγω μηχανής ολοκληρώνεται πατώντας το κουμπί "Accept", ενώ μπορεί να ακυρωθεί (και εδώ, αλλά και σε κάθε φάση) πατώντας το "Cancel":



Στιγμιότυπο οθόνης 6.23 – Ολοκλήρωση της εγκατάστασης της νέας μηχανής αναζήτησης

Τέλος, για τον έλεγχο της ομαλής λειτουργίας και της αρμονικής ενσωμάτωσης της νεοεισαχθείσας μηχανής, εκτελούμε μία αναζήτηση βασιζόμενη στο θέμα “Software Engineering”. Παρατηρούμε, ότι πλέον δεχόμαστε αποτελέσματα (αυτά με τον ροζ χρωματισμό) και από την μηχανή Yahoo Search, κάτι το οποίο σημαίνει πως η εγκατάσταση πραγματοποιήθηκε με επιτυχία:



Στιγμιότυπο οθόνης 6.24 – Αναζήτηση στο Internet μετά την εγκατάσταση καινούριας μηχανής

7

Επίλογος

Στο παρόν κεφάλαιο θα συνοψίσουμε την παρουσίαση της συγκεκριμένης διπλωματικής.

7.1 Σύνοψη και συμπεράσματα

Η διπλωματική αυτή εργασία ασχολήθηκε με τον τομέα της διασύνδεσης της τεχνικής του mind-mapping με τις υπηρεσίες που προσφέρει το Διαδίκτυο. Έχοντας ως βάση ένα εργαλείο χαρτογράφησης σκέψεων ανοικτού κώδικα, το FreeMind, ενσωμάτωσε τις απαραίτητες επεκτάσεις, προκειμένου αυτό να αξιοποιεί χρήσιμες λειτουργίες του Internet.

Πιο συγκεκριμένα, έδωσε την δυνατότητα πλοήγησης στο Διαδίκτυο μέσω του εσωτερικού web browser, ο οποίος ήταν συμβατός με όλα τα υπάρχοντα σύγχρονα πρότυπα και τις σχετικές τεχνολογίες. Επιπροσθέτως, παρείχε έναν μηχανισμό προηγμένης διαδικτυακής μετα-αναζήτησης, θέτοντας το επιθυμητό ερώτημα προς διάφορες μηχανές και αποδίδοντας τα αποτελέσματα στον χρήστη συγχωνευμένα και ταξινομημένα (λαμβάνοντας υπόψη τόσο την βαθμολόγηση της εκάστοτε μηχανής όσο και την κατάταξη του κάθε αποτελέσματος). Στις πρόσθετες επιλογές του εν λόγω μηχανισμού συγκαταλέγονταν η επιλογή του επιστρεφόμενου πλήθους αποτελεσμάτων, ο προσδιορισμός τυχόν extra keywords, ο καθορισμός του χρησιμοποιούμενου τύπου δεδομένων (web pages ή papers), ενώ όσον αφορά την παρουσίαση υφίσταντο τέσσερεις συνδυασμοί για την εμφάνιση της λίστας των αποτελεσμάτων (μεικτά / ανά μηχανή προέλευσης – συντόμως / εκτενώς). Ακόμα, επέτρεπε στον χρήστη να διαλέγει ορισμένα από τα αποτελέσματα αυτά και να τα μεταφέρει στον

mindmap, προκειμένου να τον εμπλουτίζει. Ιδιαίτερης προσοχής έχριζαν τα αποτελέσματα που αφορούσαν επιστημονικές δημοσιεύσεις και είχαν διασταυρωθεί με την τηρούμενη Βάση Δεδομένων του DBLP. Στα συγκεκριμένα, τα διαθέσιμα δεδομένα ήταν περισσότερα, καθότι είχαν πραγματοποιηθεί οι σχετικές προσθήκες από την εν λόγω βάση δεδομένων. Επίσης, σημαντικό ενδιαφέρον (ιδίως για programmers / developers) απέπνεε η όλη διαδικασία εγκατάστασης καινούριων μηχανών αναζήτησης στην εφαρμογή μας, στην οποία ξεχώρισε η φάση όπου συντελείτο η πρόγνωση του search script και της query parameter για την κατασκευή του full search URL της νεοεισαγόμενης μηχανής, καθώς και εκείνη που εκμεταλλευόταν την μεθοδολογία του screen-scraping για την εκμάθηση του τρόπου ανάγνωσης των δομικών στοιχείων των αποτελεσμάτων στο σύστημα.

