ΤΕΙ ΘΕΣΣΑΛΟΝΙΚΗΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Faceted search και εφαρμογές στην ανάπτυξη ιστοσελίδων ηλεκτρονικού εμπορίου

Πτυχιακή Εργασία
Χαράλαμπος Φερσιζίδης

Εισηγητής Καθηγητής
Μιχαήλ Σαλαμπάσης

Θεσσαλονίκη, Νοέμβριος 2011

Περιεχόμενα

Περιεχόμενα εικόνων

1. Εισαγωγή: Η ανάγκη για faceted search

1.1. Το πρόβλημα

Ζούμε σε μια εποχή που η ποσότητα της πληροφορίας, προερχόμενη συνήθως από βιβλία, ραδιόφωνα, εφημερίδες, ΜΜΕ, διαδίκτυο κλπ, έχει γιγαντωθεί τόσο πολύ που κρίνεται πλέον αναγκαίο να βελτιστοποιηθούν και τα εργαλεία αναζήτησης. Οι γνωστές σε όλους μηχανές αναζήτησης αναλαμβάνουν αυτό το έργο εδώ και πολλά χρόνια φέρνοντας εις πέρας καθημερινά αμέτρητες αιτήσεις χρηστών. Ακόμα και αυτές οι μηχανές όμως φαίνεται πως λυγίζουν μπροστά στον όγκο της πληροφορίας που έχουν να επεξεργαστούν. Αυτό φαίνεται από το γεγονός ότι πολλές φορές χρειάζεται επιπρόσθετη προσπάθεια από τον χρήστη για να βρει αυτό που πραγματικά ψάχνει. Δεν αρκεί δηλαδή απλά να ξέρει τι αναζητεί, αλλά πρέπει να σκεφτεί έναν συνδυασμό λέξεων που θα εισάγει στην μηχανή αναζήτησης ώστε να έχει τα επιθυμητά αποτελέσματα.

Το πρόβλημα του όγκου της πληροφορίας δεν εμφανίζεται μόνο στις γνωστές μηχανές αναζήτησης, (Google, Yahoo κλπ.) αλλά εντοπίζεται και στην αναζήτηση όπως αυτή υλοποιείται σε διάφορες διαδικτυακές εφαρμογές. Εκεί τα πράγματα ορισμένες φορές είναι ακόμα χειρότερα καθώς τα αποτελέσματα δεν ικανοποιούν και είναι πραγματικά πιθανό να μην τις χρησιμοποιεί κανείς, αντιθέτως οι χρήστες προτιμούν να πλοηγηθούν μέσω των μενού.

1.2. Η λύση στο πρόσωπο του faceted search

Κάπου εδώ έρχεται να συνεισφέρει το faceted search. Όπως θα δούμε και στο αντίστοιχο κεφάλαιο της πτυχιακής, το faceted search σαν σύλληψη έχει τις ρίζες του στα χρόνια του Αριστοτέλη ο οποίος με το γνωστό «δέντρο» του ταξινόμησε τα ζωντανά όντα. Η λέξη κλειδί εδώ είναι «ταξινόμηση». Από οποιαδήποτε σκοπιά και να το δείτε η λέξη αυτή προϊδεάζει για οργάνωση. Με τέτοιο όγκο πληροφορίας σήμερα η οργάνωση που μπορεί να προϋπάρχει ή να δημιουργήσουμε σε μία συλλογή πληροφοριών, μπορεί να παίξει σημαντικό ρόλο στη βελτίωση των μεθόδων αναζήτησης.

Στη μετέπειτα πορεία διάφορες υλοποιήσεις λειτούργησαν σαν προπομπός για τη μορφή που έχει το faceted search σήμερα. Μερικές από αυτές είναι το Dewey Decimal Classification του Melvil Dewey, τα Ranganathan’s Colons, το Flamenco Project κ.α. , οι οποίες με τη σειρά τους η κάθε μια προσέγγιζε το faceted search με τον δικό τους τρόπο ώσπου και αναπτύχθηκαν εφαρμογές που υλοποιούν 100% την φιλοσοφία του.

Μιλώντας για faceted search θα αναφερθούμε σε έννοιες άρρηκτα συνδεδεμένες με αυτό, όπως information retrieval, entity extraction, taxonomy, ontologies κλπ. Πρόκειται ουσιαστικά για έννοιες που αναφέρονται στη διαχείριση και ανάκτηση

της πληροφορίας. Αυτό που κάνουν πρακτικά είναι να υλοποιούν τεχνικές στις οποίες βασίζεται το faceted search για να φιλτράρει και να ταξινομήσει την πληροφορία. Με λίγα λόγια, για να φτάσουμε να υλοποιήσουμε μία αναζήτηση βασισμένη στο faceted search πρέπει να έχει προηγηθεί ένας μεγάλος όγκος προεργασίας. Θα δούμε στη συνέχεια ότι στη σχεδίαση ενός τέτοιου συστήματος δεν αρκεί μόνο να φροντίσουμε το πώς θα φαίνεται στον χρήστη αλλά η μεγαλύτερη βαρύτητα πρέπει να δοθεί στον κώδικα που τρέχει από πίσω. Ειδικά στο δεύτερο, επικρατεί μεγάλος πονοκέφαλος στις τάξεις των προγραμματιστών. Όπως σε όλες τις εφαρμογές πληροφορικής, ένα καλοστημένο UI με αποδοτική λειτουργικότητα είναι το κλειδί για μια επιτυχημένη εφαρμογή και οι faceted search εφαρμογές δεν αποτελούν εξαίρεση.

1.3. Το faceted search σήμερα

Μέσα στην τελευταία δεκαετία και φτάνοντας στο σήμερα βλέπουμε την ολοένα και αυξανόμενη δημοτικότητα του faceted search. Η μια ιστοσελίδα μετά την άλλη το ενσωματώνει ως βασική λειτουργία, κάνοντας τη ζωή των χρηστών πολύ ευκολότερη, συμπληρώνοντας σε μεγάλο βαθμό άλλες μορφές αναζήτησης (π.χ. με λέξεις κλειδιά). Ειδικότερα οι εφαρμογές ηλεκτρονικού εμπορίου τείνουν να το έχουν σαν προτεραιότητα στην ανάπτυξη μιας αποτελεσματικής αναζήτησης με facets.

Λίγα λεπτά ενασχόλησης με μια τέτοια εφαρμογή και γρήγορα καταλαβαίνει κανείς τις πολλαπλές δυνατότητες που προσφέρει το faceted search. Δε θα λέγαμε ότι εξαλείφει όλα τα μειονεκτήματα του information overload αλλά φτάνει σε ένα πάρα πολύ ικανοποιητικό σημείο καθώς προσφέρει και μια μορφή καθοδήγησης. Τα facets λειτουργούν σαν οδηγοί προς το αντικείμενο αναζήτησης. Συνδυάζοντας την τεχνική του keyword search και των facets έχουμε ένα πολύ αποδοτικό μηχανισμό που μας παρέχει αρκετά μεγάλη ακρίβεια στα αποτελέσματά μας. Εδώ θα αναφέρουμε για πρώτη φορά τον όρο σχετικότητα που θα αναλύσουμε διεξοδικά στη συνέχεια. Η ανάγκη για faceted search προήλθε και από την επιθυμία να έχουμε όσο το δυνατόν πιο σχετικά αποτελέσματα με το ερώτημα μας. Δεν είναι λίγες οι φορές που στην αναζήτηση ενός αντικειμένου δυσανασχετούμε στην θέα άσχετων εγγράφων στα αποτελέσματα. Το faceted search κάνει προσπάθειες στην εξάλειψη και αυτού του μειονεκτήματος των συμβατικών μηχανών, προσφέροντας συνδυασμούς τεχνικών που παντρεύουν την ανάκληση(recall) και την ακρίβεια (precision).

Συνοψίζοντας, η ανάγκη για αποδοτικότερη αναζήτηση μέσα από πληθώρα πληροφορίας γέννησε το faceted search. Ίσως να μην είναι η δημοφιλέστερη μέθοδος σήμερα αλλά το γεγονός και μόνο ότι η Google κινείται σιγά σιγά και προς αυτή την κατεύθυνση, νομίζω διατυπώνει με τον καλύτερο τρόπο την αποδοτικότητα και τις προοπτικές του σαν τεχνολογία.

2. Web search

2.1. Εισαγωγή

Η ιδέα της αναζήτησης πληροφοριών στο διαδίκτυο γεννήθηκε σχεδόν παράλληλα με το ίδιο το διαδίκτυο. Ο ολοένα και αυξανόμενος όγκος πληροφοριών κατέστησε αναγκαίο, από ένα σημείο και μετά, τη χρήση αυτών που σήμερα είναι ευρέως γνωστές ως μηχανές αναζήτησης. Σε αυτήν την πτυχιακή δε θα μας απασχολήσει τόσο η έννοια και η λειτουργία των μηχανών όσο η φιλοσοφία της αναζήτησης αυτής καθαυτής καθώς και οι διάφορες μέθοδοι αναζήτησης.

2.2. Γενικά

Ας πούμε λίγα λόγια για τις μηχανές αναζήτησης. Αν θέλαμε να δώσουμε έναν ορισμό ίσως να ήταν ο εξής:

*Μια **μηχανή αναζήτησης** είναι μια εφαρμογή που επιτρέπει την αναζήτηση κειμένων και αρχείων στο Διαδίκτυο. Αποτελείται από ένα πρόγραμμα που βρίσκεται σε έναν ή περισσότερους υπολογιστές στους οποίους δημιουργεί μια βάση δεδομένων, με τις πληροφορίες που συλλέγει από το διαδίκτυο, και το διαδραστικό περιβάλλον που εμφανίζεται στον τελικό χρήστη ο οποίος χρησιμοποιεί την εφαρμογή από άλλον υπολογιστή συνδεμένο στο διαδίκτυο.*

Η επιτυχία που έχουν οι μηχανές αναζήτησης οφείλεται στο ότι οργανώνουν το περιεχόμενο του διαδικτύου και έτσι το κάνουν προσιτό στους χρήστες. Επίσης, δίνουν την δυνατότητα να ψάξει κανείς στη βάση δεδομένων τους με διάφορα επιμέρους εργαλεία που κάνουν τις αναζητήσεις περισσότερο εξειδικευμένες. Για παράδειγμα, μια μηχανή μπορεί να αναζητήσει αποτελέσματα στις περισσότερες γλώσσες του κόσμου ή να τα εξάγει μόνο από συγκεκριμένη χώρα. Οι μηχανές αναζήτησης μας δίνουν και πληροφορίες για τις περισσότερες ιστοσελίδες που υπάρχουν στο διαδίκτυο, όπως για παράδειγμα πόσες ιστοσελίδες από το συγκεκριμένο website περιλαμβάνουν στην βάση τους ή ακόμη και ειδικές βαθμολογίες (RANK) για ιστοσελίδες.

Οι χρήστες του διαδικτύου προτιμούν περισσότερο τις μηχανές αναζήτησης που εξάγουν όσο το δυνατόν σχετικότερα με τις αναζητήσεις αποτελέσματα με ένα φυσικό τρόπο και όχι να προβάλουν συνέχεια διαφημιστικά μηνύματα στις πρώτες θέσεις των αποτελεσμάτων. Γι' αυτό βασικά τον λόγο αλλάζουν οι προτιμήσεις των χρηστών του διαδικτύου και μια μηχανή αναζήτησης που μπορεί να ήταν πολύ δημοφιλής για κάποια χρόνια χάνει ξαφνικά την δημοτικότητα της και περνάει στην λήθη.

2.3. Ιστορικό

Στα πρώτα στάδια ανάπτυξης του διαδικτύου υπήρχε μια λίστα από web servers, η οποία φτιάχτηκε από τον Tim Berners-Lee, που βρισκόταν στον server CERN. Η λίστα μεγάλωνε μέρα με τη μέρα καθώς όλο και περισσότεροι web servers συνδέονταν

στο διαδίκτυο σε σημείο που η συντήρηση και ο έλεγχος τους δυσκόλευε. Στην ιστοσελίδα του NCSA (National Center for Supercomputing Applications) αυτοί οι καινούριοι web servers τιτλοφορούνταν ως “What’s New!”.

Το πρώτο εργαλείο που χρησιμοποιήθηκε ποτέ για αναζήτηση στο διαδίκτυο ήταν το Archie. Το όνομα του προερχόταν από την αγγλική λέξη “Archive”, χωρίς το “v”. Δημιουργήθηκε το 1990 από τους Alan Emtage, Bill Heelan και J. Peter Deutsch, σπουδαστές Πληροφορικής στο McGill University του Montreal. Το πρόγραμμα κατέβαζε λίστες φακέλων από όλα τα αρχεία που βρισκόταν σε websites, τα οποία χρησιμοποιούσαν anonymous FTP πρωτόκολλο, δημιουργώντας μια βάση δεδομένων προς αναζήτηση σύμφωνα με τα ονόματα των αρχείων. Παρόλα αυτά το Archie δεν ταξινομούσε τα περιεχόμενα αυτών των πινάκων γιατί ο όγκος των δεδομένων ήταν πολύ περιορισμένος.

Η εμφάνιση του πρωτοκόλλου Gopher (το οποίο δημιουργήθηκε το 1991 από τον Mark McCahill στο Πανεπιστήμιο της Minnesota) οδήγησε στη δημιουργία δύο νέων προγραμμάτων, τα Veronica και Jughead. Όπως το Archie, αναζητούσαν ονόματα αρχείων και τίτλων που ήταν αποθηκευμένα στις βάσεις δεδομένων του Gopher. Το Veronica παρείχε αναζήτηση με βάση μια λέξη κλειδί ενώ το Jughead ήταν περισσότερο εργαλείο ανάκτησης πληροφοριών διαδικτυακών μενού. Αυτά τα μενού υπήρχαν σε συγκεκριμένους servers που χρησιμοποιούσαν το πρωτόκολλο Gopher.

Το καλοκαίρι του 1993, δεν υπήρχε καμία μηχανή αναζήτησης για το διαδίκτυο. Υπήρχαν μόνο κατάλογοι πληροφοριών που όμως συντηρούνταν με το χέρι. Ο Oscar Nierstrasz από το Πανεπιστήμιο της Γενεύης έγραψε ορισμένα script σε γλώσσα Perl που θα αντέγραφαν επακριβώς τους καταλόγους και θα τους μετέτρεπαν σε μια μορφή που αποτέλεσε τη βάση του W3Catalog, την πρώτη πρώιμη μηχανή αναζήτησης του διαδικτύου, η ύπαρξη της οποίας ανακοινώθηκε στις 2 Σεπτεμβρίου 1993.

Τον Ιούνιο του 1993, ο Matthew Gray, στο πανεπιστήμιο MIT, δημιούργησε κάτι που πιθανώς να ήταν το πρώτο we-robot το WWW Wanderer. Γραμμένο σε γλώσσα Perl το Wanderer χρησιμοποιούσε ευρετήριο που ονομαζόταν Wandex. Ο σκοπός του ήταν να μετρήσει το μέγεθος του διαδικτύου, το οποίο και το έκανε έως το 1995. Η εμφάνιση της 2^{ης} μηχανής αναζήτησης έγινε τον Νοέμβριο του 1993, με την ονομασία Aliweb. Το Aliweb δε χρησιμοποιούσε web-robot, αντιθέτως ενημερωνόταν από τους administrators για την ύπαρξη της εκάστοτε ιστοσελίδας.

Το JumpStation (Δεκέμβριος του 1993) χρησιμοποιούσε ένα web-robot για να ψάχνει ιστοσελίδες και να δομεί το ευρετήριο του ενώ χρησιμοποιούσε και μια φόρμα για το λογισμικό αιτήσεων. Επρόκειτο για το πρώτο εργαλείο αναζήτησης που υλοποιούσε τα 3 απαραίτητα χαρακτηριστικά μια μηχανής (crawling, indexing, searching). Δυστυχώς οι πόροι που είχε διαθέσιμους η συγκεκριμένη υλοποίηση

ήταν περιορισμένοι, οπότε αρκούνταν στο να αποθηκεύει τίτλους και επικεφαλίδες των ιστοσελίδων.

Μια από τις πρώτες "full-text crawler-based" μηχανές αναζήτησης ήταν το WebCrawler, το 1994. Αντίθετα με τους προγόνους του, έδινε τη δυνατότητα στους χρήστες να ψάχνουν για οποιαδήποτε λέξη σε οποιαδήποτε ιστοσελίδα. Η συγκεκριμένη λογική ήταν δεδομένη για κάθε επόμενη μηχανή αναζήτησης. Επίσης ήταν η πρώτη που έγινε ευρύτερα γνωστή στο κοινό. Επιπλέον το 1994, ξεκίνησε τη λειτουργία του το Lycos το οποίο μάλιστα σημείωσε μεγάλη επιτυχία.

Αργότερα, πολλές μηχανές αναζήτησης εμφανιστήκαν αναζητώντας δημοσιότητα. Ορισμένες από αυτές ήταν οι Magellan, Excite, Infoseek, Inktomi, Northern Light, Yahoo και AltaVista. Η Yahoo ήταν ανάμεσα στους πιο δημοφιλής τρόπους για αναζήτηση ιστοσελίδων από τους χρήστες. Δε χρησιμοποιούσε full-text αντίγραφα των ιστοσελίδων αλλά διατηρούσε δικό της κατάλογο.

Το 1996, το Netscape έψαχνε να ενσωματώσει στον browser της μια αποκλειστική μηχανή αναζήτησης. Το ενδιαφέρον όμως ήταν τόσο μεγάλο που τελικά έγινε συμφωνία με πέντε διαφορετικές μηχανές, Yahoo, Magellan, Lycos, Infoseek και Excite.

Γύρω στο 2000, η μηχανή αναζήτησης Google έκανε την εμφάνιση της. Τα αποτελέσματα της συγκεκριμένης μηχανής ήταν πολύ καλύτερα λόγω της χρήσης μιας καινοτομίας που ονομάστηκε PageRank. Η λογική της καινοτομίας αυτής ήταν να κατατάσσει τις ιστοσελίδες σύμφωνα με το PageRank τους οπότε να εμφανίζει στα αποτελέσματα αυτές με το μεγαλύτερο, που κατ' επέκταση ήταν και οι πιο δημοφιλείς. Η κατασκευή της χρησιμοποιούσε ένα λιτό interface σε αντίθεση με τις υπόλοιπες μηχανές που ήταν ενσωματωμένες σε κάποιο portal.

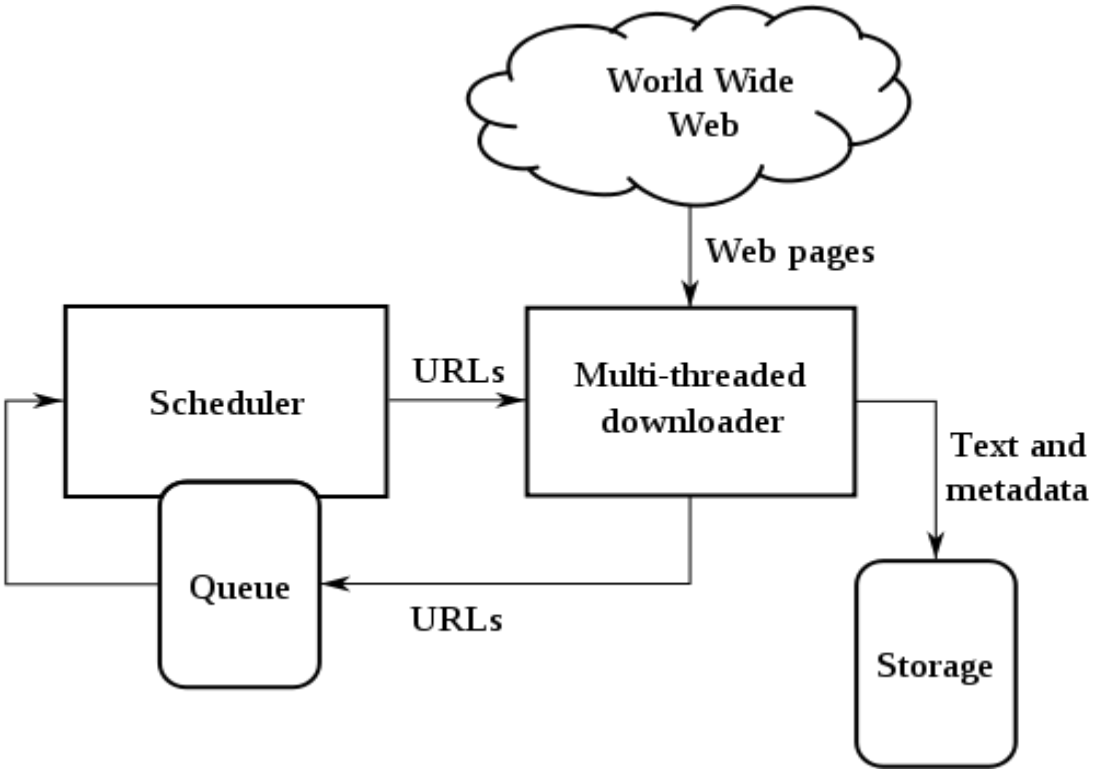
Η Microsoft πρωτοπαρουσίασε την μηχανή αναζήτησης MSN το 1998, η οποία όμως χρησιμοποιούσε τα αποτελέσματα της Inktomi. Το 1999 υπήρχε μια μίξη αποτελεσμάτων από τις μηχανές Inktomi και Looksmart ενώ στα τέλη του έτους χρησιμοποιούταν τα αποτελέσματα από το Altavista. Το 2004, η Microsoft ξεκίνησε την κατασκευή μιας δικής της μηχανής η οποία θα υλοποιούσε ένα δικό της web crawler, το msnbot. Τέλος τον Ιούλιο του 2009 έκανε την εμφάνιση της η μηχανή αναζήτησης Bing.