Συμπληρώνοντας τα παραπάνω, οφείλουμε να επισημάνουμε την αδυναμία της απόλυτα επιτυχούς αντιμετώπισης όλων των πιθανών περιπτώσεων ιστοσελίδων αποτελεσμάτων των διαφόρων μηχανών αναζήτησης κατά την wrapping process. Ο λόγος που συμβαίνει αυτό, είναι κυρίως η μεγάλη ποικιλομορφία της δόμησης της HTML των συγκεκριμένων webpages. Μία άλλη δυσκολία παρουσιάζεται στην πάντοτε ορθή πρόγνωση του search URL, πρόβλημα για το οποίο εντοπίσαμε ως κυρίαρχη αιτία την αυτοματοποιημένη άρνηση ορισμένων μηχανών αναζήτησης να μας επιτρέψουν την πρόσβαση στον σχετικό κώδικα, λόγω του γεγονότος ότι αντιλαμβάνονταν πως θέταμε τις αντίστοιχες αιτήσεις όχι ως χρήστες – άνθρωποι, αλλά ως μηχανές. Επιπλέον, προκειμένου να λάβουμε τα επιθυμητά αποτελέσματα σχεδόν από όλες τις API-embedded search engines, κάναμε χρήση ειδικών developer passwords / keys, τα οποία μας παρείχαν οι ιδιοκτήτες των αντίστοιχων μηχανών αναζήτησης. Τέλος, μικροπροβλήματα κατά την διαδικασία του result rendering οφείλονται στον επιτρεπόμενο από το HTML πρότυπο κακοφορμισμό ή στην ενδεχόμενη παραβίαση της αναμενόμενης μορφής της λίστας των αποτελεσμάτων.

Σε κάθε περίπτωση, η συνεισφορά της παρούσας διπλωματικής εργασίας τόσο στην συγκεκριμένη περιοχή μελέτης (των εφαρμογών χαρτογράφησης σκέψεων), όσο και στον γενικότερο χώρο των εργαλείων προώθησης της δημιουργικότητας του χρήστη, κρίνεται ως μοναδική (καθότι δεν υπάρχει κανένα άλλο ομοειδές εγχείρημα), καινοτόμα (αφού αναδεικνύει έναν δρόμο εργασίας, ο οποίος δεν έχει ακόμα εξερευνηθεί) και επιτυχής (καθώς στην πλειονότητα των περιπτώσεων το σύστημα λειτουργεί απρόσκοπτα, ενώ τα διάφορα ανακύψαντα εμπόδια είναι υπερβάσιμα).

7.2 Μελλοντικές επεκτάσεις

Η συγκεκριμένη ενότητα περιλαμβάνει ορισμένες ιδέες για την πραγμάτωση των σημαντικότερων επεκτάσεων, οι οποίες θα μπορούσαν να θεωρηθούν ως χρήσιμες για αυτή την διπλωματική εργασία.

- Ως πρώτη πρόταση, παραθέτουμε την επέκταση του υποσυστήματος του screen-scraper, με απώτερο σκοπό να αντιμετωπίζει όλες τις πιθανές περιπτώσεις, οι οποίες μπορούν να προκύψουν. Μέχρι στιγμής, έχει ληφθεί μέριμνα για το ενδεχόμενο όπου επαρκούν το δομικό μονοπάτι επί της HTML δενδρικής δομής, το πρώτο δεξιό closing tag (αντίστοιχο του τελευταίου element του εν λόγω path) ή το πρώτο δεξιό opening tag (αναλόγως πιο από τα δύο απαντάται πρώτο), οι πιθανώς απαραίτητες κανονικές εκφράσεις ή ακριβείς φράσεις αποκοπής (cut-off regular expressions / exact phrases) και ίσως ο λεγόμενος περιορισμός θέσης φύλλου (leaf position constraint). Για τα προαναφερθέντα στοιχεία, ο μελετητής μπορεί να διατρέξει προηγουμένως στο κείμενο, προκειμένου είτε να ξαναθυμηθεί τον ρόλο τους είτε να τα δει σε μεγαλύτερο βάθος, ενώ πρέπει να πούμε πάλι, ότι αυτά απαιτούνται για τον προσδιορισμό του καθενός από τα τρία υποστοιχεία που απαρτίζουν ένα αποτέλεσμα. Το σύστημά μας δεν διαχειρίζεται την περίπτωση που τα result subelements διαχωρίζονται με HTML style στοιχεία (όπως τα <style/>, <font/>) ή μέσω style attributes σε HTML structural elements (π.χ. ή). Μάλιστα το τελευταίο ενδεχόμενο χωρίζεται σε δύο υποπεριπτώσεις, στην πρώτη εκ των οποίων η τιμή του πεδίου δίνεται άμεσα (instant value), ενώ στην δεύτερη υπάρχει αναφορά για την χρησιμοποιούμενη τιμή μέσω συντόμευσης (value hyper-reference). Για να γίνουμε πιο σαφείς, χαρακτηριστικά δίνουμε το εξής παράδειγμα για την πρώτη περίπτωση: <a style = "*color : red*" >, ενώ για την δεύτερη το ακόλουθο: <tr class = "*title*">, όπου στην αρχή του HTML document έχει οριστεί: <style> tr.title {*bgcolor : blue* ;} </style>. Ο wrapper που θα λάμβανε υπόψη του και τα δεδομένα που παρουσιάσαμε μόλις τώρα, θα ήταν σχεδόν πλήρης, καθώς θα δύναται να αντιμετωπίσει το 90% και άνω των περιστατικών μηχανών αναζήτησης με καλοδομημένη επιστρεφόμενη σελίδα αποτελεσμάτων.
- Η δεύτερη πρότασή μας αφορά την επέκταση του συστήματος για την υποστήριξη και άλλων τύπων μηχανών αναζήτησης. Κινούμενοι σε αυτή την κατεύθυνση, η εφαρμογή μας θα μπορούσε να λειτουργεί και εκτός των πλαισίων των webpages ή των papers και να διαχειρίζεται παραδείγματος χάριν και blogs, images, audio ή video. Υπενθυμίζουμε στο σημείο αυτό, ότι οι εσωτερικές δομές έχουν κατασκευαστεί έτσι (σε όλη την έκταση του κώδικα), ούτως ώστε να είναι έτοιμες να ανταποκριθούν σε ένα τέτοιο εγχείρημα, προκειμένου όποιος ασχοληθεί με αυτό να μπορεί να επικεντρώσει τις προσπάθειές του μόνο στις περαιτέρω ενέργειες που απαιτούνται, δίχως να χρειάζεται να καταναλώσει χρόνο και πνευματικό κόπο για τετριμμένες εργασίες.
- Ως τρίτη πρόταση, δίνουμε την στήριξη των στοιχείων της προηγμένης λειτουργικότητας για τους ήδη υλοποιηθέντες τύπους δεδομένων ή για άλλους που ενδεχομένως έχουν μετέπειτα προστεθεί. Έτσι, θα μπορούσε κανείς να περιορίζει την αναζήτησή του και να

λαμβάνει πιο εξειδικευμένα αποτελέσματα, τα οποία θα προσέγγιζαν περισσότερο τις ανάγκες του. Επεξηγώντας, για paper searching επί παραδείγματι, να επιτρέπεται στον χρήστη να φιλτράρει τα επιστρεφόμενα αποτελέσματα βάσει του συγγραφέα (author), της ημερομηνίας (date), της θεματικής περιοχής (subject) ή του εκδότη (publisher).