2.4. Λειτουργία των μηχανών αναζήτησης

Μια μηχανή αναζήτησης εκτελεί λειτουργίες με την εξής σειρά:

- Web Crawling
- Indexing

- Searching



Εικόνα 1 - Δομή ενός Web Crawler

Η λειτουργία τους βασίζεται στην αποθήκευση δεδομένων από πληροφορίες που αντλούν από τα αρχεία HTML. Αυτές οι σελίδες ανακτούνται από έναν Web Crawler ο οποίος ακολουθεί κάθε σύνδεσμο της ιστοσελίδας. Στη συνέχεια τα περιεχόμενα της ιστοσελίδας αναλύονται και αποφασίζεται από το λογισμικό πως θα ταξινομηθούν (λέξεις που εντοπίζονται σε τίτλους, επικεφαλίδες και ειδικά πεδία που ονομάζονται meta tags). Τα δεδομένα σχετικά με την ιστοσελίδα αυτή καθαυτή αποθηκεύονται σε μια βάση δεδομένων για μετέπειτα αιτήσεις(ερωτήματα). Ένα ερώτημα μπορεί να είναι και μία λέξη μόνη της. Ο σκοπός λοιπόν του ευρετηρίου είναι να επιτρέπει την πληροφορία να εντοπίζεται όσο το δυνατόν γρηγορότερα. Μερικές μηχανές αναζήτησης, όπως το Google, αποθηκεύουν ολόκληρο ή μέρος του πηγαίου κώδικα (cache) όπως και πληροφορίες για τις ιστοσελίδες, ενώ άλλες όπως το Altavista, αποθηκεύουν την κάθε λέξη της ιστοσελίδας που θα βρουν.

Όταν ένας χρήστης εισάγει ένα ερώτημα (συνήθως χρησιμοποιώντας λέξεις κλειδιά), η μηχανή ελέγχει το ευρετήριο της για να βρει την ιστοσελίδα που αντιστοιχεί όσο πιστότερα γίνεται σε αυτά τα κριτήρια. Σαν αποτέλεσμα παρουσιάζει τον τίτλο και μέρος του κειμένου της ιστοσελίδας. Δυστυχώς δεν υπάρχουν ακόμα μηχανές που να κάνουν ταξινόμηση βάσει της ημερομηνίας.

Οι περισσότερες μηχανές αναζήτησης κάνουν χρήση boolean operators όπως το AND,OR και NOT για να επεκτείνουν ένα ερώτημα. Ο χρήστης τους χρησιμοποιεί όταν θέλει να κάνει πιο συγκεκριμένη η πιο στοχευόμενη την αναζήτηση του. Η

μηχανή αναζητεί τις λέξεις ή φράσεις ακριβώς όπως δόθηκαν. Μερικές μηχανές αναζήτησης διαθέτουν την ιδιότητα proximity search που παρέχει στους χρήστες τη δυνατότητα να ορίσουν την απόσταση μεταξύ δύο λέξεων-κλειδιών. Επίσης η αναζήτηση γίνεται με βάση στατιστικές μετρήσεις όπως επίσης και για το βαθμό σχετικότητας της λέξης με το περιεχόμενο της εκάστοτε ιστοσελίδας. Τέλος μια άλλη παραλλαγή είναι το ask.com όπου οι χρήστες εισάγουν τα ερωτήματα σαν να απευθύνονται σε ανθρώπους.

Η χρησιμότητα της μηχανής αναζήτησης εξαρτάται από τη σχετικότητα των αποτελεσμάτων που εξάγει. Μπορεί μια λέξη να υπάρχει σε εκατομμύρια ιστοσελίδες αλλά μερικές από αυτές είναι πιο σχετικές με την λέξη κλειδί. Για παράδειγμα η λέξη κλειδί «στοίχημα» μπορεί να υπάρχει οπουδήποτε αλλά το πιο πιθανό είναι να συσχετιστεί με την ιστοσελίδα του στοιχήματος του ΟΠΑΠ. Οι περισσότερες μηχανές αναζήτησης λοιπόν χρησιμοποιούν μεθόδους για να ταξινομήσουν τα αποτελέσματα ώστε αρχικά να εμφανίζονται τα πιο σχετικά και δημοφιλή. Η μέθοδος που χρησιμοποιεί μια μηχανή αναζήτησης για να ταξινομεί τα αποτελέσματα ποικίλει ανάλογα την μηχανή. Επίσης σε βάθος χρόνου αλλάζουν και οι μέθοδοι αναζήτησης αυτοί καθαυτοί λόγω της εξέλιξης του διαδικτύου.

2.5. Keyword search (Αναζήτηση με λέξη-κλειδί)

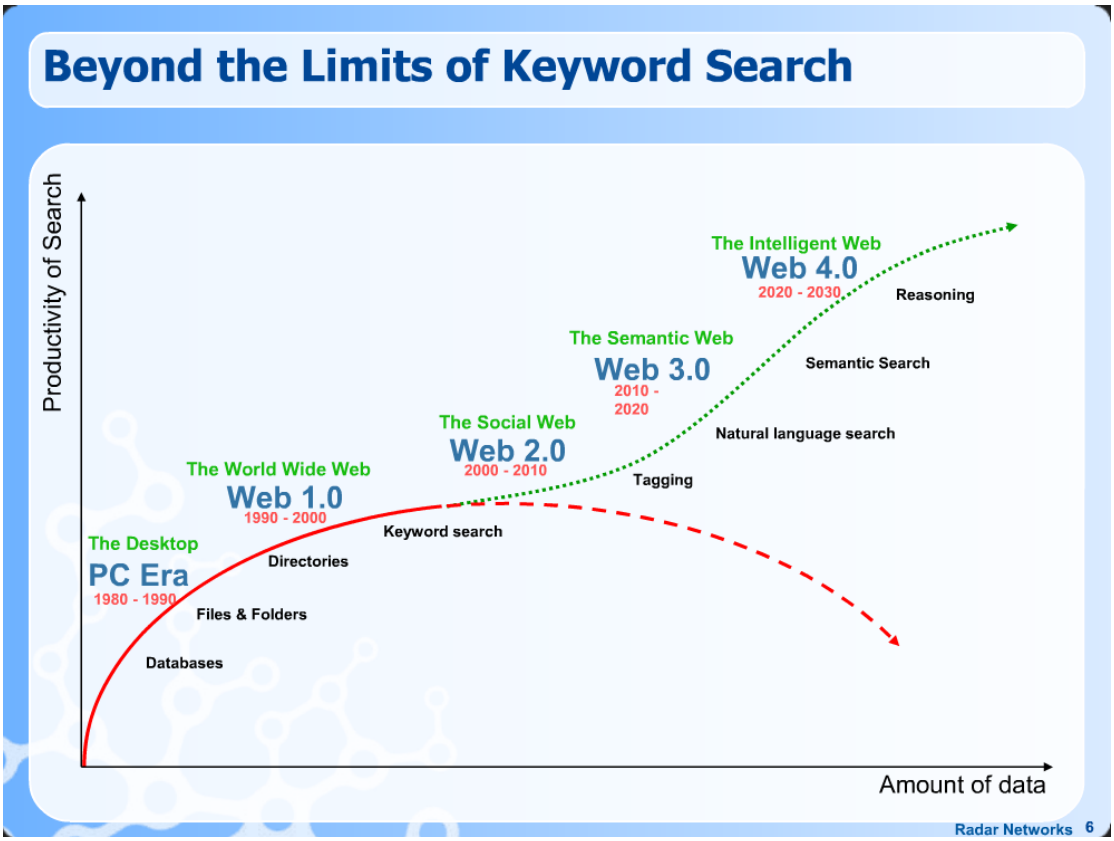
Πρόκειται για μια τεχνική που χρησιμοποιείται από τους χρήστες προκειμένου να αναζητήσουν το περιεχόμενο της αρεσκείας τους στο διαδίκτυο. Είναι η πιο δημοφιλής τεχνική που βασίζεται στη λογική του «Ψάχνω κάτι, άρα το γράφω στη μηχανή αναζήτησης για να μου το βρει». Όπως αναφέρθηκε και πιο πάνω μεγάλο ρόλο παίζει η σχετικότητα της εκάστοτε σελίδας με την λέξη που εισάγουμε. Το ιδανικό είναι να εισάγουμε δύο με τρεις λέξεις κλειδιά, οι οποίες να είναι όσο πιο στοχευμένες γίνεται, προκειμένου να λάβουμε τα καλύτερα δυνατά αποτελέσματα. Οι μηχανές αναζήτησης λοιπόν διαβάζουν τις λέξεις που τους δώσαμε και ανάλογα την σχετικότητα, τη σαφήνεια, την δημοτικότητα των λέξεων και των ιστοσελίδων που αντιστοιχούν, εξάγουν αποτελέσματα. Αξίζει να αναφέρουμε εδώ το γεγονός ότι κάθε ιστοσελίδα στον κώδικα της έχει ενσωματωμένες λέξεις κλειδιά που εξυπηρετούν ακριβώς αυτό το σκοπό, να βρίσκει εύκολα ο χρήστης αυτό που αναζητεί χωρίς απαραίτητα να χρησιμοποιήσει πολύ συγκεκριμένες λέξεις. Για παράδειγμα όλες οι εταιρίες πληροφορικής θα έχουν ενσωματώσει λέξεις όπως service ή web οπότε η πιο δημοφιλής ιστοσελίδα που χρησιμοποιεί αυτές τις λέξεις θα εμφανίζεται ψηλότερα στις αναζητήσεις για ”hosting” ή “Service”. Ο χρήστης πληκτρολογώντας τη λέξη hosting μπορεί να ψάχνει για φιλοξενία σε ξενοδοχείο αλλά εν τέλει να πάρει στα αποτελέσματα του «φιλοξενία» σε server.

Με το πέρασμα του χρόνου, και όσο αυξάνεται η πληροφορία στο διαδίκτυο, θα γίνεται ολοένα και δυσκολότερο να πετύχουμε αποδοτικά αποτελέσματα

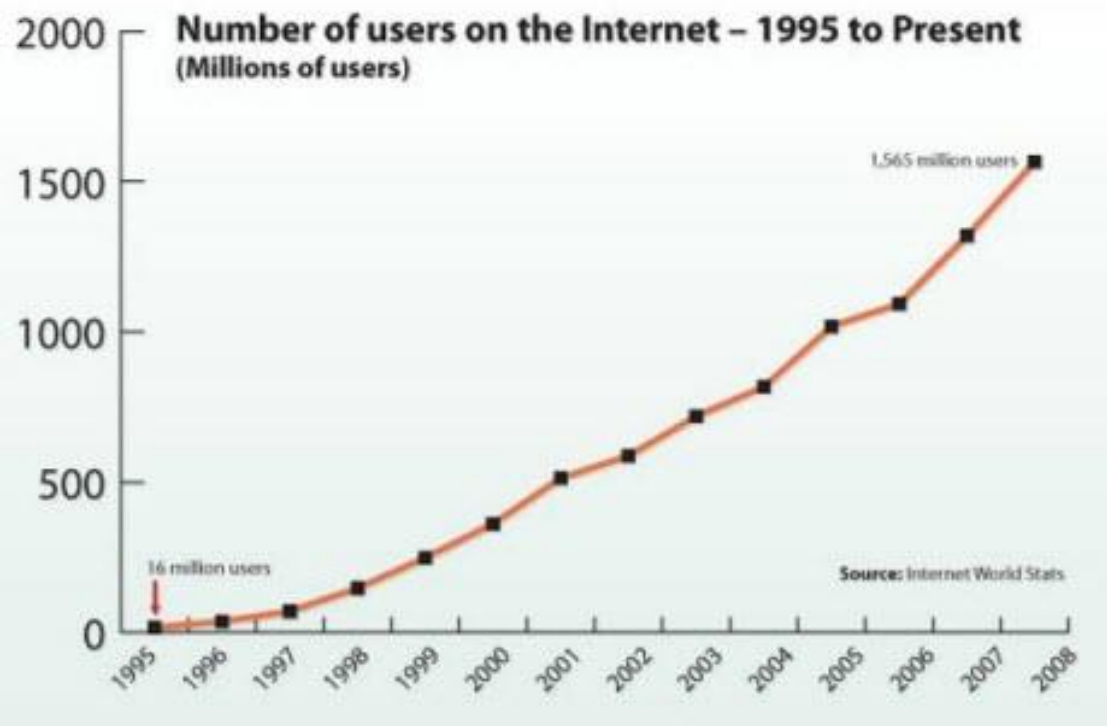
χρησιμοποιώντας αναζήτηση με λέξεις κλειδιά. Ήδη εν έτη 2011 τα αποτελέσματα είναι μερικές φορές αμφιλεγόμενα. Παρατηρείται μάλιστα ότι ακόμα και ένα χρόνο πριν ήταν ευκολότερο να βρεις κάτι π.χ. στο Google από ότι φέτος.

Σαν τεχνική δεν τα πηγαίνει άσχημα φυσικά ακόμα και σήμερα αλλά με την ραγδαία διόγκωση των δεδομένων που λαμβάνει τόπο κάθε στιγμή, θα χρειαστούμε κάτι καλύτερο για το μέλλον.

Το διαδίκτυο αριθμεί 1.3 δισεκατομμύρια ανθρώπους online ανά πάσα στιγμή και γύρω στις 100 εκατομμύρια ενεργές ιστοσελίδες. Όσο θα αυξάνονται αυτοί οι αριθμοί τόσο περισσότερο θα μειώνεται η αποτελεσματικότητα της εν λόγω τεχνικής. Θα είναι πρακτικά σαν να ψάχνουμε βελόνα στα άχυρα, μόνο που τα άχυρα θα αυξάνονται ολοένα και περισσότερο κάθε μέρα.



Εικόνα 2 - Οδεύοντας προς το Semantic Web



Εικόνα 3 - Χρήστες του διαδικτύου

2.6. Semantic Search

Μια πρόσφατη τεχνική, που προσπαθεί να βελτιώσει την ακρίβεια των αναζητήσεων και των αποτελεσμάτων, είναι το Semantic search. Η υλοποίηση αυτή προσπαθεί να καταλάβει τις προθέσεις του χρήστη καθώς και το νόημα των λέξεων που χρησιμοποιεί στην αναζήτηση του ώστε να εξάγει πιο σχετικά αποτελέσματα.

Αντί για τη χρήση του PageRank (που χρησιμοποιεί το Google) το Semantic search κάνει χρήση των semantics. Πρόκειται ουσιαστικά για μια προσπάθεια να γίνει κατανοητό το νόημα και η σημασία των λέξεων ώστε να επιτευχθεί μεγαλύτερη σχετικότητα στα αποτελέσματα.

2.7. Semantic Search και αποσαφήνιση

Για να καταστεί κατανοητό το τι ψάχνει ο χρήστης θα χρειαστεί αποσαφήνιση των λέξεων-κλειδιών που χρησιμοποίησε. Όταν μια λέξη είναι ασαφής μπορεί να έχει πολλά διαφορετικά νοήματα. Για παράδειγμα η λέξη «ασφάλεια» μπορεί να σημαίνει το να μη διατρέχεις κίνδυνο ή και την ασφάλεια ενός καταστήματος (φύλακες, συναγερμούς). Σε αυτήν την περίπτωση ξεκινάει η διαδικασία της αποσαφήνισης όπου το πιο πιθανό νόημα επιλέγεται σαν κριτήριο για τα αποτελέσματα. Αυτή η διαδικασία χρησιμοποιεί πληροφορία που είναι αποθηκευμένη στη βάση δεδομένων και λαμβάνει υπόψη τα νοήματα και από τις υπόλοιπες λέξεις που υπάρχουν στην πρόταση ή το υπόλοιπο κείμενο.

2.8. Faceted search

Πρόκειται για την περίπτωση που θα αναλύσουμε σε αυτή την πτυχιακή. Το faceted search ή αλλιώς faceted navigation ή faceted browsing είναι μια τεχνική πρόσβασης σε συλλογές πληροφοριών που ταξινομούνται με facets και επιτρέπει τους χρήστες να φιλτράρουν την διαθέσιμη πληροφορία. Η ταξινόμηση κατά facets επιτρέπει τον ορισμό πολλαπλών facets ανά αντικείμενο επιτρέποντας έτσι την ύπαρξη πολλαπλών επιλογών αναζήτησης αντί του μονόδρομου που προσφέρει το taxonomic σύστημα. Για παράδειγμα ένα άρθρο για τους υπολογιστές μπορεί να βρεθεί κάτω από το facet “informatics” ή ακόμα και από το “technology”.

Τα facets συνήθως παράγονται από την ανάλυση των κειμένων χρησιμοποιώντας τη μέθοδο του entity extraction ή χρησιμοποιώντας και ήδη υπάρχοντα πεδία στη βάση δεδομένων, όπως author, descriptions, language, format. Αυτή η προσέγγιση επιτρέπει στις υπάρχουσες ιστοσελίδες, τις περιγραφές προϊόντων ή στα διάφορα άρθρα να έχουν επιπλέον metadata δεδομένα και να μπορούν να αναζητούνται μέσω ενός facet.

Το faceted search έχει γίνει ένα σημαντικό χαρακτηριστικό για τις μηχανές αναζήτησης που βελτιώνει σημαντικά την αποτελεσματικότητα της και βοηθάει τον χρήστη να αναζητήσει ευκολότερα ότι ψάχνει.

Εικόνα 4 - Faceted Search στην πράξη

Η παραπάνω εικόνα αναφέρεται σε faceted browsing και όχι σε κάποια αναζήτηση χρήστη. Έχουμε το facet “Manufacturer” το οποίο έχει κάποιες τιμές για επιπρόσθετο φιλτράρισμα. Κάνοντας κλικ στο “Sony” θα έχουμε στα αποτελέσματα όσες κάμερες είναι Sony. Αν κάνουμε κλικ και πιο δίπλα, κάτω από το facet “Resolution”, στο 6 megapixels θα έχουμε στα αποτελέσματα τις μηχανές Sony με 6 megapixels. Το faceted search δίνει όμως και τη δυνατότητα να αφαιρέσουμε facets από τα φίλτρα μας. Αν διαγράψουμε το facet Sony τότε θα μας μείνουν οι κάμερες με 6 megapixels ανεξαρτήτως κατασκευαστή.

Το faceted search όπως είδαμε παρέχει έναν πολύ αποτελεσματικό τρόπο ώστε οι χρήστες να συγκεκριμενοποιήσουν τα κριτήρια αναζήτησης μόνο στα όσα τους χρειάζονται. Ορισμένα πλεονεκτήματα αυτής της τεχνική είναι:

- Άμεση ανταπόκριση – Οι χρήστες βλέπουν άμεσα τα αποτελέσματα των επιλογών τους ανάλογα με τα κριτήρια.
- Οι χρήστες ξέρουν από πριν πόσα αποτελέσματα θα δουν. Οι μηδενικές τιμές απαλείφονται για να μην επηρεάζουν τα υπόλοιπα αποτελέσματα.
- Δεν υπάρχει συγκεκριμένος τρόπος αναζήτησης. Οι χρήστες μπορούν να προσθέσουν και να αφαιρέσουν facets κατά βούληση.

2.9. Ειδικές περιπτώσεις

2.9.1. Wolfram

Μια ιδιαίτερη περίπτωση μηχανής αναζήτησης που αξίζει να αναφερθεί είναι το Wolfram. Πρόκειται για μια online υπηρεσία που συλλέγει και υπολογίζει πληροφορίες για αυτή καθαυτή την λέξη που θα εισάγει ο χρήστης αντί να αναζητήσει έγγραφα στο διαδίκτυο που να αναφέρονται σε αυτή.

http://www.wolframalpha.com/

Δοκιμάζοντας να εισάγετε το όνομα μιας πόλης θα βρείτε μια πληθώρα πληροφοριών μαζεμένες και ταξινομημένες σε πίνακες χωρίς να χρειαστεί να αναζητήσετε άρθρα ή κείμενο. Οι πληροφορίες περιλαμβάνουν από γεωγραφικές πληροφορίες μέχρι και ιστορικές καθώς και πολλές επιπλέον, όπως καιρικές συνθήκες τη στιγμή της αναζήτησης κ.α.

2.9.2. Mitos

Επίσης ιδιαίτερη περίπτωση είναι και ο Mitos το οποίο μπορείτε να το βρείτε στο link http://groogle.csd.uoc.gr:8080/mitos/ . Ο Mitos είναι μια μηχανή αναζήτησης που φτιάχτηκε στα πλαίσια του μαθήματος HY-463 (Συστήματα Ανάκτησης Πληροφορίας), από φοιτητές του Τμήματος Επιστήμης Υπολογιστών του Πανεπιστημίου Κρήτης. Η υλοποίησή του ξεκίνησε το εαρινό εξάμηνο του 2006 και έκτοτε βρίσκεται σε συνεχή ανάπτυξη σε συνεργασία και με το Εργαστήριο Πληροφοριακών Συστημάτων του Ινστιτούτου Πληροφορικής του Ιδρύματος Τεχνολογίας Έρευνας.

Στην ελληνική μυθολογία, ο Θησέας κατάφερε να βγει έξω από το λαβύρινθο στη Κνωσό, με τη βοήθεια του μίτου της Αριάδνης, μία μπάλα νήματος την οποία ξετύλιγε καθώς προχωρούσε μέσα στο λαβύρινθο. Κατά αναλογία, ο Mitos, είναι μία μηχανή αναζήτησης που στοχεύει στο να καθοδηγήσει τους χρήστες στο λαβύρινθο του διαδικτύου.

Ο Mitos συνδυάζει το keyword με το faceted search. Μπορείτε να φιλτράρετε τα παραγόμενα αποτελέσματα σύμφωνα με τα facets που θα βρείτε για μεγαλύτερη ακρίβεια στα αποτελέσματα.

3. Faceted search

Στο κεφάλαιο αυτό θα αναλύσουμε σε βάθος το faceted search, τις τεχνικές και τις τεχνολογίες που χρησιμοποιεί καθώς και πρακτικές εφαρμογές της συγκεκριμένης μεθόδου.

Πριν όμως επεκταθούμε περισσότερο στο faceted search θα πρέπει να αναλύσουμε ορισμένες έννοιες που είναι απόλυτα συνυφασμένες με αυτό. Οι έννοιες αυτές με τη σειρά που θα αναλυθούν είναι οι εξής:

- Information Retrieval
- Information Extraction
- Entity Extraction

3.1. Information Retrieval (IR)

Πρόκειται ουσιαστικά για την εξόρυξη πληροφορίας από το διαδίκτυο, από έγγραφα, από meta-δεδομένα των εγγράφων καθώς και από σχεσιακές βάσεις δεδομένων. Το IR βασίζεται στην επιστήμη της Πληροφορικής, στα Μαθηματικά, στη Βιβλιοθηκονομία, στην επιστήμη της πληροφορίας, στη Στατιστική κ.α.

Ένα μεγάλο μειονέκτημα του IR είναι ότι επεξεργάζεται πάρα πολύ μεγάλο όγκο πληροφορίας το οποίο αναγκάζει πολλές εταιρίες και πανεπιστήμια να χρησιμοποιούν αυτοματοποιημένα IR συστήματα τα οποία φιλτράρουν τις πληροφορίες και συγκρατούν μόνο όσα δεδομένα ακριβώς χρειάζονται.

Ένα τρανό παράδειγμα τέτοιων εφαρμογών είναι φυσικά οι γνωστές σε όλους μηχανές αναζήτησης.

3.2. Information Extraction

Το Information Extraction είναι μια διαδικασία που ανήκει στο IR (Information Retrieval) και έχει σαν στόχο να εξάγει δομημένη πληροφορία από μη δομημένα ή και ημιδομημένα έγγραφα στο διαδίκτυο. Στις περισσότερες περιπτώσεις η δραστηριότητα αυτή αφορά έγγραφα τα οποία αποτελούνται από ανθρώπινη γλώσσα, στα πλαίσια του NLP(Natural Language Processing).

Ένας επιπλέον στόχος του IE είναι να κάνει και ένα λογικό διαχωρισμό των δεδομένων που επεξεργάζεται και να ξεχωρίσει αν αναφέρεται στο αντικείμενο αναφοράς μας ή όχι.

3.2.1. IE και WWW

Το μεγάλο στοίχημα φυσικά του IE ήταν η επιτυχής επεξεργασία του τεράστιου όγκου δεδομένων που υπάρχει στο διαδίκτυο. Απαραίτητες προϋποθέσεις για να ευδοκιμήσει η συγκεκριμένη διαδικασία είναι τα συστήματα που ενσωματώνουν τη λειτουργία IE να είναι φθηνά και ευέλικτα ώστε να προσαρμόζονται στην ολοένα και μεγαλύτερη γιγάντωση της πληροφορίας στο διαδίκτυο.

3.3. Entity Extraction

Η διαδικασία όπου εξάγονται με αυτόματο τρόπο meta-δεδομένα από μη δομημένα έγγραφα ονομάζεται Entity Extraction. Η όλη προσέγγιση έχει να κάνει με τον εντοπισμό λέξεων κλειδιών όπως ονόματα, τοποθεσίες, ημερομηνίες, ειδικοί όροι, όροι σχετικοί με προϊόντα κ.α. Με αυτόν τον τρόπο δίνεται η δυνατότητα σε εταιρίες όχι μόνο να βελτιώσουν τις αναζητήσεις με λέξεις κλειδιά αλλά και να δώσουν ώθηση και σε άλλες τεχνικές αναζήτησης όπως το faceted και το semantic search.

Είναι συνηθισμένο να λέγεται ότι το 90% της χρήσιμης πληροφορίας από οποιαδήποτε ιστοσελίδα στο διαδίκτυο είναι αποθηκευμένη σε μη δομημένα έγγραφα και όχι σε βάσεις δεδομένων. Ευτυχώς, με τη βοήθεια τεχνολογιών όπως οι RSS, Atom κ.α. μπορούμε να προσπελάσουμε τεράστιο όγκο τέτοιων εγγράφων. Αξίζει να αναφέρουμε εδώ ότι χάρη στην Apache Foundation και την υλοποίηση ορισμένων γλωσσικών εργαλείων δίνεται η δυνατότητα ακόμη και σε μικρές εταιρίες και επιχειρήσεις, χωρίς να υπάρχουν απαραίτητα γνώσεις προγραμματισμού, να υλοποιήσουν αξιόλογα συστήματα για entity extraction.

3.4. Faceted Search: Γενικά

Όπως είναι προφανές, η ολοένα και μεγαλύτερη διόγκωση της πληροφορίας που υπάρχει στο διαδίκτυο καθιστά απαραίτητη και τη βελτιστοποίηση των μεθόδων αναζήτησης της. Περιγράψαμε ορισμένους τρόπους παραπάνω αλλά εδώ θα αναλύσουμε το faceted search, μέθοδος η οποία προτιμάται όλο και περισσότερο , ειδικά σε ηλεκτρονικά καταστήματα. Αυτό ισχύει γιατί καθιστά πολύ ευκολότερη την εύρεση του προϊόντος που αναζητάμε. Επιπλέον παρακάτω θα γίνει ενδελεχής ανάλυση για τον τρόπο που προσεγγίζουμε το faceted search όταν έχουμε να κάνουμε με web design.

Αξίζει να αναφέρουμε το γεγονός ότι λόγω της παντοκρατορίας της Google οι χρήστες ίσως αργήσουν να εγκλιματιστούν με την συγκεκριμένη τεχνολογία και να την αφομοιώσουν όσο έχουν αφομοιώσει το keyword search. Η εν λόγω εταιρία βέβαια κάνει σταθερά βήματα προς την κατεύθυνση του faceted search.

Ας θυμηθούμε λίγο τον ορισμό του faceted search που αναφέρθηκε πιο πάνω και να περάσουμε σε μερικά παραδείγματα για να κατανοήσουμε καλύτερα την φιλοσοφία του.

Το faceted search ή αλλιώς faceted navigation ή faceted browsing είναι μια τεχνική πρόσβασης σε συλλογές πληροφοριών που ταξινομούνται με facets και επιτρέπει τους χρήστες να φιλτράρουν την διαθέσιμη πληροφορία.

Αντί να αναλύσουμε τον ορισμό του facet θα δώσουμε ένα παράδειγμα ώστε να γίνει καλύτερα κατανοητό τι ακριβώς σημαίνει και σε τι αποσκοπεί. Τα facets είναι τα χαρακτηριστικά βάση των οποίων ταξινομείται το αντικείμενο αναζήτησης μας. Αυτά συνήθως ή ορίζονται από τον administrator ή παράγονται από τις μεθόδους information retrieval και entity extraction που περιγράψαμε παραπάνω. Από κάποιο σημείο και μετά βέβαια, όταν η πληροφορία είναι ογκώδης, προτιμούμε τα αυτοματοποιημένα συστήματα και ήδη έτοιμα facets.

Ας υποθέσουμε ότι έχουμε το παρακάτω κείμενο περιγραφής ενός κεντρικού επεξεργαστή και μια ιστοσελίδα όπου και θέλουμε να τον εισάγουμε στη βάση μας σαν προϊόν και να ορίσουμε τα facets.

Η AMD παρουσιάζει την εξέλιξη των τετραπύρηνων επεξεργαστών !
*Η προηγμένη γενιά **A8 X4** είναι ειδικά σχεδιασμένη με βάση την επαναστατική μικρο-αρχιτεκτονική των **32nm**, γεγονός που μεγιστοποιεί τις επιδόσεις.*

*Ο νέος **AMD A8 X4 3850** προσφέρει ασυναγώνιστες πολυ-πύρηνες επιδόσεις για εξαιρετική high definition εμπειρία, αναβαθμισμένη multitasking απόδοση και με πρωτοποριακές λύσεις εξοικονόμησης ενέργειας. Με συχνότητα λειτουργίας στα **2,90 GHz**, ενώ είναι εξοπλισμένος με **4MB Cache**, προσφέροντας άφθονη ταχύτητα και ισχύ σε κάθε εφαρμογή.*

Με μια πρώτη παρατήρηση καταλαβαίνουμε ότι πρόκειται για έναν επεξεργαστή κατασκευασμένο από την AMD. Άρα αυτόματα έχουμε εξάγει το πρώτο facet «Κατασκευαστής» του οποίου μάλιστα μία τιμή του είναι το AMD. Σε κείμενο άλλου επεξεργαστή θα μπορούμε να εξάγουμε και άλλες τιμές για το facet επεξεργαστής. Πιο κάτω βλέπουμε ένα άλλο υποψήφιο facet την «Αρχιτεκτονική» που από το συγκεκριμένο κείμενο εξάγουμε την τιμή 32nm. Λίγο παρακάτω εντοπίζουμε τα facets «Συχνότητα» και «Cache» με τιμές 2,90GHz και 4MB αντίστοιχα.

Τα συγκεκριμένα facets που ορίστηκαν πιο πάνω θα ισχύσουν για κάθε επόμενο κείμενο που αφορά CPU οπότε θα εξαχθούν ακόμα περισσότερες τιμές.

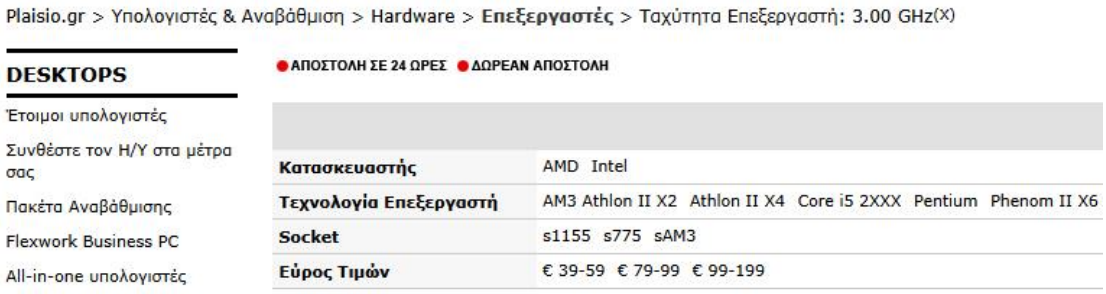
Ας ρίξουμε μια ματιά τώρα στο www.plaisio.gr όπου και υλοποιείται το faceted search για να δούμε πως περίπου θα δείχνει ένα ολοκληρωμένο σύστημα με facets.

περιορίστε τα αποτελέσματα

Κατασκευαστής	AMD Intel
Τεχνολογία Επεξεργαστή	AM3 Athlon II X2 AMD Fusion APU-Llano Athlon II X4 Celeron Core 2 Duo Core i3 2XXX Core i3 5XX Core i5 2XXX Core i5 6XX Core i5 7XX Core i7 2XXX Core i7 8XX Core i7 9XX Pentium Phenom II X2 Phenom II X4 Phenom II X6 Sempron
Socket	s FM1 s1155 s1156 s1366 s775 sAM3
FSB	1066 MHz 2000 MT-sec 4.8 GT-sec 5200 MT-sec 6.4 GT-sec 800 MHz
Ταχύτητα Επεξεργαστή	2.40 GHz 2.60 GHz 2.70 GHz 2.80 GHz 2.90 GHz 2.93 GHz 3.00 GHz 3.06 GHz 3.10 GHz 3.20 GHz 3.3 GHz 3.33 GHz 3.40 GHz 3.46 GHz

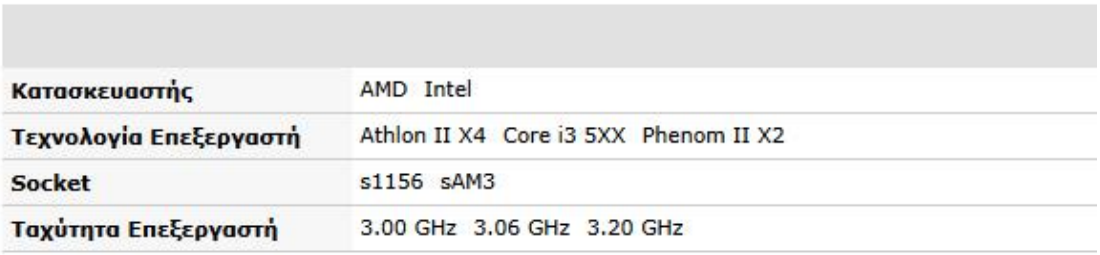
Εικόνα 5 - Faceted Search στο plaisio.gr

Εδώ έχουμε τα εξής facets: «Κατασκευαστής» «Τεχνολογία Επεξεργαστή» «Socket» «FSB» «Ταχύτητα Επεξεργαστή» με τις εκάστοτε τιμές τους. Τα αποτελέσματα μας θα είναι όλοι οι επεξεργαστές που πληρούν τα εξής χαρακτηριστικά. Το faceted search μας δίνει τη δυνατότητα να περιορίσουμε τα αποτελέσματα σε αυτά που πραγματικά θέλουμε. Για χάρη του παραδείγματος θα επιλέξουμε έναν επεξεργαστή με ταχύτητα στα 3.00GHz. Το αποτέλεσμα είναι να διαμορφωθούν τα facets μας ως εξής:



Εικόνα 6 - Faceted search στο plaisio.gr II

Παρατηρούμε περιορίστηκαν αρκετά τα αποτελέσματα μας αλλά έχουμε και νέο facet «Εύρος τιμών» όπου μπορούμε να αναζητήσουμε αναλυτικότερα τον επεξεργαστή που θέλουμε. Τα αποτελέσματα μας πλέον περιλαμβάνουν όλους τους επεξεργαστές που έχουν συχνότητα λειτουργίας 3.00GHz. Πάνω δεξιά η ιστοσελίδα μας έχει ενημερώσει για τα facets που έχουμε επιλέξει. Το faceted search δίνει και τη δυνατότητα απεπιλογής. Μπορούμε να διαγράψουμε το facet που επιλέξαμε και να επιστρέψουμε στα προηγούμενα αποτελέσματα. Μπορούμε να διαγράφουμε facets από τις επιλογές μας κατά βούληση και με όποια σειρά θέλουμε βελτιώνοντας έτσι την εμπειρία της αναζήτησης και τα αποτελέσματα μας. Για παράδειγμα μπορούμε να επιλέξουμε από το «Εύρος τιμών» το €79-99 και στη συνέχεια να απεπιλέξουμε το 3.00GHz και να καταλήξουμε με τα αποτελέσματα που αφορούν όλους τους επεξεργαστές στο εύρος που επιλέξαμε. Τα facets μας θα διαμορφωθούν ως εξής:



Εικόνα 7 - Faceted Search στο πλαίσιο III

Όπως γίνεται εύκολα αντιληπτό οι δυνατότητες φιλτραρίσματος και οι συνδυασμοί τους είναι πάρα πολλοί και πολύ αποδοτικοί. Η απόδοση βέβαια είναι σχετική και εξαρτάται από το πλήθος των facets και την ακρίβεια με την οποία χαρακτηρίζουν τα προϊόντα. Όπως θα δούμε και στο κεφάλαιο του Web Design υπάρχει πάντα η χρυσή τομή μεταξύ του πλήθους των facets και της ευχρηστίας τους.

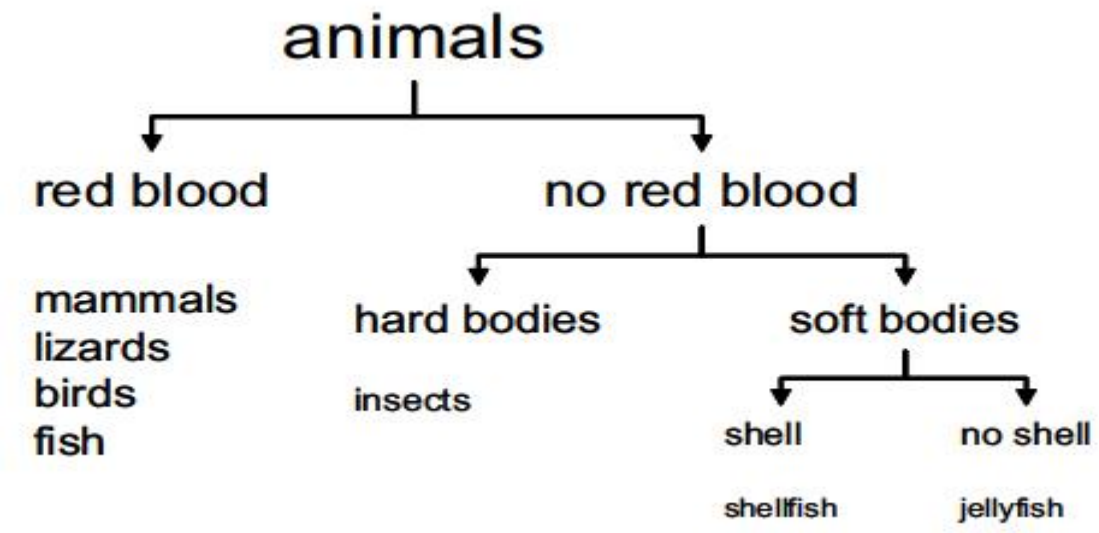
Μετά από μια γενική περιγραφή για το τι εστί faceted search θα προχωρήσουμε σε μια πιο λεπτομερής προσέγγιση ώστε να φτάσουμε στο σημείο να καταλάβουμε τι κρύβεται κάτω από αυτό που μόλις περιγράψαμε.

3.5. Faceted Search: Τι είναι τα facets;

Πριν μπούμε σε περισσότερες λεπτομέρειες όσον αφορά αυτό καθεαυτό το faceted search θα πρέπει να δούμε τι ακριβώς είναι τα facets και από πού προήλθαν. Συγκεκριμένα θα μιλήσουμε για την έννοια του classification γενικά και classification στο faceted search ειδικότερα. Στο συγκεκριμένο κεφάλαιο θα κάνουμε και μια μικρή ιστορική αναδρομή η οποία, για να ακριβολογούμε, αριθμεί σχεδόν 2,5 χιλιετίες!