Οι προηγουμένως παρουσιασθείσες προτάσεις συνιστούν μόνο κάποιες από τις γόνιμες σκέψεις, που φιλοδοξούν να επιτύχουν την ωφέλιμη επέκταση της εφαρμογής η οποία έχει ήδη κατασκευασθεί. Αυτό όμως, δεν σημαίνει ότι αποκλείουν την ανάληψη πρωτοβουλιών για την μετατόπιση του επίκεντρου των επιχειρούμενων προσθηκών, καθώς κάτι τέτοιο θεωρείται ότι αφήνεται στην έμπνευση, την δημιουργικότητα και την φαντασία του συνεχιστή.

" ..if impossible is unreachable, then why is there the sky?"

8

Βιβλιογραφία

Στο τελευταίο αυτό κεφάλαιο, παραθέτουμε την αξιοποιηθείσα, από την παρούσα διπλωματική εργασία, βιβλιογραφία, αναφορές στην οποία γίνονται καθ' όλη την έκταση του κειμένου. Ως πηγές χρησιμοποιήθηκαν επιστημονικές δημοσιεύσεις, βιβλία (τόσο σε έντυπη όσο και σε ηλεκτρονική μορφή), ιστοσελίδες πληροφόρησης και ιστότοποι με διαδραστικές υπηρεσίες εκμάθησης. Σημειώνουμε απλώς, ότι στα αριστερά δίνουμε την εκάστοτε βιβλιογραφική αναφορά, ενώ στα δεξιά παρέχουμε τα σχετικά στοιχεία.

[Shn07] B. Shneiderman, "*Creativity support tools: accelerating discovery and innovation*", ACM Community, vol. 50, no. 12, pp. 20-32, 2007.

[WP1] Wikipedia, The Free Encyclopedia – "*Mind Map*", < http://en.wikipedia.org/wiki/Mind_map >

[Gol01] J. Goldstein, "*Mind Mapping, Concept Mapping and Creativity*", FOCUS: Concept Development for Computer Animation, New York, USA, 2001.

[WP2] Wikipedia, The Free Encyclopedia – "*Concept Map*", < http://en.wikipedia.org/wiki/Concept_map >

[WP3] Wikipedia, The Free Encyclopedia – "*Ontology (Information Science)*", < http://en.wikipedia.org/wiki/Ontology_(information_science) >

[WP4] Wikipedia, The Free Encyclopedia – "*List of Mind-Mapping Software*", < http://en.wikipedia.org/wiki/List_of_mind_mapping_software >

[CKG+06] C.-H. Chang, M. Kayed, M. Girgis, K. Shaalan, "*A Survey of Web Information Extraction Systems*", IEEE Trans. Knowl. Data Eng., vol.18, no. 10, pp. 1411-1428, 2006.

[MMK99] I. Muslea, S. Minton, C. Knoblock, "*A Hierarchical Approach to Wrapper Induction*", Agents, pp. 190-197, 1999.

[SA99] A. Sahuguet, F. Azavant, "*Building Light-Weight Wrappers for Legacy Web Data-Sources Using W4F*", Proceedings of 25th International Conference on Very Large Data Bases, Edinburgh, Scotland, UK, 1999.

[HMK07] D. Huynh, S. Mazzocchi, D. Karger, "*Piggy Bank: Experience the Semantic Web inside your web browser*", J. Web Sem., vol. 5, no. 1, pp. 16-27, 2007.

[DOS03] M. Daconta, L. Obrst, K. Smith, "*The Semantic Web: a guide to the future of XML, Web services and knowledge management*", J. Whiley & Sons Inc. Publishing, New York, USA, 2003.