3.5.1. Classification: Το δέντρο του Αριστοτέλη

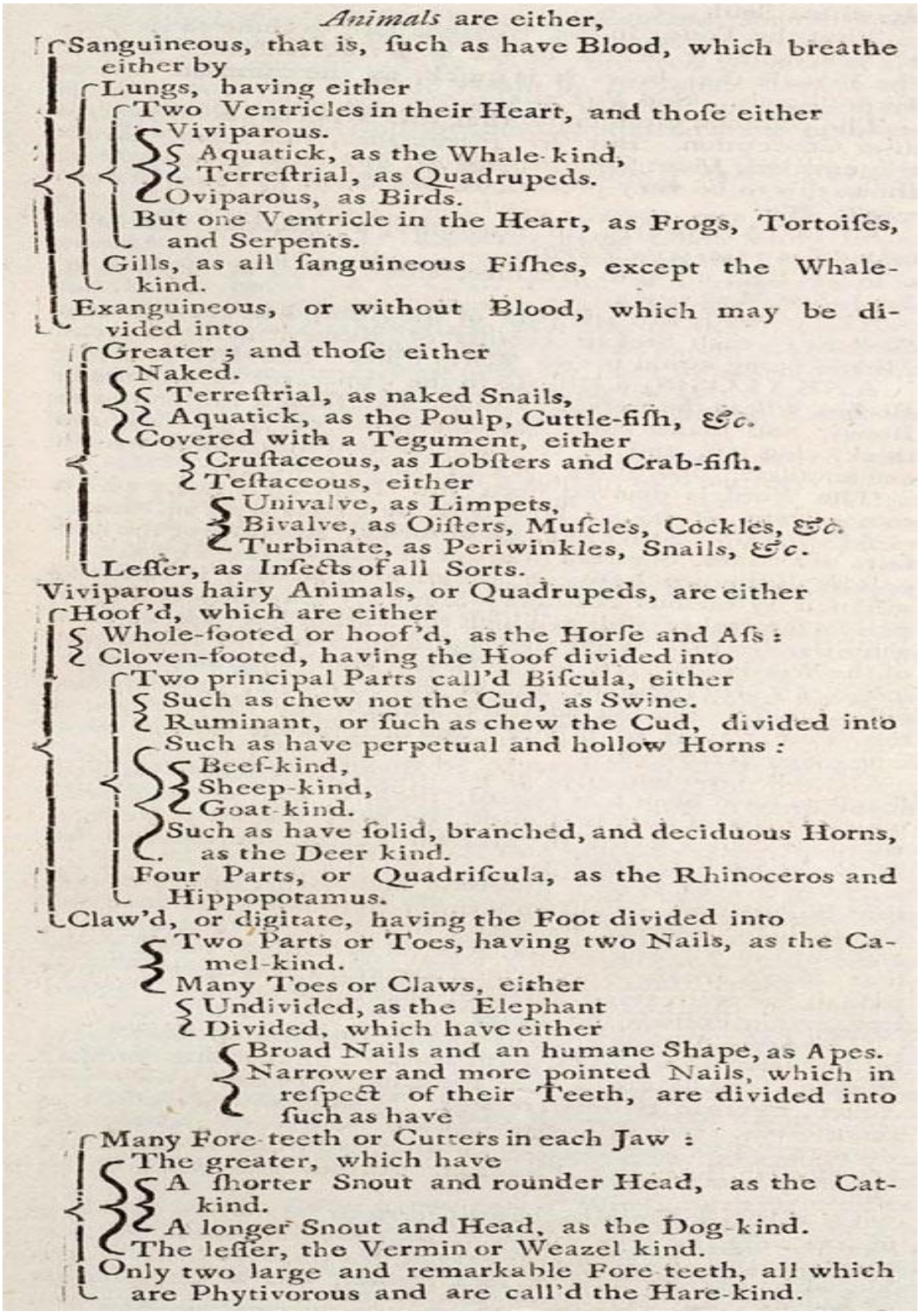
Ο Αριστοτέλης ήταν ίσως ο μόνος άνθρωπος που στην εποχή του τα ήξερε όλα. Έννοιες όπως η φιλοσοφία, η λογική, οι φυσικές επιστήμες κ.α. είναι άρρηκτα συνδεδεμένα με το όνομα του. Τι σχέση έχει όμως ο Αριστοτέλης με το faceted search; Ήταν ανάμεσα στους πρώτους που επινόησαν σύστημα ταξινόμησης. Συγκεκριμένα, ο Αριστοτέλης το εφάρμοσε στα ζωντανά όντα. Αρχικά τα χώρισε σε 2 κατηγορίες: ζώα και φυτά. Στη συνέχεια χώρισε τα ζώα σε αυτά που έχουν αίμα και σε αυτά που δεν έχουν. Αυτά που έχουν τα χώρισε παραπέρα σε θηλαστικά και σε ζώα που κάνουν αυγά.



Εικόνα 8 - Το δέντρο του Αριστοτέλη

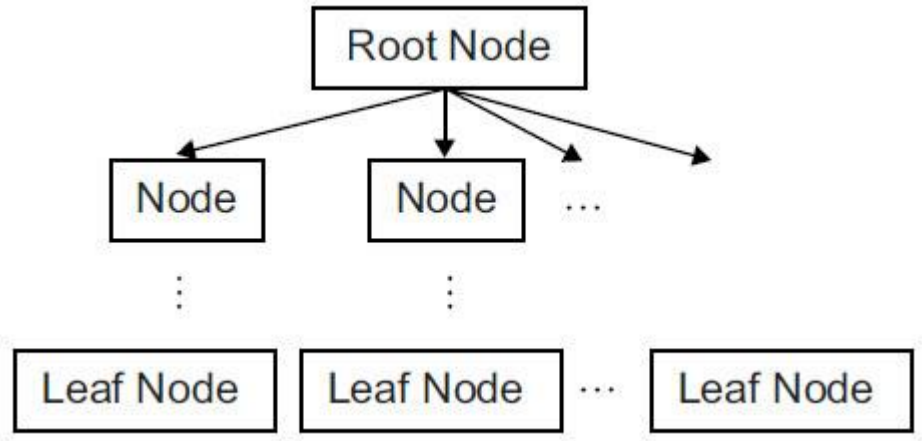
Το συγκεκριμένο δέντρο είχε γίνει ορόσημο για τη ζωολογία για πολλές εκατονταετίες και έπειτα. Ο Αριστοτέλης ήταν λοιπόν ο πρώτος ταξινόμος (taxonomist) και ο ρόλος του μάλιστα είναι ιδιαίτερα δημοφιλής σε όσους ασχολούνται με την ταξινομία. Αν το καλοσκεφτούμε η λέξη taxonomy είναι ελληνική και προέρχεται από τις λέξεις «τάξης» και «νόμος».

Ενώ παλιότερα η λέξη taxonomy αναφερόταν συγκεκριμένα στον ζωικό διαχωρισμό σήμερα χρησιμοποιείται παντού όπου υπάρχει ανάγκη για ιεραρχική ταξινόμηση όπως επίσης και σε οτιδήποτε αντιπροσωπεύεται από δέντρα σαν αυτό του Αριστοτέλη.



Εικόνα 9 - Αναλυτική παρουσίαση του δέντρου του Αριστοτέλη

Το δέντρα αυτά όπως παρατηρούμε έχουν έναν ριζικό κόμβο (root node) στην κορυφή, τα «φύλλα» στο τέλος και κάθε κόμβος ενώνεται με τον επόμενο και τον προηγούμενο με «κλαδιά»



Εικόνα 10 - Ανατομία ενός δέντρου

Κάθε γονικός κόμβος μπορεί να έχει πολλά παιδιά αλλά ένας οποιοδήποτε κόμβος που δεν είναι ριζικός έχει μόνο έναν γονέα. Η περίπτωση όπου ένας κόμβος έχει πολλούς γονικούς κόμβους το ονομάζουμε πολυιεραρχία (polyhierarchy). Τέτοιες υλοποιήσεις προσφέρουν ελευθερία αλλά αποδεικνύονται ιδιαίτερα πολύπλοκες και δεν προτιμούνται.

Ο ριζικός κόμβος σε ένα ταξινομικό σύστημα αναπαριστά την κεντρική ταξινόμηση, το κεντρικό χαρακτηριστικό του αντικειμένου που μας ενδιαφέρει ανά περίπτωση. Στο δέντρο του Αριστοτέλη αυτός ο κόμβος αποτελεί όλα τα ζωντανά όντα ενώ στα φύλλα του βρίσκονται τα ίδια τα ζώα. Πολύ σημαντική ιδιότητα που έχουν τα δέντρα είναι ότι για κάθε σετ αντικειμένων που χαρακτηρίζονται από έναν κόμβο υπάρχει μια και μοναδική διαδρομή από τον ριζικό κόμβο. Ως εκ τούτου καταλαβαίνουμε ότι οι ταξινομήσεις που μελετάμε χαρακτηρίζονται από μια αυστηρή λογική τοποθέτηση της γνώσης που αναπαριστά. Αξίζει να αναφέρουμε ότι σε όλα τα δέντρα ο σκοπός είναι να οργανωθούν συγκεκριμένα αντικείμενα και όχι αφηρημένες έννοιες.

3.5.2. Facets: Ranganathan’s Colons

Το 1876 η προσέγγιση του Αριστοτέλη στην ταξινόμηση κέντρισε το ενδιαφέρον του Melvil Dewey ώστε να τον κάνει να υλοποιήσει το Dewey Decimal Classification (DDC), ένα ταξινομικό σύστημα που χρησιμοποιείται από πολλές βιβλιοθήκες. Το 2003 μάλιστα το συγκεκριμένο σύστημα είχε ήδη 110.000 «παιδιά» στο «δέντρο» του. Κάθε κόμβος του DCC είναι μια σειρά από δεκαδικά ψηφία που στις περισσότερες περιπτώσεις κάθε επόμενο ψηφίο αναπαριστά ένα κλαδί του δέντρου. Για παράδειγμα δείτε τη συγκεκριμένη διαδρομή στον κόμβο «Cats»:

600 Technology
630 Agriculture and related technologies
636 Animal husbandry
636.8 Cats

Ασφαλώς και θα αναρωτηθείτε τώρα τι σχέση έχουν οι γάτες με την τεχνολογία. Στην πραγματικότητα στο DCC οι γάτες ανήκουν τουλάχιστον σε 2 κατηγορίες. Μία είναι η ζωολογία και μία είναι η κτηνοτροφία που είδαμε στο παράδειγμα. Τα βιβλία όμως που περιγράφουν τη συμπεριφορά της γάτας που ανήκουν; Αυτό εξαρτάται από αυτόν που κατασκευάζει τον κατάλογο και την υποκειμενική του άποψη.
Εδώ μπορούμε να δούμε την πρόκληση στη χρήση μιας ενιαίας ιεραρχικής ταξινομίας για να αντιπροσωπεύσουμε μια διαφορετική συλλογή της πληροφορίας που διαθέτουμε. Κάθε κλαδί στην ιεραρχία απαιτεί από εμάς να κάνουμε μια ανεξίτηλη επιλογή, μέσα από αυθαίρετους ορισμούς καμιά φορά.

Ο άνθρωπος που εντρύφησε περισσότερο στο πρόβλημα ήταν ο Shiyali Ramamrita Ranganathan, ένας ινδός μαθηματικός και βιβλιοθηκάριος. Ο Ranganathan ήθελε ένα ταξινομικό σύστημα, το οποίο ξέφευγε λίγο από τα στενά όρια της αυστηρής λογικής που χαρακτήριζε τα μέχρι τότε δέντρα. Έτσι το 1933 εισήγαγε αυτό που είναι γνωστό στο χώρο ως Colon classification scheme. Μέχρι στιγμής ομολογουμένως οι πληροφορίες για αυτό το σύστημα μοιάζουν να ξεφεύγουν από το θέμα της πτυχιακής αλλά κάπου εδώ θα αναφέρουμε ότι το σύστημα του Ranganathan ήταν το πρώτο σύστημα ταξινόμησης βιβλιοθήκης που βασίζεται σε ανάλυση facets. Ο ίδιος ο Ranganathan δεν τα αποκαλούσε facets αλλά isolates.

Ο Ranganathan διαχώρισε τον κόσμο της γνώσης σε 5 βασικές κατηγορίες:

- Προσωπικότητα
- Σκοπός
- Ενέργεια
- Χώρος
- Χρόνος

Αυτές οι κατηγορίες αποτέλεσαν την βάση της ανάλυσης των facets. Ο όρος colon classification προήλθε από τον διαχωρισμό που κάνει ανάμεσα στα αντικείμενα με τη χρήση του σημείου στίξης «άνω κάτω τελεία» (colon). Δείτε το παρακάτω παράδειγμα:

L2153:4725:63129:B28

Τη συγκεκριμένη αλληλουχία την χωρίζουμε σε επιμέρους facets σύμφωνα με το πρότυπο του Ranganathan

Medicine (L) → Digestive system (L2) → Mouth (L21) → Palate (L215) → Soft palate (L2153)
Disease (4) → Structural disease (47) → Tumor (472) → Cancer (4725)

Treatment (6) → Treatment by chemical substances (63) → Treatment by a chemical element
(631) → Treatment by a group 2 chemical element (6312) → Treatment by radium (63129)
Mathematical study (B) → Algebraical study (B2) → Statistical study (B28)

Όπως μπορούμε να δούμε στο παράδειγμα το σύστημα ξεφεύγει από τα αυστηρά ταξινομικά συστήματα όπως το DCC ή οποιοδήποτε άλλο σύστημα που χρησιμοποιεί αυστηρή ιεραρχική ταξινόμηση.

Το σύστημα του Ranganathan επέτρεψε τον διαχωρισμό σε ολοένα και περισσότερα facets κάθε αντικειμένου το οποίο λειτούργησε ως βάση για το faceted search αργότερα.

3.5.3. Ontologies

Παρόλο που η πτυχιακή ασχολείται με το faceted search θα ήταν λάθος να μην αναφέρουμε έννοιες και τεχνικές πρόσβασης της πληροφορίας οι οποίες βοήθησαν στην επέκταση των ορίων του faceted search. Συγκεκριμένα κάποιοι επιστήμονες που ασχολούνται με την εξόρυξη πληροφοριών θεώρησαν ότι τα facets περιορίζουν τις αναζητήσεις τους οπότε κατέφυγαν στις οντότητες (ontologies). Όπως και η λέξη taxonomy έτσι και η λέξη ontologies προέρχεται από την ελληνική γλώσσα και συγκεκριμένα από τις λέξεις «όντος» που σημαίνει έμβιος και «λόγος» που δηλώνει την θεωρία. Στην επιστήμη της πληροφορίας η λέξη ontology έχει πιο πρακτική σημασία: Δηλώνει ένα πλήθος θεμάτων και τη σχέση μεταξύ τους. Για παράδειγμα, χρησιμοποιώντας ontologies στην πρόταση «Ο Νίκος είναι ο συγγραφέας του βιβλίου Ζωή», εστιάζουμε περισσότερο στη σχέση μεταξύ του συγγραφέα και του βιβλίου παρά στις έννοιες «συγγραφέας» και «βιβλίο» χωριστά.

Το 2001 το World Wide Web Consortium δημιούργησε το Web Ontology Working Group, το οποίο με τη σειρά του δημιούργησε την γλώσσα Web Ontology Language το 2004. Αυτή η γλώσσα αποτέλεσε το στάνταρντ για τις μεθόδους του semantic web. Το project αυτό δεν έχει τελειώσει γιατί δίνεται βαρύτητα στον τρόπο με τον οποίο θα αναπαρασταθεί η πληροφορία.

3.6. Information Retrieval

Έχοντας κατανοήσει τα χαρακτηριστικά και το κίνητρο των συστημάτων για αναπαράσταση της πληροφορίας, ερχόμαστε τώρα να λύσουμε το πρόβλημα της πρόσβασης στην πληροφορία που χρειαζόμαστε.

Σε αυτό το κεφάλαιο θα ερευνήσουμε με μεγαλύτερη λεπτομέρεια την έννοια του information retrieval.

3.6.1. Σχετικότητα

Ο κύριος σκοπός του information retrieval είναι να βοηθήσει τον χρήστη να ανακτήσει έγγραφα τα οποία σχετίζονται με τις πληροφοριακές ανάγκες του. Η έννοια της σχετικότητας που εξετάζεται εδώ είναι δύσκολο να οριστεί. Το 1964 ο William Goffman έγραψε:

Ως σχετικότητα ορίζουμε την ποσότητα της πληροφορίας που αντιπροσωπεύει το ερώτημα(query) που έθεσε ο χρήστης. Η σχέση βέβαια μεταξύ του εγγράφου και του ερωτήματος παρόλο που είναι αναγκαίο να αποσαφηνιστεί δεν αρκεί για να ορίσουμε τη σχετικότητα μεταξύ των δύο.

Έχουν γίνει πολυάριθμες συζητήσεις, συνέδρια καθώς και έχουν γραφτεί πάρα πολλά κείμενα που προσπαθούν να αναλύσουν την έννοια της σχετικότητας στην πληροφορία αλλά στο τέλος αντί να συγκλίνουν οι απόψεις μοιάζουν να διαφωνούν συνέχεια ως το πώς "μετριέται" η σχετικότητα. Ίσως ο τελικός χρήστης να είναι ο αρμόδιος κριτής καθώς ο ίδιος γνωρίζει τι ακριβώς ψάχνει. Αυτό το έχουν σκεφτεί πολλοί επιστήμονες οπότε έχουν συγκεντρώσει τις προσπάθειες τους ώστε να μελετήσουν σενάρια αναζήτησης από διάφορους χρήστες και να αναλύσουν τα αποτελέσματα.

3.6.2. Set Retrieval

Η αναζήτηση στο διαδίκτυο εφαρμόζεται πλέον σχεδόν παντού. Μέχρι να φτάσουμε σε αυτό το σημείο όμως προηγήθηκαν μηχανές που δούλευαν τελείως διαφορετικά από ότι οι μοντέρνοι διάδοχοι τους. Αυτές οι παλιότερες μηχανές υιοθετούσαν ένα σύστημα που είναι γνωστό ως Set Retrieval Model. Το μοντέλο αυτό επέστρεφε αποτελέσματα από έγγραφα σε τυχαία σειρά αντί για ταξινομημένα. Η σχετικότητα σαν έννοια ήταν ανύπαρκτη.

Το μοντέλο ήταν επίσης γνωστό και σαν Boolean retrieval model γιατί τα συστήματα που το χρησιμοποιούσαν επέτρεπαν στον χρήστη να εισάγει ερωτήματα χρησιμοποιώντας τελεστές AND και OR στα ερωτήματα του. Πολλά τέτοια συστήματα επέκτειναν την λειτουργικότητα τους επιτρέποντας περισσότερους τελεστές προς χρήση από τους χρήστες. Επίσης, ορισμένα συστήματα τους επέτρεπαν να ορίσουν σε ποιο σημείο ενός έγγραφου ενδεχομένως να βρισκόταν η λέξη ή η φράση που ψάχνουν(πχ επικεφαλίδα, πρόλογος, επίλογος κλπ).

Field Code	Field Name	Field Code	Field Name
PN	Patent Number	IN	Inventor Name
ISD	Issue Date	IC	Inventor City
TTL	Title	IS	Inventor State
ABST	Abstract	ICN	Inventor Country
ACLM	Claim(s)	LREP	Attorney or Agent
SPEC	Description/Specification	AN	Assignee Name
CCL	Current US Classification	AC	Assignee City
ICL	International Classification	AS	Assignee State
APN	Application Serial Number	ACN	Assignee Country
APD	Application Date	EXP	Primary Examiner
PARN	Parent Case Information	EXA	Assistant Examiner
RLAP	Related US App. Data	REF	Referenced By
REIS	Reissue Data	FREF	Foreign References
PRIR	Foreign Priority	OREF	Other References
PCT	PCT Information	GOVT	Government Interest
APT	Application Type		

Εικόνα 11 - Interface που χρησιμοποιεί Boolean

Η παραπάνω εικόνα μας δείχνει ένα παράδειγμα από interface που χρησιμοποιεί Boolean. Οι χρήστες ορίζουν ερωτήματα έχοντας στη διάθεσή τους πολλούς τελεστές και διάφορα εργαλεία ώστε να κάνουν πιο επιτυχημένη την αναζήτηση τους.

Παρόλη την αποτελεσματικότητα του, το set retrieval μοντέλο έχει ορισμένες στοιχειώδης αδυναμίες. Καθώς οι χρήστες εισάγουν τα ερωτήματα τους, έρχονται σε ένα δίλλημα. Να προτιμήσουν μέγιστη ακρίβεια ή μέγιστη ανάκληση. Δυστυχώς όμως δε γίνεται να τα συνδυάσουν. Αυτή η δυσκολία είναι που κατέστησε σπάνιες τις μηχανές αναζήτησης που χρησιμοποιούν αυτό το μοντέλο, παρόλο που πολλές από τις μοντέρνες υλοποιήσεις μηχανών χρησιμοποιούν Boolean τελεστές.

3.6.3. Ranked Retrieval

Υπάρχει άραγε εναλλακτική στο μοντέλο του Set Retrieval που θεωρείται δύσχρηστο; Η απάντηση είναι ναι και πρόκειται για μια μέθοδο που χρησιμοποιείται από τις μοντέρνες μηχανές αναζήτησης. Η μέθοδος αυτή είναι γνωστή ως Ranked Retrieval ή Relevance Ranking.

Το 1961 ο Calvin Mooers, ο οποίος φημίστηκε για την επινόηση της έννοιας Information Retrieval, εξέφρασε τη δυσαρέσκεια του για το μοντέλο του Set

Retrieval. Μέχρι τότε ίσχυε η πεποίθηση ότι η άλγεβρα Boole ήταν το ιδανικό εργαλείο για την υλοποίηση των μηχανών αναζήτησης. Βέβαια αυτή η άποψη αποδείχθηκε, μέσω του μοντέλου Set Retrieval, ότι ήταν λάθος. Στην αναζήτηση λοιπόν εναλλακτικής λύσης ορισμένοι επιστήμονες της πληροφορικής επιχείρησαν μια εντελώς διαφορετική προσέγγιση. Αντί λοιπόν να απαιτείται η εισαγωγή δομημένων ερωτημάτων, υλοποίησαν τον κώδικα της μηχανής με τέτοιο τρόπο ώστε να εισάγονται, μη δομημένα και σε μορφή απλού κειμένου, ερωτήματα από τους χρήστες ώστε να τους αποβάλλει την ανάγκη για δόμηση πολύπλοκα ορισμένων ερωτημάτων. Αντί λοιπόν για μια καθόλα ακριβής λίστα αποτελεσμάτων αντιμετωπίζουν πιο σφαιρικά την ανάκτηση εγγράφων ώστε να εξάγουν πιο πολλά σχετικά έγγραφα.

Εικόνα 12 - Ranked retrieval στο rexa.info

Η παραπάνω φωτογραφία μας δείχνει ένα παράδειγμα ranked retrieval από την ιστοσελίδα Rexa.info, μια ψηφιακή βιβλιοθήκη και μηχανή αναζήτησης που καλύπτει θέματα που αφορούν την επιστήμη των υπολογιστών. Αν στη συγκεκριμένη εφαρμογή εισάγουμε το ερώτημα “faceted search” θα διαπιστώσουμε ότι θα έχουμε 93,641 αποτελέσματα! Παρόλα αυτά τα αποτελέσματα προέρχονται ουσιαστικά από την αναζήτηση του ερωτήματος faceted OR search το οποίο δίνει μέγιστη ανάκληση αλλά πολύ μικρή ακρίβεια. Οι προγραμματιστές της ιστοσελίδας εξισορροπούν το μειονέκτημα της ακρίβειας, εμφανίζοντας πρώτα τα αποτελέσματα που είναι σχετικότερα με το ερώτημα.

Τα ερωτήματα απλού κειμένου είναι, χωρίς αμφιβολία, πολύ ευκολότερο να συνταχθούν από τους χρήστες σε σύγκριση με τις Boolean εκφράσεις. Βέβαια το τίμημα αυτής της ελευθερίας είναι ότι δεν υπάρχει καλό φιλτράρισμα των εγγράφων. Όπως διαπιστώνουμε ένα τέτοιο σύστημα βασίζεται περισσότερο στη σχετικότητα του ερωτήματος με το έγγραφο παρά με ακριβές φιλτράρισμα της πληροφορίας.

Το ranked retrieval μοντέλο οφείλει την ύπαρξη και την επιτυχία του στον Gerald Salton και στον Spark Jones. Ο πρώτος έχει συνεισφέρει τόσο πολύ σε αυτό που αποκαλούμε σήμερα information retrieval, που υπάρχει και βραβείο με το όνομα του, το Gerald Salton Award. Από όλες τις υλοποιήσεις του, αυτή που μας ενδιαφέρει εδώ είναι το μοντέλο vector space. Το συγκεκριμένο ιδιόμορφο μοντέλο αναπαριστά κάθε έγγραφο χρησιμοποιώντας vectors λέξεων και φράσεων ή όρων. Κάθε κομμάτι ενός vector εγγράφου αναπαριστά, θεωρητικά τουλάχιστον, την ισχύ του όρου που περιέχει, η οποία με η σειρά της χαρακτηρίζει το έγγραφο. Αυτό που χρειαζόταν ήταν να αποφασιστούν οι τιμές αυτών των κομματιών. Το συγκεκριμένο πρόβλημα έμελε να το λύσει η Spark Jones, άλλη μια μεγάλη επιστήμονας στον τομέα του information retrieval. Η λύση που πρότεινε είναι μια στατική μεταγλώττιση των όρων όπου αργότερα θα εισάγονταν στο vector space μοντέλο. Το συγκεκριμένο σχήμα είναι γνωστό σήμερα ως *term frequency-inverse document frequency.*

Το term frequency, είναι ο αριθμός των εμφανίσεων ενός όρου σε ένα έγγραφο διαιρούμενος με το συνολικό αριθμό των όρων που αυτό περιλαμβάνει. Όσο μεγαλύτερο είναι το term frequency τόσο περισσότερο αντιπροσωπεύει το έγγραφο ο εν λόγω όρος. Το inverse document frequency, δίνει έμφαση στο λόγο του πλήθους των σπάνιων όρων ως προς τους συνηθισμένους. Οι σπάνιοι όροι έχουν μεγαλύτερο βάρος σε σχέση με τους υπόλοιπους.

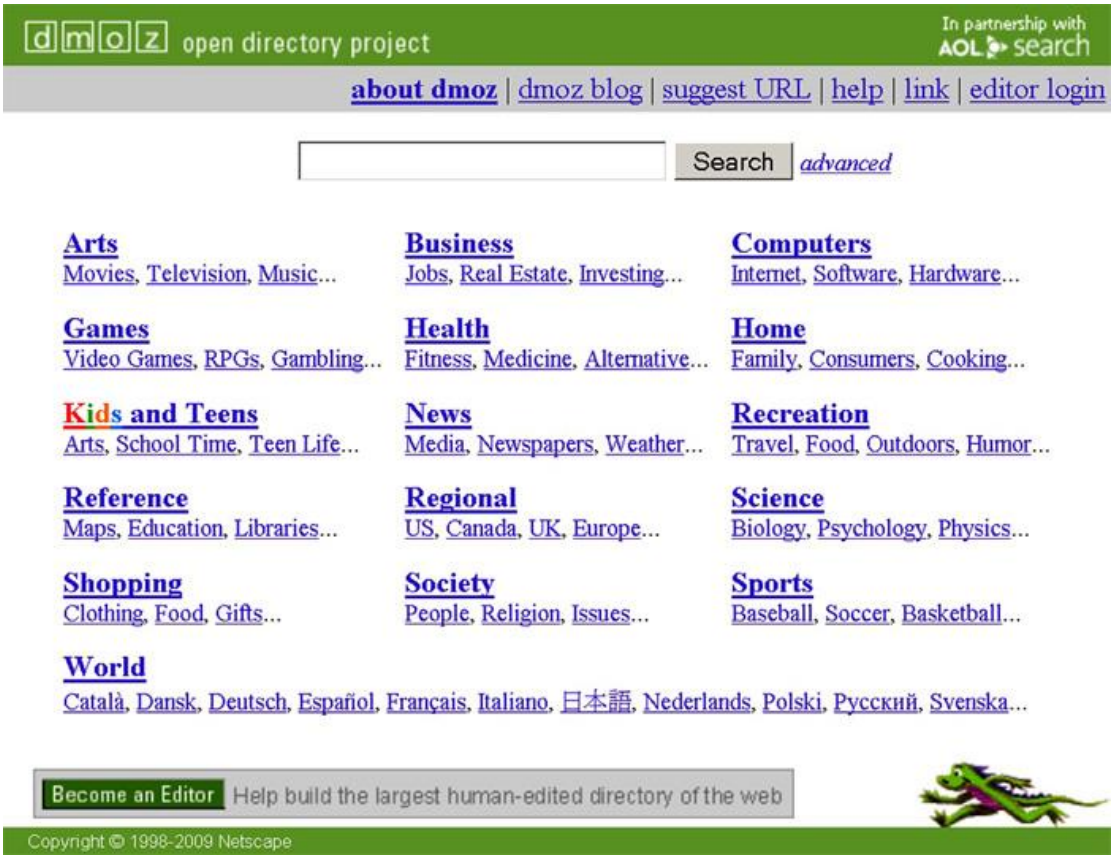
Έχουν γίνει πολλές υλοποιήσεις από ranked retrieval συστήματα από τότε που πρωτοπροσπάθησαν οι Salton και Jones. Οι σημαντικότερες από αυτές έχουν να κάνουν με μηχανές αναζήτησης που ξεφεύγουν από την λογική της επεξεργασίας ερωτήματος και στοχεύουν περισσότερο στα κείμενα αυτά καθ’ αυτά. Δύο υλοποιήσεις που είχαν σκοπό να μετρήσουν αυτά τα αποτελέσματα είναι ο αλγόριθμος HITS και ο αλγόριθμος PageRank γνωστός από τη χρήση του στο Google.

Η επιτυχία των ranked έναντι των set retrieval systems παρόλα τα πλεονεκτήματα τους έχουν και ένα τίμημα. Στα ranked retrieval συστήματα δεν μπορούμε να ορίσουμε ποια αποτελέσματα ταιριάζουν ή όχι με το ερώτημα μας. Το συγκεκριμένο μειονέκτημα ήταν ιδιαίτερα προβληματικό για να το υιοθετήσει το faceted search το οποίο χρησιμοποιεί κατά κόρον ένα set retrieval σύστημα.

3.6.4. Directory Navigation

Η αναζήτηση με τη χρήση απλού κειμένου έχει γίνει η πιο δημοφιλής στους χρήστες σε σημείο που εύκολα ξεχνάμε τις υπόλοιπες υλοποιήσεις. Σε προηγούμενες παραγράφους συζητήσαμε για ταξινομήσεις, μια μέθοδο που χρησιμοποιήθηκε για

την αναπαράσταση της πληροφορίας για περισσότερο από 2 χιλιετίες. Ο τρόπος που οργανώνονται τα δεδομένα και η πληροφορία με αυτή τη μέθοδο καθιστά την πληροφορία εύκολα προσβάσιμη. Το πιο γνωστό ίσως παράδειγμα αυτής της εφαρμογής είναι το Web directory που υλοποίησε η Yahoo στα μέσα της δεκαετίας του ’90 και το Open Directory Project που φαίνεται στην παρακάτω εικόνα.



Εικόνα 13 - Open Directory Project

Όπως μπορούμε να καταλάβουμε, η συγκεκριμένη μέθοδος προσφέρει ένα σημαντικό πλεονέκτημα σε σχέση με το keyword search όπως και από οποιοδήποτε set ή ranked μοντέλο. Η λέξη “directory” προδίδει οργανωμένο περιεχόμενο, το οποίο προσφέρει στους χρήστες έναν οδηγό ως προς το τι μπορούν να αναζητήσουν ή τι ενδεχομένως να θελήσουν να αναζητήσουν. Παρόλο που φαίνεται εξαρχής ότι η πληροφορία που θα αναζητήσουν είναι προεπιλεγμένη, η πλοήγηση με τη χρήση web directories δίνει τη δυνατότητα στο χρήστη να κάνει προοδευτική αναζήτηση από φάκελο σε φάκελο μέχρι να βρει αυτό που πραγματικά ζητάει. Η ευκολία όμως αυτή δεν είναι απαλλαγμένη από ελαττώματα. Ο τρόπος με τον οποίο είναι οργανωμένοι οι φάκελοι και η πληροφορία μέσα σε αυτούς από τον ταξινόμο δεν είναι εξαρχής γνωστός. Μπορεί να υπάρχει μια γενική κατεύθυνση αλλά από τη στιγμή που μπαίνει ανθρώπινος παράγοντας, μπορεί να υπάρξουν περιπτώσεις όπου μια προφανής διαδρομή προς μια πληροφορία να μην είναι τόσο προφανής τελικά.

Το πώς πρέπει να οργανώνεται η πληροφορία είναι πολύ διαφορετικό στο μυαλό του κάθε ταξινόμου και φυσικά του χρήστη. Επί σειρά ετών οι επιστήμονες της πληροφορικής διαφωνούν ως προς την οργάνωση της πληροφορίας σε web directories. Όπως είδαμε και σε προηγούμενο κεφάλαιο, στο σύστημα DDS η

«γάτα» βρισκόταν και στην κατηγορία «τεχνολογία». Μια λύση που χρησιμοποιήθηκε είναι η παρουσίαση των εναλλακτικών διαδρομών και επιλογών ανά στοιχείο ώστε να αποφευχθεί η σύγχυση.

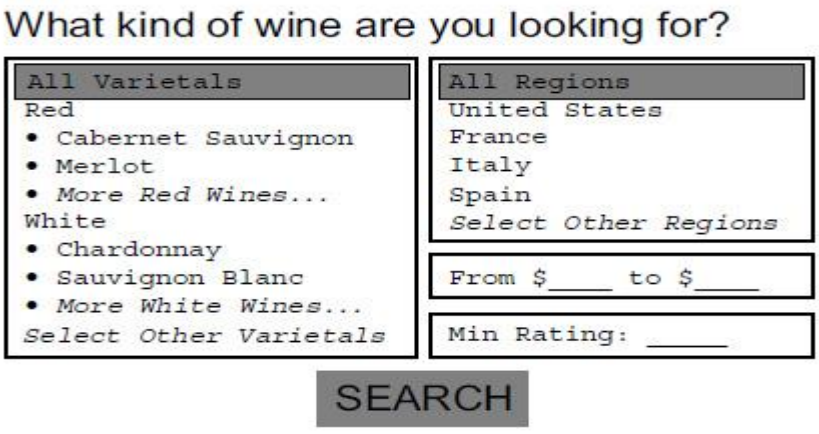
3.7. Faceted Information Retrieval

Στο παρόν κεφάλαιο μπορούμε να βάλουμε μαζί ό,τι μελετήσαμε στα προηγούμενα κεφάλαια. Αρχικά βλέπουμε ότι το faceted classification αντιμετωπίζει ορισμένους περιορισμούς όσον αφορά την αναπαράσταση της πληροφορίας. Παραπάνω είδαμε ότι οι λύσεις του search και directory navigation έχουν επίσης περιορισμούς. Επίσης, θα δούμε πως μπορούμε να δημιουργήσουμε συστήματα με καλύτερα interface με τη χρήση του faceted search. Πριν όμως τα δούμε όλα αυτά, θα μελετήσουμε πρώτα δύο προγόνους του faceted search, το parametric search και το faceted navigation.

3.7.1. Parametric search

Ως παράδειγμα σε αυτό το κεφάλαιο θα χρησιμοποιήσουμε ένα αντικείμενο πολύ δημοφιλές στους κύκλους των ερευνητών πάνω στο faceted search: το κρασί. Τα facets που συνήθως χρησιμοποιούνται, περιλαμβάνουν την ποικιλία του σταφυλιού, την παλαιότητα, την περιοχή, την κατηγορία, την τιμή κ.α. Το ερώτημα που προκύπτει είναι πως θα καταστήσουμε αυτό το σύνολο των facets έτοιμο προς χρήση, από κάποιον χρήστη ο οποίος ενδιαφέρεται να βρει κρασί που ταιριάζει στις προτιμήσεις του. Το περιβάλλον του parametric search συνδυάζει Boolean εκφράσεις πάνω σε facets. Ένα ερώτημα μπορεί να περιλαμβάνει τους τελεστές AND και OR. Τιμές που περιλαμβάνονται στο ίδιο facet χρησιμοποιούν το OR ενώ τιμές μεταξύ διαφορετικών facets χρησιμοποιούν το AND. Στη συνέχεια το σύστημα τυπώνει τα αποτελέσματα που ικανοποιούν το ερώτημα.

Ας δούμε λίγο ένα σενάριο χρήσης του parametric search. Έστω ότι υπάρχει κάποιος χρήστης που επιθυμεί να ψάξει κρασί κόκκινο από Γαλλία με rating 90 και τιμή το πολύ $10. Το σύστημα θα δημιουργήσει το παρακάτω ερώτημα: {Varietal: Red, Region: France, Rating: ≥90, Price: ≤$10}.



Εικόνα 14 - Παράδειγμα αναζήτησης κρασιού με τη χρήση του Faceted Search

Όπως γίνεται προφανές από αυτό το απλό παράδειγμα, διαφορετικά ειδή facets δημιουργούν διαφορετικούς συνδυασμούς ερωτημάτων. Για κάποια ιεραρχικά facets όπως το varietal ο χρήστης μπορεί να διαλέξει κάποιο «φύλλο» από το «δέντρο» των τιμών ή κάποιο «κόμβο». Επίσης μπορεί να βλέπει ταυτόχρονα και «φύλλα» και «κόμβους» που του δίνει μεγαλύτερη ευχέρεια κινήσεων. Για τις αριθμητικές τιμές, τις περισσότερες φορές ο χρήστης προτιμάει να ορίσει εύρος. Στο συγκεκριμένο σύστημα δεν υπάρχουν περιορισμοί σε αυτό. Σε άλλα συστήματα προβλέπουν τέτοιες περιπτώσεις και ορίζουν ένα «ταβάνι» (αναζήτηση κρασιού από κάποια τιμή και πάνω ίσως είναι άσκοπο). Ζητήματα που αφορούν το design τέτοιων συστημάτων θα αναλυθούν λεπτομερέστερα παρακάτω. Παρόλα αυτά θα σταθούμε σε μερικά τώρα που χρήζουν αναφοράς. Το parametric search υλοποιεί το set retrieval μοντέλο. Ως αποτέλεσμα οι χρήστες δυσκολεύονται στη δημιουργία ερωτημάτων. Αυτό καθιστά το σύστημα ως no-user friendly το οποίο ίσως αποτρέψει το χρήστη από το να το ξαναχρησιμοποιήσει. Επίσης ελλοχεύει ο κίνδυνος του ”a million or none” όπου με ένα ασαφές ερώτημα έχουμε άπειρα άσχετα εντελώς αποτελέσματα ή με ένα πολύ αυστηρό ίσως δε βρούμε και τίποτα.

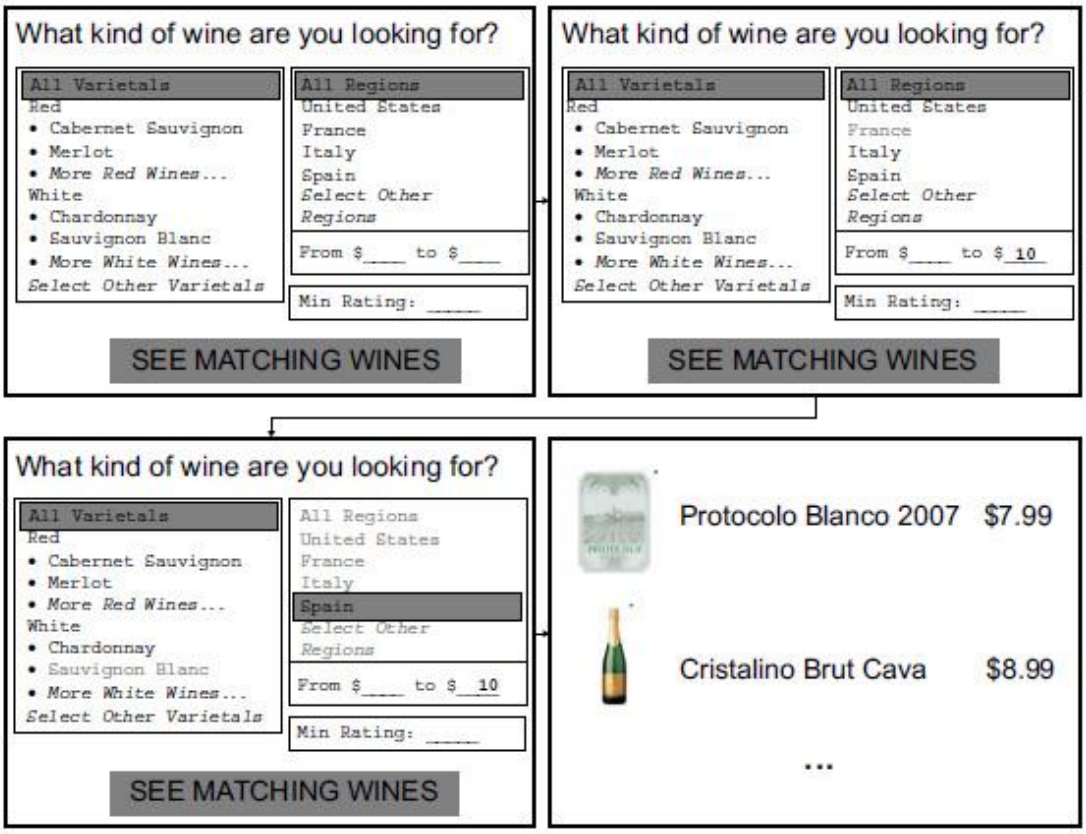
Συνοψίζοντας, το Parametric search ενώ προσφέρει ευχέρεια, στον αντίποδα δε βοηθάει και δεν κατευθύνει το χρήστη ως προς τη δημιουργία ερωτημάτων.

3.7.2. Faceted Navigation

Το faceted navigation έρχεται να καλύψει το κομμάτι που λείπει από το parametric search: Την καθοδήγηση του χρήστη. Όπως είδαμε στο parametric search, ο χρήστης πρέπει μονομιάς να προσπελάσει τα facets που τον ενδιαφέρουν, ενώ το faceted navigation επιτρέπει στον χρήστη να προσπελαύνει προοδευτικά τα facets βλέποντας το αποτέλεσμα κάθε επιλογής επί τόπου. Για να καταλάβουμε καλύτερα τη διαφορά θα χρησιμοποιήσουμε το παράδειγμα με το κρασί. Ο χρήστης θα μπορούσε να ξεκινήσει π.χ. με το όριο των $10 στην τιμή του μπουκαλιού. Αυτή η επιλογή αυτόματα αφαιρεί από τις επιλογές όσα κρασιά είναι ακριβότερα από $10 το μπουκάλι. Για παράδειγμα, αν δεν υπάρχουν Γαλλικά κρασιά κάτω από $10 με το Parametric Search θα οδηγούμασταν κατευθείαν σε μηδενικό αποτέλεσμα ενώ τώρα έχουμε μια οπτική του τι γίνεται στην πραγματικότητα με τη βοήθεια των διαθέσιμων facets. Με αυτό τον τρόπο μπορούμε να συνεχίσουμε την πλοήγηση στα facets αντί να ξεκινήσουμε αναζήτηση εκ νέου.