[BFG01] R. Baumgartner, S. Flesca, G. Gottlob, "*Supervised Wrapper Generation with Lixto*", Proceedings of 27th International Conference on Very Large Data Bases, Rome, Italy, 2001.

[AM01] J. Aslam, M. Montague, "*Models for Metasearch*", Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New Orleans, Louisiana, USA, 2001.

[WP5] Wikipedia, The Free Encyclopedia – "*List of Popular Search Engines*", < http://en.wikipedia.org/wiki/List_of_search_engines >

[LM06] A. Langville, C. Meyer, "*Google's PageRank and beyond: the science of search engine rankings*", Princeton University Press, New Jersey, USA, 2006.

[Ley02] M. Ley, "*The DBLP Computer Science Bibliography: Evolution, Research Issues, Perspectives*", String Processing and Information Retrieval, Springer Publications, Trier, Germany

[AD05] P. Ard, P. Dimple, "*Lucene Search Engine: An Overview*", DRTC-HP International Workshop on Building Digital Libraries, Bangalore, India, 2005.

[HG04] E. Hatcher, O. Gospodnetic, "*Lucene in Action*", Manning Publications Co., Greenwich, Connecticut, USA, 2004.

[Fre98] D. Freitag, "*Information Extraction from HTML: Application of a General Machine Learning Approach*", AAAI / IAAI, pp. 517-523, 1998.

[AGM03] A. Arasu, H. Garcia-Molina, "*Extracting Structured Data from Web Pages*", Proceedings of the 19th International Conference on Data Engineering, Bangalore, India, 2003.

[WP6] Wikipedia, The Free Encyclopedia – "*DOM* ", < http://en.wikipedia.org/wiki/Document_Object_Model >

[WP7] Wikipedia, The Free Encyclopedia – "*SAX*", < http://en.wikipedia.org/wiki/Simple_API_for_XML >

[WP8] Wikipedia, The Free Encyclopedia – "*Depth First Search*", < http://en.wikipedia.org/wiki/Depth-first_search >

[GA05] K. Golub, A. Ardo, "*Importance of HTML structural elements and metadata in automated subject classification*", Research and Advanced Technology for Digital Libraries, Springer Publications, Lund, Sweden, 2005.

[WP9] Wikipedia, The Free Encyclopedia – "*Regular Expression (Computing)*", < http://en.wikipedia.org/wiki/Regular_expression >

[Dar01] I. Darwin, "*Java Cookbook*", O' Reilly Media Inc., Sebastopol, California, USA, 2001.

[Fla05] D. Flanagan, "*Java in a Nutshell*", O' Reilly Media Inc., Koeln, Germany, 2005.

[Hol04] S. Holzner, "*Eclipse: A Java Developer's Guide*", O' Reilly Media Inc., Cambridge, England, UK, 2004.

[GBM03] D. Gallardo, E. Burnette, R. McGovern, "*Eclipse in Action: A guide for Java developers*", Manning Publications Co., Greenwich, Connecticut, USA, 2003.

[RF05] E. Robson, E. Freeman, "*Head first, HTML with CSS & XHTML*", O' Reilly Media Inc., Sebastopol, California, USA, 2005.

[W3SCH1] W3Schools, Full Web Building – Developer Site, "*Learning HTML*", < http://www.w3schools.com/html/default.asp >

[W3SCH2] W3Schools, Full Web Building – Developer Site, "*Learning HTML Styles:*

CSS ", < http://www.w3schools.com/html/html_css.asp >

[W3SCH3] W3Schools, Full Web Building – Developer Site, "*Learning XML*", < http://www.w3schools.com/xml/default.asp >

[W3SCH4] W3Schools, Full Web Building – Developer Site, "*Learning DTD*", < http://www.w3schools.com/dtd/default.asp >

[W3SCH5] W3Schools, Full Web Building – Developer Site, "*Learning JavaScript*", < http://www.w3schools.com/js/default.asp >