Μέσα από την προοδευτική προσέγγιση του faceted navigation, δίνεται η δυνατότητα στους χρήστες να εξαλείψουν τα «αδιέξοδα». Έτσι, στην πλειοψηφία των περιπτώσεων τα αποτελέσματα ικανοποιούν τις αναζητήσεις του χρήστη. Παράλληλα, επιλύεται και το πρόβλημα του “million or none” του parametric search. Παρόλα αυτά, είναι φορές που ορισμένες τιμές οδηγούν σε «αδιέξοδα». Π.χ. αν επιλέξουμε κρασιά κάτω από $1 θα έχουμε μηδενικά αποτελέσματα. Ορισμένα συστήματα κάνουν ελέγχους στις τιμές ώστε να αποφευχθούν τέτοιες επιλογές. Γενικότερα το faceted navigation έχει αποδοτική εφαρμογή σε facets που έχουν μια λογική σειρά. Τι γίνεται όμως στην περίπτωση που έχουμε και κείμενο προς

αναζήτηση; Το ερώτημα αυτό μας οδηγεί στο επόμενο κεφάλαιο και το σκοπό αυτής της πτυχιακής: Το Faceted Search.

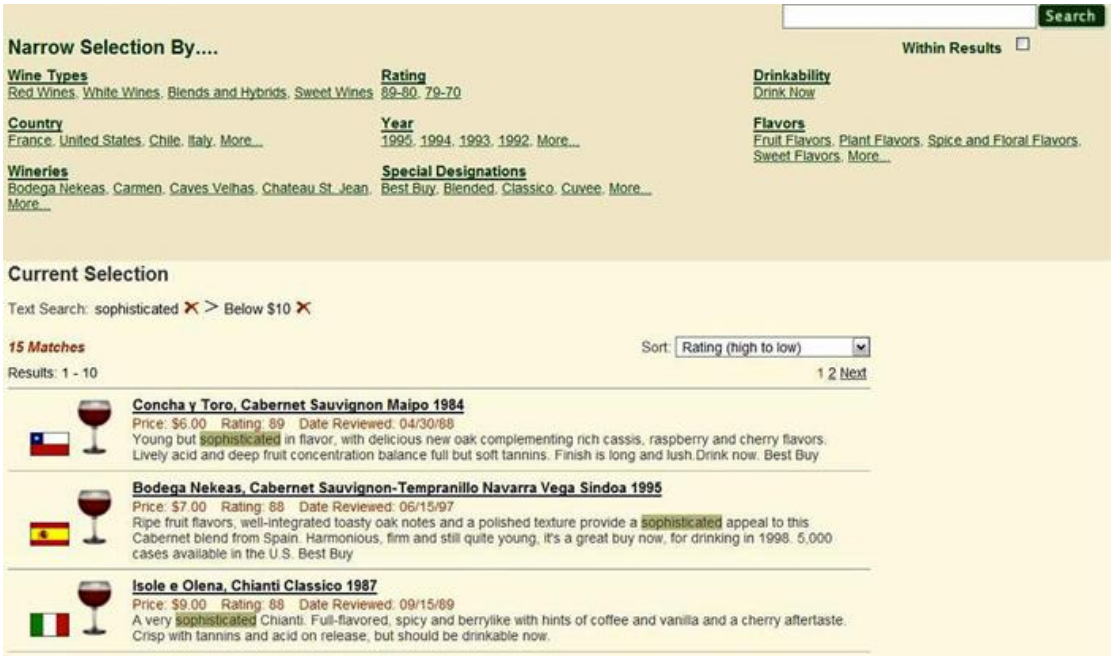


Εικόνα 15 - Παράδειγμα αναζήτησης κρασιού με τη χρήση του Faceted Search II

3.7.3. Faceted Search

Επιτέλους φτάνουμε στο faceted search, το λόγο που γράφεται αυτή η πτυχιακή. Ο στόχος αυτής της παραγράφου είναι να προσφέρει μια βασική σε πρώτη φάση ενημέρωση για το faceted search που θα χρειαστούμε στα πιο ειδικά θέματα στη συνέχεια. Στο προηγούμενο κεφάλαιο μελετήσαμε information retrieval τεχνικές και αναλύσαμε τις διαφορές του parametric search από το faceted navigation. Να αναφέρουμε εδώ, ότι στις περισσότερες περιπτώσεις θα έχουμε να κάνουμε με συλλογές ημιδομημένων εγγράφων κειμένου (σε αντίθεση με τα δομημένα και μη-δομημένα που αναλύσαμε στο προηγούμενο κεφάλαιο), τα οποία περιέχουν συνδυασμούς από δομημένα δεδομένα, που ονομάζονται metadata. Το meta προέρχεται από την ελληνική λέξη «μετά» που σημαίνει μαζί αλλά στην περίπτωση μας έχει την έννοια του «σχετικά». Όντως τα meta data ενός εγγράφου αποτελούνται από χαρακτηριστικές λέξεις και ορισμούς που βρίσκονται διάσπαρτα μέσα στο κείμενο και το χαρακτηρίζουν. Όταν λοιπόν εφαρμόσουμε faceted search τεχνικές σε αυτά τα έγγραφα μπορούμε να συνδυάσουμε απλή αναζήτηση στο μη δομημένο κείμενο και faceted navigation στο δομημένο περιεχόμενο. Αυτή η λογική είναι ο πυρήνας του faceted search. Ας επιστρέψουμε στο παράδειγμα του κρασιού. Αυτή τη φορά όμως θα παρουσιάσουμε ένα πιο πλούσιο μοντέλο, όσον αφορά το περιεχόμενο, το οποίο περιλαμβάνει πλέον και κείμενο περιγραφής για κάθε κρασί.

Στην παρακάτω εικόνα μπορούμε να δούμε πως το faceted search συνδυάζει αναζήτηση κειμένου με facets.



Εικόνα 16 - Συνδυασμός αναζήτησης facets και απλού κειμένου

Στο παράδειγμα που είχαμε δει στην αρχή της πτυχιακής, με τη χρήση της ιστοσελίδας www.plaisio.gr, είχαμε πάρει μια πρώτη ιδέα του πως λειτουργεί μια ιστοσελίδα που χρησιμοποιεί το faceted search. Στο τωρινό παράδειγμα, πάμε ένα βήμα παραπέρα και βλέπουμε πως συνδυάζεται και η αναζήτηση κειμένου παράλληλα με τα facets. Στο παράδειγμα λοιπόν με το κρασί θα ξεκινήσουμε την αναζήτηση κάνοντας αναζήτηση απλού κειμένου και συγκεκριμένα θα χρησιμοποιήσουμε τη λέξη “sophisticated” στην περιγραφή του. Από αυτά τα αποτελέσματα χρησιμοποιώντας το facet τιμή θα τα περιορίσουμε στα κρασιά που έχουν τιμή κάτω από $10. Η αναζήτηση επιστρέφει 15 αποτελέσματα που ικανοποιούν αυτές τις συνθήκες ταξινομημένα ανά rating. Ο χρήστης μπορεί να ορίσει περισσότερα φίλτρα επιλέγοντας τιμές από επιπλέον facets.

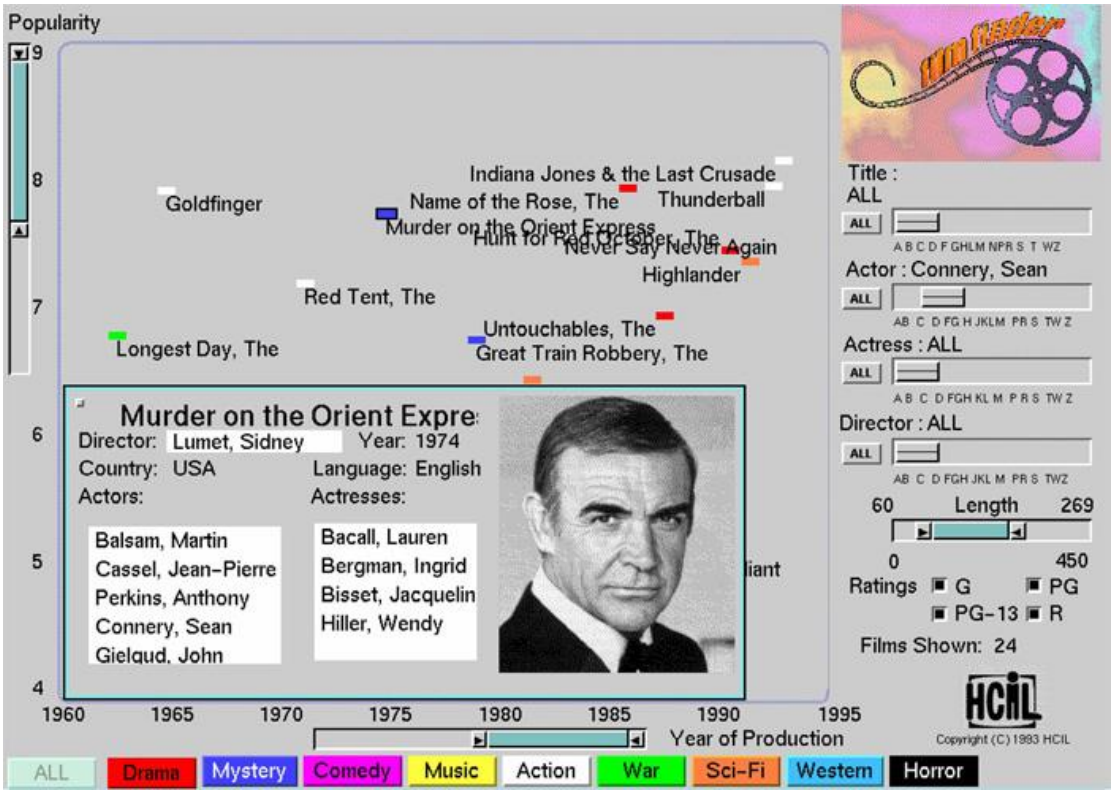
Όπως το faceted navigation έτσι και το faceted search απαλείφει τιμές facets που θα οδηγήσουν σε μηδενικό αποτέλεσμα. Για παράδειγμα, rating μεταξύ 90-100 δεν είναι διαθέσιμο σε αυτές τις τιμές “sophisticated” κρασιού που επιλέξαμε. Αυτό το παράδειγμα συνοψίζει την λειτουργία του faceted search. Αυτό είναι όμως όλο; Ίσως αναρωτιέστε τώρα αν ολόκληρη η πτυχιακή αναλύει τη συγκεκριμένη λειτουργία. Η πραγματικότητα όμως είναι πολύ διαφορετική. Θα μπορούσαμε να παρομοιάσουμε το faceted search με το σκάκι. Είναι πολύ εύκολο και γρήγορο να καταλάβει κάποιος τους κανόνες αλλά θα χρειαστεί χρόνια μελέτης για να κατανοήσει πλήρως τις δυνατότητες του. Το επόμενο κομμάτι της πτυχιακής αναλύει περισσότερες λεπτομέρειες καθώς και διάφορες υλοποιήσεις επιστημόνων, ενώ αργότερα θα ασχοληθούμε με την υλοποίηση του faceted search όταν κατασκευάζουμε μια ιστοσελίδα, τα προβλήματα που παρουσιάζονται και τα σημεία που πρέπει να προσεχθούν.

3.8. Ακαδημαϊκή έρευνα πάνω στο Faceted

Στο προηγούμενο κεφάλαιο ακολουθήσαμε μια λογική σειρά μέχρι να φτάσουμε στο faceted search, αναλύοντας τεχνολογίες και ιστορικά γεγονότα. Πλέον θα αναλύσουμε το faceted search εις βάθος μέσα και από κάποιες ακαδημαϊκές μελέτες. Αυτές οι μελέτες από μόνες τους και πάλι δε θα είναι αρκετές για να καλύψουν όλο το φάσμα.

3.8.1. Dynamic Queries και Query Previews

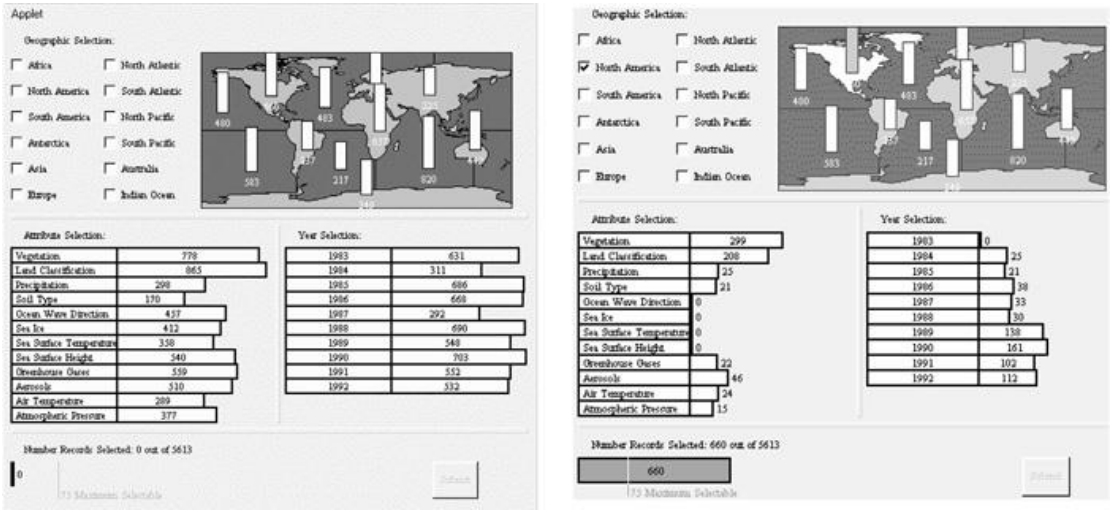
Είναι λίγο δύσκολο να βεβαιωθούμε για το πότε ακριβώς έλαβε τόπο η πρώτη επίσημη υλοποίηση με faceted navigation. Δύο υποψήφιες υλοποιήσεις είναι τα Dynamic queries και query previews, που μελετήθηκαν στο πανεπιστήμιο του Maryland, όπως επίσης και μια υλοποίηση του view-based search στο πανεπιστήμιο του Huddsfield. Κανένα από αυτά τα συστήματα δεν αναφέρει στις προδιαγραφές του τα facets. Ωστόσο αυτές ήταν οι προσπάθειες που, στα μέσα του 1990, έπαιξαν καταλυτικό ρόλο στη μορφή του faceted search σήμερα. Τα dynamic queries επιτρέπουν στον χρήση να έχει μια οπτική απεικόνιση αποτελεσμάτων μέσω ενός αλληλεπιδραστικού συστημάτων εισαγωγής ερωτημάτων. Ένα τέτοιο σύστημα είναι ιδιαίτερα ταχύ στην επεξεργασία αποτελεσμάτων, ειδικά όταν το συγκρίνουμε με υλοποιήσεις που χρησιμοποιούν SQL και βάσεις δεδομένων.



Εικόνα 17 - FilmFinder

Η παραπάνω εικόνα μας δείχνει το σύστημα FilmFinder, το οποίο δημιουργήθηκε από τους Shneiderman και Christopher Ahlberg και επιτρέπει αναζήτηση σε μια βάση δεδομένων για ταινίες. Η γραφική του σχεδίαση προσφέρει πολλές

δυνατότητες αλληλεπίδρασης, όπως parametric search πάνω από κάποια facets. Αυτά τα δυναμικά ερωτήματα που δημιουργούνται επιστρέφουν πολύ γρήγορα αποτελέσματα και αμβλύνουν λίγο το μειονέκτημα της δυσκολίας του parametric search όσον αφορά τη διατύπωση ερωτημάτων. Επιπλέον, τα οπτικά user-interface είναι πολύ φιλικότερα στο μέσο χρήστη από ότι αυτά που χρησιμοποιούν γραμμή εντολών. Όπως και να χει όμως, εξακολουθούν να υπάρχουν συνδυασμοί που επιστρέφουν μηδενικά αποτελέσματα. Ο Shneiderman μαζί με τους Khoa Doan και Catherine Plaisant αντιμετώπισαν το πρόβλημα με τη χρήση query previews. Η συγκεκριμένη μέθοδος αποτρέπει τα περιττά βήματα που θα καταλήξουν σε μηδενικό αποτέλεσμα. Με λίγα λόγια κάθε query preview αντικαθιστά το parametric search με faceted navigation.



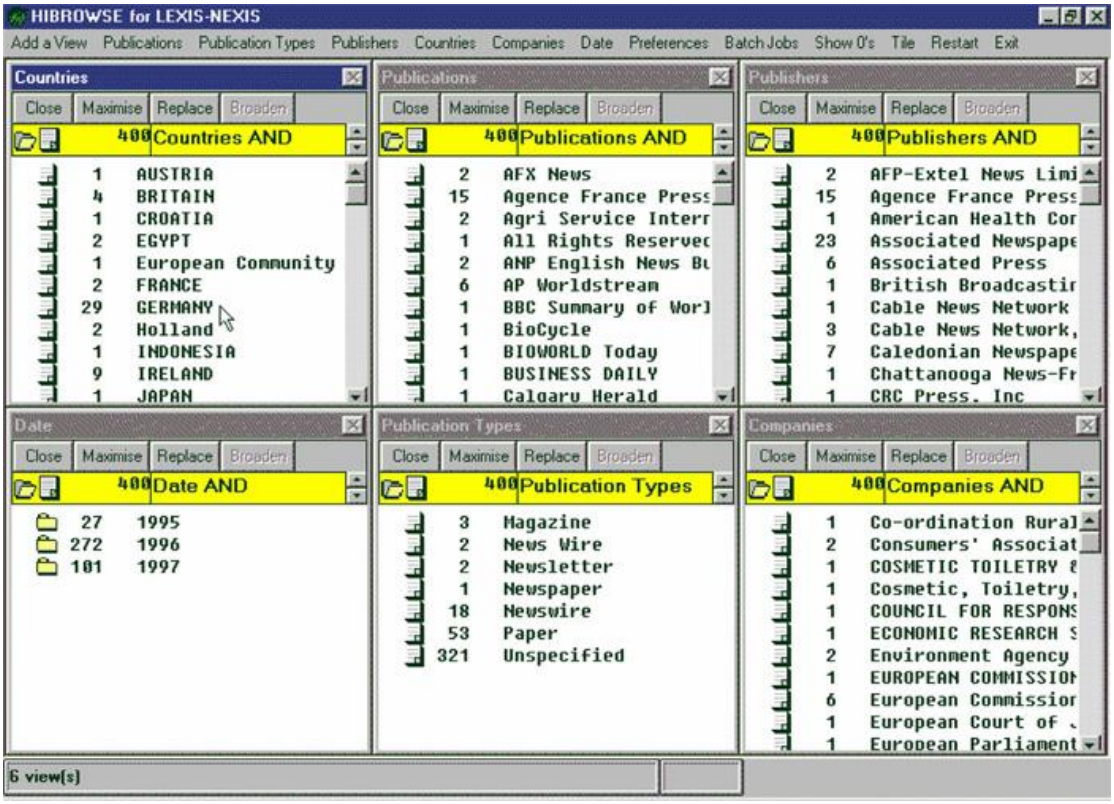
Εικόνα 18 - Faceted Search εφαρμογή στη NASA

Η παραπάνω εικόνα μας δείχνει πως τα query previews χρησιμοποιούνται μέσα από το faceted navigation για συλλογή πληροφοριών από την NASA. Το interface του συστήματος αυτού είναι χωρισμένο σε δύο στήλες. Η αριστερή παρουσιάζει το πριν και η δεξιά δείχνει το μετά. Η επιλογή της βόρειας Αμερικής από το facet "Geography" απαλείφει τις τιμές "Sea Ice" από το facet "attribute" και "1983" από το facet "year". Επιπλέον, με το δυναμικά ανανεώσιμο περιεχόμενο του επιτρέπει στους χρήστες να έχει μια γενική εικόνα ανά πάσα στιγμή από την συλλογή των εγγράφων που έχει φιλτράρει.

3.8.2. View-based search

Ο Steven Pollit, στο Πανεπιστήμιο του Huddersfield, ξεκίνησε μόνος του ένα project πάνω στο faceted navigation. Ο ίδιος παρόλα αυτά δεν το περιέγραφε ως τέτοιο. Αντιθέτως το ονόμασε View-based search. Με τη χρήση αυτού του συστήματος δινόταν η δυνατότητα στους χρήστες, που χρησιμοποιούσαν information retrieval συστήματα, να κάνουν αποδοτική χρήση βάσεων δεδομένων χωρίς τη βοήθεια ενδιάμεσων μεθόδων αναζήτησης. Παρουσίαζαν το σύστημα αυτό ως μια προσέγγιση με τη βοήθεια της οποίας ο χρήστης θα είχε την ευκαιρία να αναζητήσει σε μια βάση δεδομένων με υπέρ-αποδοτικούς τρόπους, καθώς και μέσω των όψεων που προσέφερε η sql να παρουσιάζουν πολλά περιεχόμενα ταυτόχρονα. Επιπλέον,

θα μπορούσαν να κάνουν συγκρίσεις και να κατέληγαν ευκολότερα εκεί που ήθελαν. Αν θέλαμε εδώ να κάνουμε ένα συσχετισμό, θα μπορούσαμε να πούμε ότι τα views στο σύστημα του Pollitt είναι αυτά που ονομάζουμε facets. Στην πραγματικότητα, η view-based αναζήτηση δεν είναι καν αναζήτηση, αλλά μια εκτέλεση faceted navigation. Ο Pollit ίσως και να παρουσίασε την πρώτη δουλειά που βασίστηκε σε hierarchical faceted navigation.



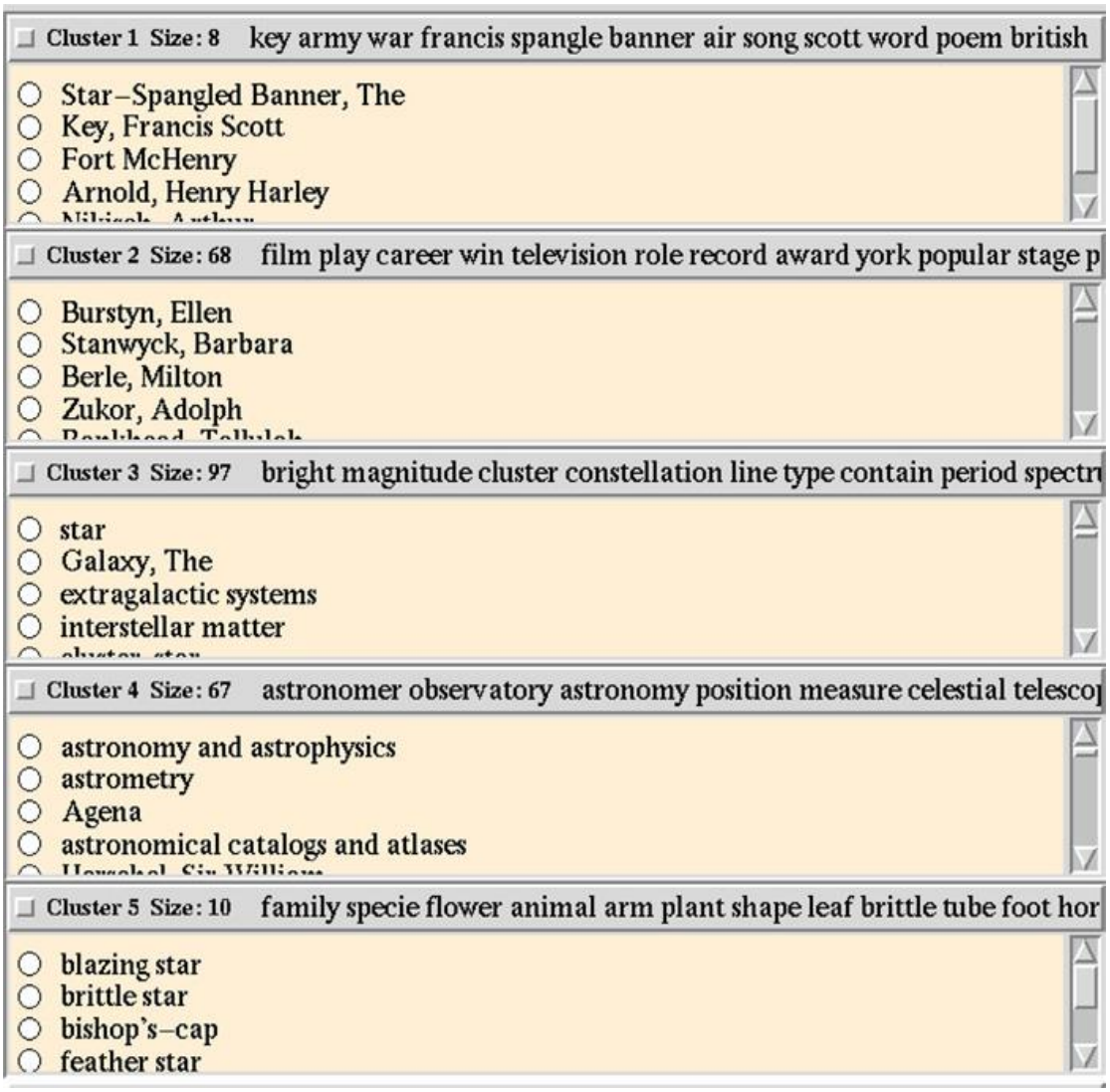
Εικόνα 19 - HIBROWSE

Η εικόνα μας δείχνει το HIBROWSE, μια υλοποίηση του view-based search από τον Pollitt, που συλλέγει έγγραφα από το Lexis-Nexis. Και το view-based search και τα query previews έχουν δύο κοινούς περιορισμούς. Ο πρώτος είναι ότι μπορεί να υποστηρίζουν faceted navigation αλλά όχι faceted search. Η έλλειψη της δυνατότητας αναζήτησης με την εισαγωγή απλού κειμένου αποθαρρύνει τους χρήστες, που ειδικά τη σημερινή εποχή, χρησιμοποιούν αυτή τη μέθοδο πάρα πολύ. Ο δεύτερος περιορισμός είναι ότι παρουσιάζει στο χρήστη ή κάποια προεπιλεγμένα facets ή facets τα οποία επιλέχθηκαν ρητά από τους χρήστες. Το σύστημα δυστυχώς δεν προσφέρει κανέναν αυτοματισμό στην παρουσίαση των ερωτημάτων.

3.8.3. Flamenco Project

Ο άνθρωπος που σχετίζεται περισσότερο με το faceted search είναι ο Marti Hearst. Στα μέσα του 1990 κατασκεύασε το σύστημα Scatter/Gather, ένα σύστημα που χρησιμοποιεί clusters για την περιήγηση ανάμεσα σε μεγάλες συλλογές εγγράφων. Τα έγγραφα χωρίζονται σε clusters και στο χρήστη παρουσιάζεται μια περίληψη του κειμένου. Η σχέση αυτής της μεθόδου με το faceted search είναι ότι έχει ίδιες

επιλογές στο interface που χρησιμοποιούν. Και τα δυο χρησιμοποιούν αλληλεπίδραση στις συλλογές των δεδομένων καθώς και φόρμα με query previews.

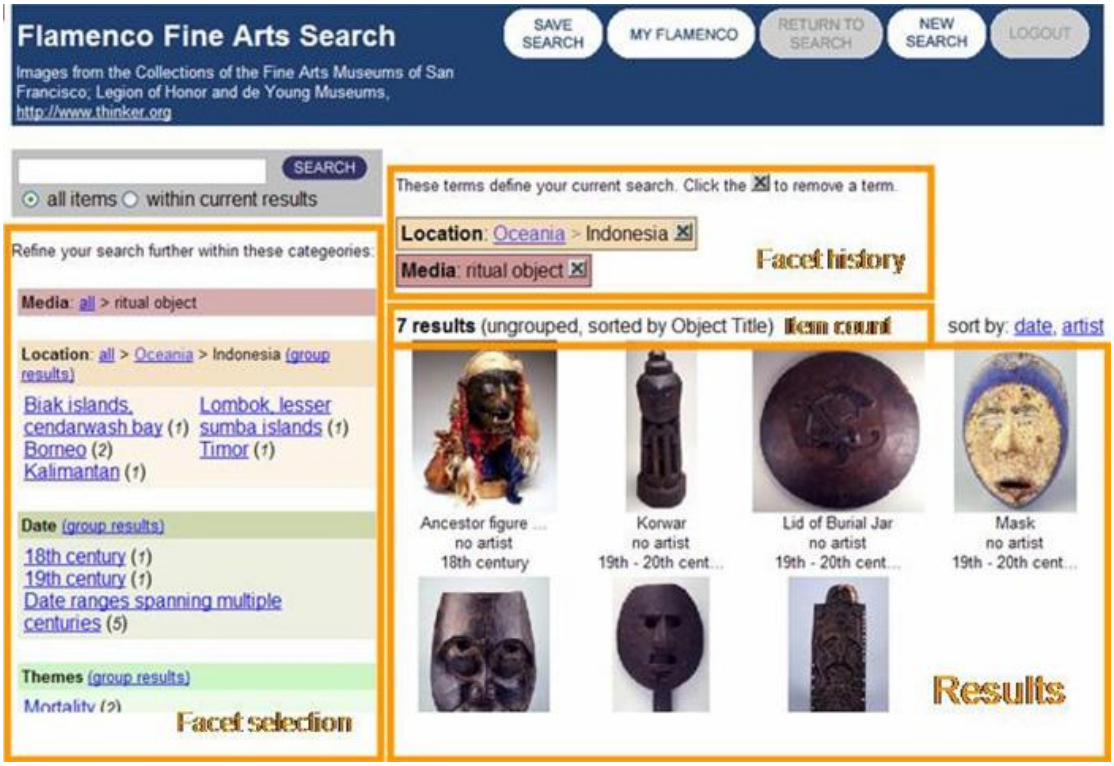


Εικόνα 20 - Scatter/Gather

Ωστόσο το Scatter/Gather προϋποθέτει ότι τα έγγραφα θα περιέχουν μη δομημένο κείμενο.

Μεταγενέστερα του Scatter/Gather ο Hearst, μέσω του Πανεπιστημίου Berkeley, πήρε μέρος σε ένα project που ήταν επικεντρωμένο πλήρως στο faceted search. Πρόκειται για το FLexible information Access using MEtadata in Novel Combinations γνωστό και ως Flamenco. Είναι ένα project που αναπτυσσόταν για πάνω από μια δεκαετία υλοποιώντας εργαλεία για faceted search, ενώ παράλληλα γινόταν έρευνες χρηστικότητας του. Η κεντρική φιλοσοφία του είναι η κατασκευή ενός open-source faceted search συστήματος με hierarchical facets. Επιπλέον, το σύστημα περιλάμβανε μελετημένο interface, το οποίο αυτοματοποιούσε την δημιουργία metadata. Συν τοις άλλοις, ο Hearst έκανε και μια σύγκριση μεταξύ του Flamenco και του Scatter/Gather.

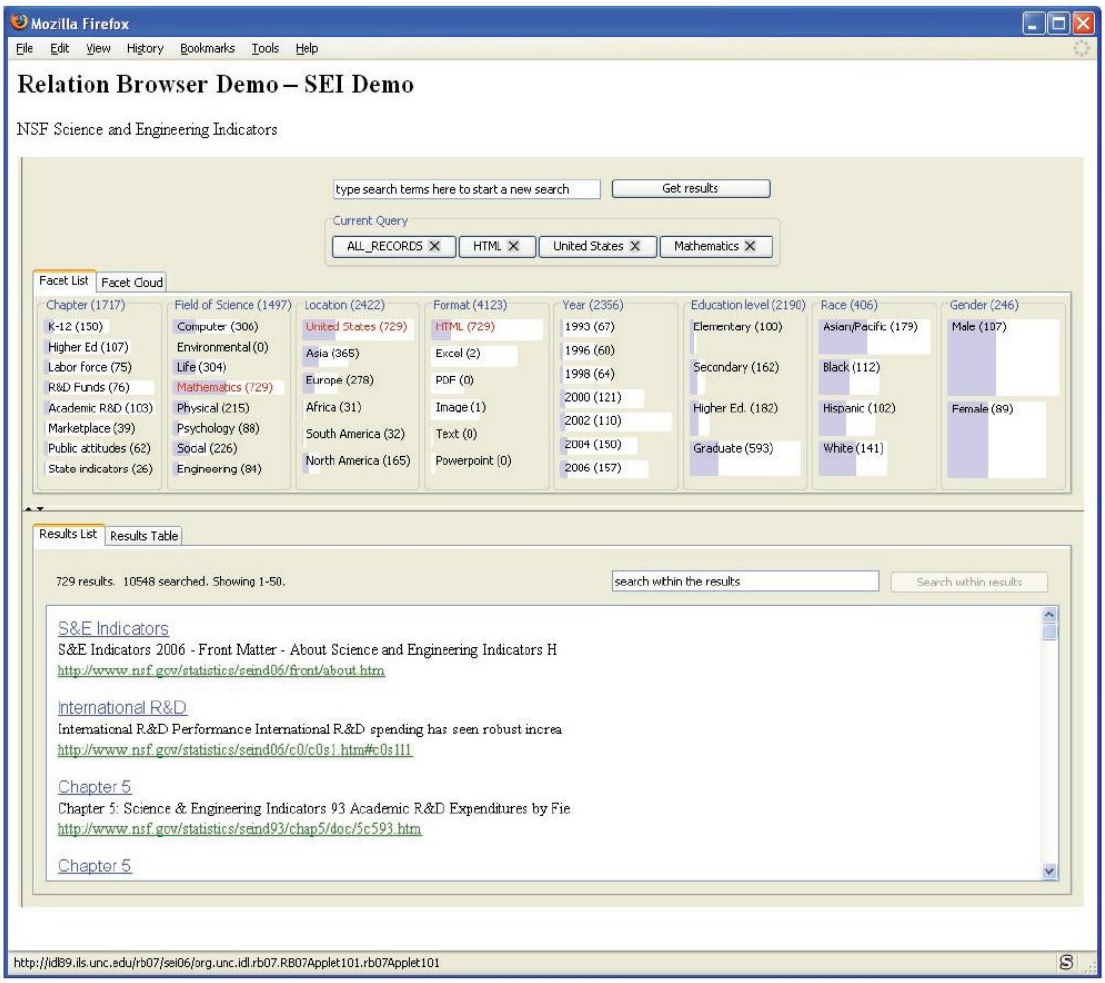
Η παρακάτω εικόνα μας δείχνει ένα παράδειγμα μιας εφαρμογής Flamenco: Ένα faceted search σύστημα που περιλαμβάνει συλλογές εικόνων εκθεμάτων από διάφορα καλλιτεχνικά μουσεία του San Francisco.



Εικόνα 21 - Flamenco Project

3.8.4. Relation Browser

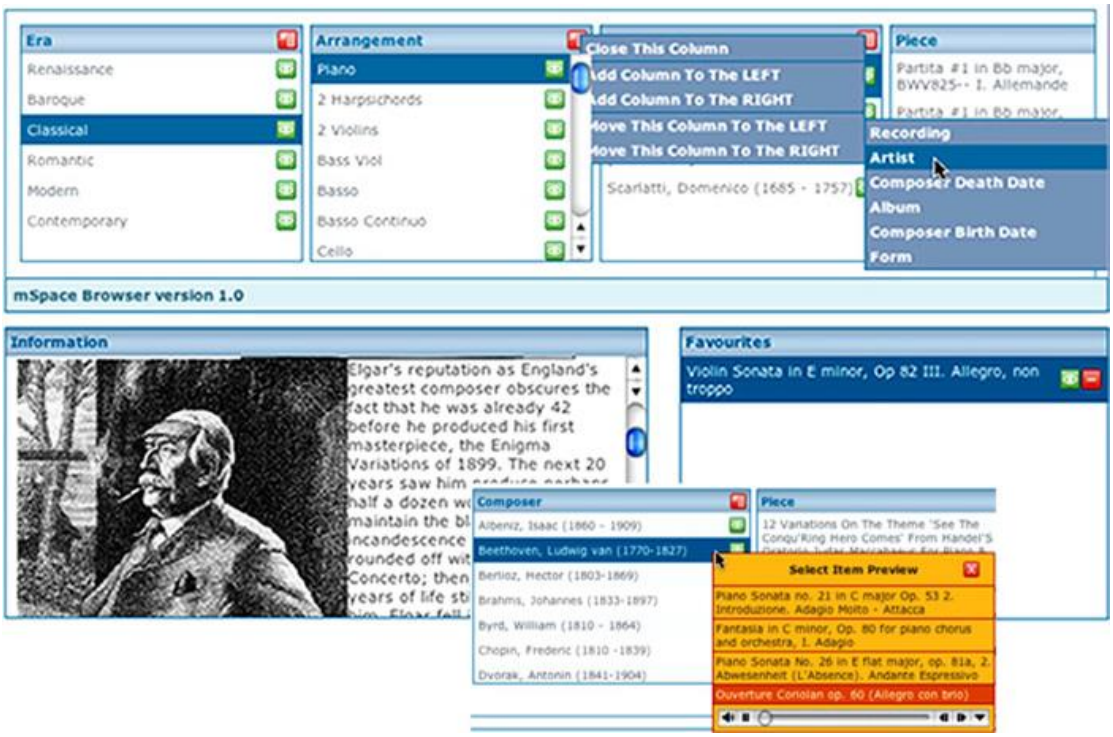
Παράλληλα με τον Hearst και το Flamenco Project, στο Πανεπιστήμιο της North Carolina, ο Gary Marchiolini έκανε τη δικιά του προσπάθεια με ένα project που ονομάστηκε Relation Browser. Το συγκεκριμένο project, που αρχικά δημιουργήθηκε για λογαριασμό του US Bureau of Labor Statistics, στόχευε να βελτιώσει τους ήδη υπάρχοντες μηχανισμούς αναζήτησης και περιήγησης, προσφέροντας ένα σύστημα που έκανε χρήση previews. Σε αντίθεση με το Flamenco, το Relation Browser είχε την καινοτομία της γρήγορης περιήγησης σε ένα κείμενο απλά περνώντας το ποντίκι πάνω από την επιθυμητή πληροφορία. Επίσης, με το mouse-over του ποντικιού δημιουργούνταν δυναμικά ερωτήματα. Στην παρακάτω εικόνα βλέπουμε το interface του RB++, την πιο πρόσφατη έκδοση του Relation Browser. Τα χαρακτηριστικά του περιλάμβαναν τη χρήση γραφικών στηλών και μπαρών για την απεικόνιση των hits σε κάποια τιμή ενός facet.



Εικόνα 22 - Relation Browser demo

3.8.5. mSpace

Ο ερευνητής του Πανεπιστημίου του Southampton mc shraefel (ναι το όνομα του είναι case-sensitive) υλοποίησε το mSpace project, το οποίο το περιγράφει ως μια αλληλεπιδραστική σχεδίαση, που υποστηρίζει μεταβαλλόμενο περιεχόμενο ορισμένο από τον χρήστη και υλοποιεί τρείς τεχνικές που υποστηρίζουν αυτήν την σχεδίαση: preview cues, dimensional sorting και spatial context. Φυσικά οι στόχοι του mSpace έχουν πολλά κοινά με τα προηγούμενα project που αναφέραμε. Παρόλα αυτά, το mSpace δίνει έμφαση περισσότερο στη σχεδίαση του interface και στη φιλικότητα προς το χρήστη.

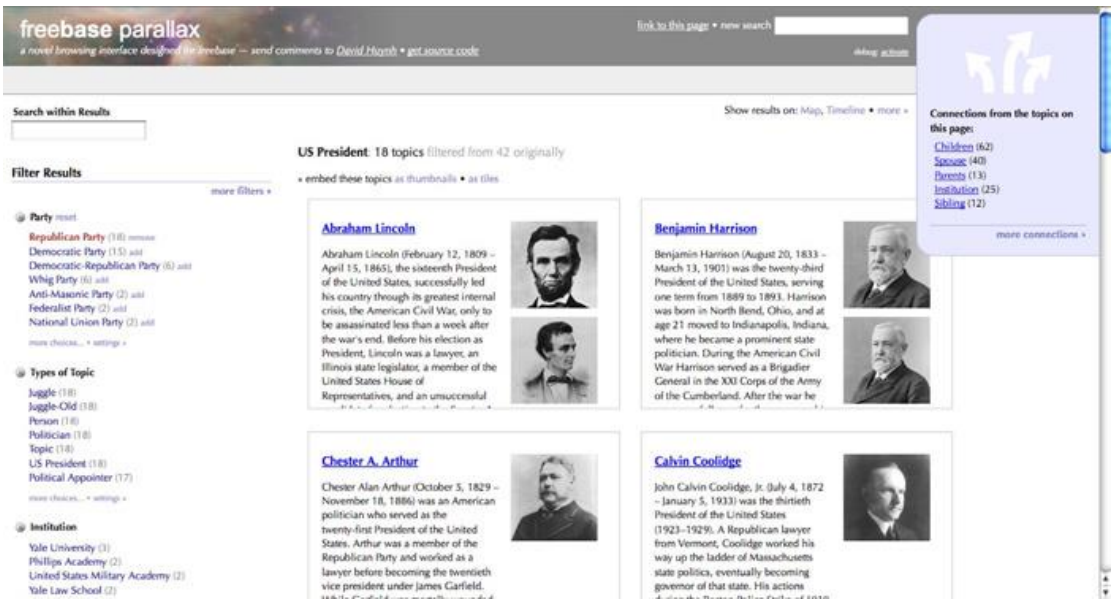


Εικόνα 23 - mSpace

3.8.6. Parallax

Το τελευταίο project που θα αναφέρουμε είναι το Parallax. Ο David Huynh ανέπτυξε το Parallax στο MIT, σαν ένα κομμάτι του SIMILE project, αλλά τελικά συνέχισε την δουλειά στην Metaweb, όπου και το συμπεριέλαβε στο interface του Freebase. Το Freebase είναι ένα σύστημα που διατηρούσε βάση δεδομένων με δομημένα στοιχεία. Το interface του Parallax προσφέρει set-based browsing και επεκτείνει το faceted search, ώστε να προσφέρει εναλλασσόμενες οπτικές μεταξύ των οντοτήτων.

Καθώς ο χρήστης παρατηρεί ορισμένα σετ από αποτελέσματα, το Parallax όχι μόνο προσφέρει φίλτρα που βασίζονται σε facets που σχετίζονται με τα αποτελέσματα, αλλά χρησιμοποιεί και συστήματα που σχετίζουν αυτά τα facets μεταξύ τους, παρουσιάζοντας επιπλέον σετ αποτελεσμάτων. Όπως και το mSpace έτσι και το Parallax δίνει μεγάλη έμφαση στο user interface και στη σχεδίαση. Αυστηρά μιλώντας, το Parallax δεν είναι ένα faceted search σύστημα αλλά υλοποιεί semantic web τεχνικές. Αυτό του δίνει τη δυνατότητα να παρέχει ποικίλα σετ σχέσεων μεταξύ των οντοτήτων. Το Parallax είναι εντέλει ένα πολύ μεγάλο βήμα προς την semantic web εποχή, χρησιμοποιώντας πολλές από τις τεχνικές που έκαναν το faceted search επιτυχημένο.



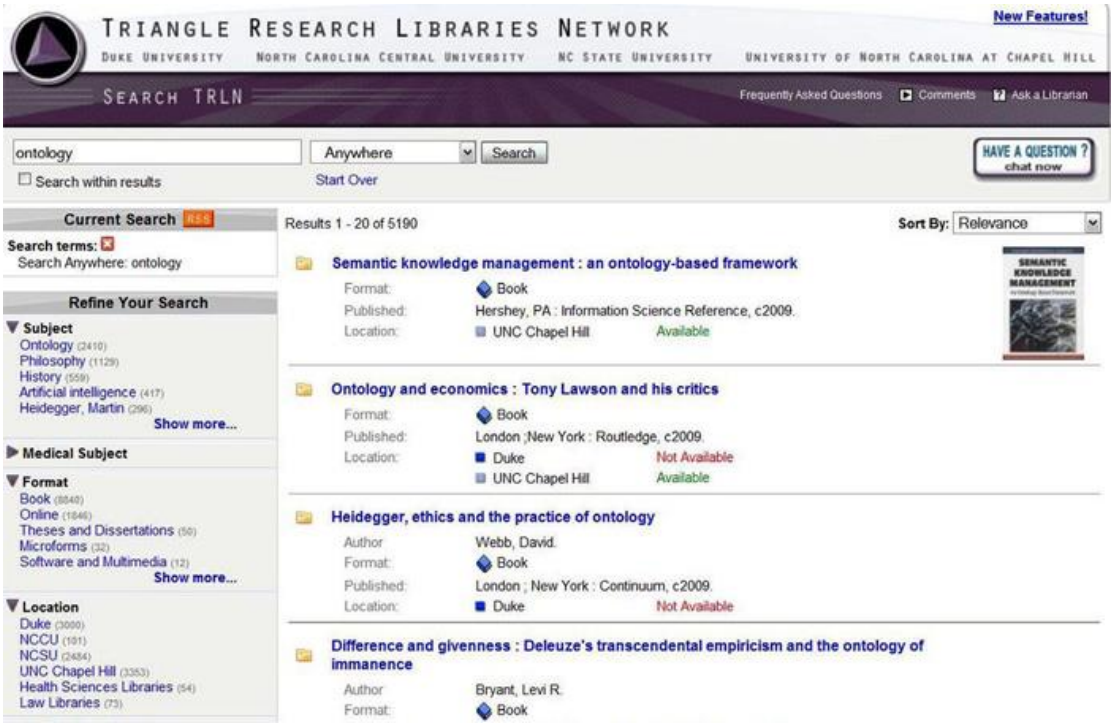
Εικόνα 24 - Parallax

3.9. Εμπορικές εφαρμογές

Πολλά από τα ακαδημαϊκά project που είδαμε στο προηγούμενο κεφάλαιο μας οδήγησαν στο να καταλάβουμε καλύτερα το faceted search και τις δυνατότητες του, ώστε να καταφέρουμε να ξεφύγουμε από τους περιορισμούς που έθεταν οι προηγούμενες εφαρμογές όσον αφορά το information retrieval. Η μεγαλύτερη απόδειξη επιτυχίας όμως του faceted search είναι αναμφίβολα η ευρεία απήχηση που έχει στις εμπορικές εφαρμογές. Από τις προηγούμενες δεκαετίες μέχρι και σήμερα, το faceted search ξεκίνησε ως ένα σύνολο εσωτερικών project στα πανεπιστήμια και κατέληξε να είναι η νούμερο ένα προτίμηση στις σημερινές εμπορικές εφαρμογές. Παρακάτω θα αναλύσουμε μερικές από αυτές.

3.9.1. Endeca

Η Endeca δημιουργήθηκε το 1999 με την προοπτική να προσφέρει faceted search υπηρεσίες, με την ονομασία Guided Navigation, σε μεγάλες εταιρίες. Είναι περισσότερο γνωστή για την προσφορά υπηρεσιών σε εφαρμογές ηλεκτρονικού εμπορίου, ενώ παράλληλα παρείχε υπηρεσίες τέτοιου τύπου και σε άλλες εταιρίες. Ουσιαστικά η εταιρία αυτή καθιέρωσε το faceted search ως λύση για τις εμπορικές εφαρμογές και για τα websites που προσφέρουν δυνατότητες ηλεκτρονικού εμπορίου. Στην παρακάτω εικόνα, βλέπουμε ένα παράδειγμα εφαρμογής που επιμελήθηκε η εν λόγω εταιρία. Πρόκειται για το interface αναζήτησης της Triangle Research Libraries Network, το οποίο αναπτύχθηκε στο Πανεπιστήμιο της βόρειας Carolina. Το interface αυτό, βασισμένο στο faceted search, επιτρέπει στον χρήστη να περιηγηθεί σε έναν ημι-δομημένο κατάλογο, ο οποίος συσχετίζει κάθε βιβλίο με faceted μετά-δεδομένα και περιγραφικό κείμενο.



Εικόνα 25 - Υλοποίηση Faceted Search εφαρμογής Endeca

3.9.2. eBay

Το ebay δημιουργήθηκε το 1996 και θεωρείται σήμερα η μεγαλύτερη αγορά παγκοσμίως. Παρόλο που προσφέρει και άλλες υπηρεσίες είναι περισσότερο γνωστό σαν μια ιστοσελίδα πραγματοποίησης δημοπρασιών.

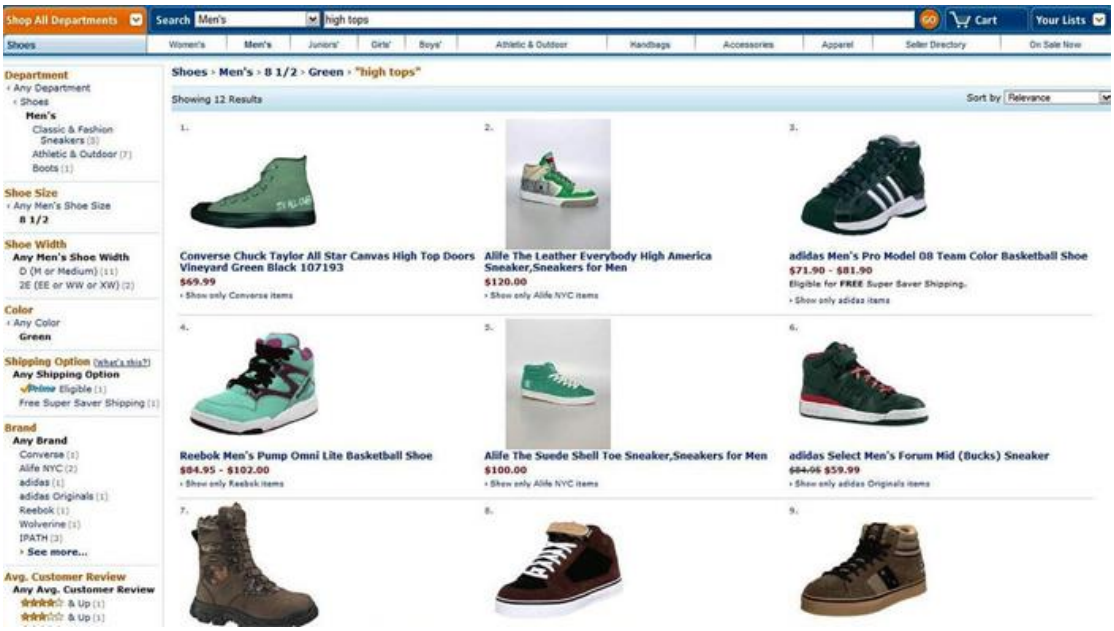
Εικόνα 26 - eBay

Το 2006 το ebay συμπεριέλαβε στην ιστοσελίδα του ένα project, το οποίο πρόσφερε μια τυπική εφαρμογή αγοροπωλησίας που δεν είχε σχέση με δημοπρασίες, αλλά

κανονικές αγορές. Το project αυτό δεν ευδοκίμησε, αλλά τα faceted search χαρακτηριστικά του υιοθετήθηκαν από το κυρίως website.

3.9.3. Amazon

Ιδρύθηκε το 1994 και θεωρείται το μεγαλύτερο ηλεκτρονικό κατάστημα βιβλίων, παρόλο που σήμερα έχει στη γκάμα του μεγάλη ποικιλία προϊόντων. Το 2002 αποκαλύφθηκε το project "Ruby", ένα πειραματικό faceted search website το οποίο λειτουργούσε σαν «πολυκατάστημα καταστημάτων». Αργότερα το Amazon υλοποίησε το faceted search σε όλο το φάσμα των προϊόντων του.



Εικόνα 27 - Amazon

3.9.4. Open source projects

Παρόλο που οι εμπορικές υλοποιήσεις ήταν οι πρώτες που ώθησαν την εξέλιξη και τη φήμη του faceted search, δεν πέρασε πολύς καιρός μέχρι που είδαν το φως open source υλοποιήσεις. Το 2006 η CNET Networks ανέπτυξε το Solr Project, ένα εγχείρημα που αρχικά είχε αναπτυχθεί για ενδοεταιρική χρήση το 2004 στο Lucene project, ένα από τα πιο διαδεδομένα open-source libraries για αναζήτηση. Παρόλο που το Solr προσέφερε πάρα πολλές βελτιώσεις στο Lucene αλλά εννοείται η μεγαλύτερη από αυτές είναι το faceted search. Άλλο ένα open-source project, το οποίο χρήζει αναφοράς και το οποίο χρησιμοποιήθηκε στο project που συνοδεύει την πτυχιακή και θα αναλυθεί στη συνέχεια, είναι ένα component του Drupal (πασίγνωστο CMS).

Η εικόνα παρακάτω μας δείχνει το Solr και το faceted search που χρησιμοποιεί. Όπως μπορούμε να διακρίνουμε οι εφαρμογές open source μοιάζουν πάρα πολύ με τις εμπορικές υλοποιήσεις.

Εικόνα 28 - Open source project πάνω στο Faceted Search

3.10. Faceted search και κατασκευή ιστοσελίδων

Σε αυτό το σημείο της πτυχιακής θα μιλήσουμε για την προσέγγιση που ακλουθούμε, όταν πρόκειται να κατασκευάσουμε μια ιστοσελίδα που υλοποιεί faceted search τεχνικές. Θα αναφερθούμε επίσης στα προβλήματα που προκύπτουν. Θα διαχωρίσουμε τα κομμάτια της κατασκευής και των προβλημάτων που έχουμε να αντιμετωπίσουμε σε δύο μέρη: Front End και Back End. Παρόλο το διαχωρισμό, πολλές φορές θα κλιθούμε να αντιμετωπίσουμε προβλήματα που διασταυρώνουν αυτά τα δύο μέρη. Το Back End θα μας απασχολήσει ως προς το τι δεδομένα θα παρουσιάσουμε και στο Front End θα ασχοληθούμε με το πώς θα τα παρουσιάσουμε.

Χαρακτηρίσαμε το faceted search ως κάτι που μαθαίνεται εύκολα ως προς τη βάση του, αλλά παίρνει χρόνια για να το κατανοήσουμε πλήρως. Ευτυχώς εδώ δε χρειάζεται να είμαστε ειδήμονες για να καταφέρουμε αξιοπρεπείς κατασκευές.

3.10.1. Back End

Θα ξεκινήσουμε με το Back End. Θα ασχοληθούμε με μερικά από τα πιο σημαντικά θέματα που θα έρθουμε αντιμέτωποι στην προσπάθεια μας να υλοποιήσουμε faceted search σε μια ιστοσελίδα. Προβλήματα όπως το πλήθος των εγγράφων, η αποδοτικότητα του συστήματος, γλωσσικά προβλήματα κ.α. είναι μερικά μόνο από αυτά που θα αντιμετωπίσουμε.

3.10.2. Scale

Η αξία μιας συλλογής εγγράφων, συνήθως, συνδέεται με το μέγεθος της. Όταν μιλάμε για πληροφορία τότε σίγουρα όσο περισσότερη υπάρχει τόσο το καλύτερο. Πρακτικά το faceted search προσφέρει καλύτερη διαχείριση πληροφορίας από την απλή αναζήτηση, ειδικά όταν έχουμε να κάνουμε με μεγαλύτερο όγκο πληροφορίας. Από την άλλη όμως, μεγάλο μέγεθος πληροφορίας σημαίνει μεγαλύτερο κόστος σε συντήρηση, hardware και επιπλέον ανεβαίνει η πολυπλοκότητα του συστήματός μας.

Όταν μιλάμε για scale στο faceted search αναφερόμαστε στους ακόλουθους παράγοντες:

- Αριθμός εγγράφων
- Αριθμός facets ανά έγγραφο
- Κείμενο προς αναζήτηση ανά έγγραφο

Οι ακριβείς απαιτήσεις σε αποθηκευτική ικανότητα εξαρτώνται κατά πολύ από τη δομή των δεδομένων που χρησιμοποιούνται για την υλοποίηση του faceted search. Γενικότερα ορίζουμε δυο πίνακες. Ο πρώτος ο οποίος αντιστοιχεί facets και keywords με τα έγγραφα και ο δεύτερος που περιέχει την αντίθετη αντιστοιχία, δηλαδή έγγραφα με τιμές facets. Το μέγεθος ενός συστήματος πρακτικά υπολογίζεται από τον αριθμό των εγγράφων και το μέσο «μήκος» τους, το οποίο ουσιαστικά αντιπροσωπεύει το πλήθος των facets και των τιμών τους καθώς και το προς αναζήτηση κείμενο. Για μικρές συλλογές εγγράφων, όλη αυτή η πληροφορία αποθηκεύεται στην κύρια μνήμη. Αυτή η προσέγγιση βέβαια έχει επιπτώσεις στην απόδοση. Για μεγαλύτερες συλλογές η λύση της κύριας μνήμης απλά ξεφεύγει από τα όρια του εφικτού. Μην ξεχνάμε ότι η μνήμη που έχουν αυτή τη στιγμή οι περισσότεροι serves μετρούνται σε GB, ενώ η χωρητικότητα των σκληρών δίσκων σε TB και φυσικά το κόστος της κύριας μνήμης θα γίνει απαγορευτικό αν αποφασίσουμε να χρησιμοποιήσουμε μόνο RAM.

Για διάφορους λόγους μπορεί η πληροφορία να είναι αποθηκευμένη στον ίδιο server ή σε κάποιον άλλον. Εδώ έχουμε επιπλέον να κάνουμε με την υπολογιστική ισχύ του άλλου server καθώς και την απόδοση του δικτύου. Ειδικά το τελευταίο αποδεικνύεται το σημαντικότερο πρόβλημα σε αυτές τις περιπτώσεις.

3.10.3. Efficiency

Σε σύγκριση με τις υπόλοιπες τεχνικές information retrieval, το faceted search έχει ανάγκη από αρκετή υπολογιστική ισχύ. Όταν γίνεται επεξεργασία ενός ερωτήματος, το πρώτο μέλημα του είναι να υπολογίσει τον αριθμό των εγγράφων που ικανοποιούν το ερώτημα. Το δεύτερο μέλημα είναι να υπολογίσει και να μας δείξει τις τιμές των facets που χαρακτηρίζουν τα έγγραφα του αποτελέσματος. Η εφαρμογή συνήθως μας δείχνει και τον αριθμό που αντιστοιχούν σε κάθε τιμή facet. Ο υπολογισμών αυτών των facet είναι πολύ απαιτητικότερος από την απλή ανάκτηση των εγγράφων.

Υπάρχουν δύο τρόποι να υπολογίσουμε τα facets που αναφέραμε πριν: Top-down και bottom-up προσεγγίσεις. Ο πρώτος τρόπος ξεκινάει από τα facets που υπάρχουν στον πρώτο πίνακα (βλ. scale) και ψάχνει ποια ταιριάζουν με τα αποτελέσματα. Ο δεύτερος τρόπος είναι το ακριβώς ανάποδο. Αναλύει τα αποτελέσματα και ανακτά τα facets από αυτά. Και οι δυο τρόποι είναι εξαιρετικά απαιτητικοί στους υπολογισμούς.

Το κόστος των τρόπων αυτών εξαρτάται από τη δομή των δεδομένων, καθώς και από το ποσοστό αποθηκευμένης πληροφορίας στη RAM έναντι του δίσκου. Η bottom-up προσέγγιση είναι ασύμφορη στην περίπτωση που τα έγγραφα είναι αποθηκευμένα σε εξωτερική μονάδα, επειδή μπορεί να υπάρχει ανομοιομορφία στη δομή των δεδομένων και αυτό δυσχεραίνει τη συλλογή τους.

Ο Yonik Seely, κατασκευαστής του Solr, περιγράφει μια προσέγγιση που συνδυάζει και τους δύο τρόπους, ώστε να πετύχει μεγαλύτερη απόδοση από ότι με ένα τρόπο μόνο του. Δυστυχώς οι εμπορικοί προμηθευτές τέτοιων συστημάτων λίγη σημασία δίνουν στο συγκεκριμένο κομμάτι της υλοποίησης.

Επίσης είναι σημαντικό να πάρουμε απόφαση να μετρήσουμε την αποδοτικότητα του συστήματος μας, ειδικά στην περίπτωση που εξυπηρετούμε πολλούς χρήστες ταυτόχρονα. Αν η κύρια έννοια μας είναι η παραγωγικότητα, δηλαδή ο αριθμός των ερωτημάτων που θα επεξεργαστεί το σύστημα ανά δευτερόλεπτο, τότε μπορούμε να προσθέσουμε επιπρόσθετους server ώστε να εξυπηρετούμε μεγαλύτερο αριθμό χρηστών ταυτόχρονα. Εδώ βέβαια μας ενδιαφέρει και το latency του δικτύου μας, το οποίο όμως λίγα πράγματα μπορούμε να κάνουμε άμεσα για να το μειώσουμε.

Όπως σημειώθηκε, είναι δυνατόν να αποθηκεύσουμε την πληροφορία μας σε πολλαπλούς server. Αυτό μπορεί να μειώσει το latency σε παράλληλες αιτήσεις χρηστών. Μπορεί όμως να δημιουργήσει ένα πολύπλοκο σύστημα hardware το οποίο ανεβάζει το κόστος, ειδικά αυτό της συντήρησης. Στον αντίποδα, αν έχουμε ένα μόνο server να επεξεργάζεται τα ερωτήματα μας, είμαστε σε καλύτερη μοίρα λειτουργικά, αλλά αυτό έχει και τον δικό του βαθμό πολυπλοκότητας. Πολλαπλοί επεξεργαστές και πολύ-νηματικός προγραμματισμός μπορεί να μας δίνουν υπολογιστική ισχύ, αλλά μπορεί να υπάρχει έντονος ανταγωνισμός για τους πόρους του συστήματος που ίσως να οδηγήσει σε δυσλειτουργία ή και προσωρινή αναστολή λειτουργίας. Συν τοις άλλοις, όσο δελεαστική και να είναι η ύπαρξη πολλαπλών επεξεργαστών, τα σημερινά συστήματα δεν είναι σε θέση να εκμεταλλευτούν στο 100% την υπολογιστική τους ισχύ.

Ένας άλλος σημαντικός παράγοντας που επηρεάζει την αποδοτικότητα είναι η συχνότητα, με την οποία ανανεώνονται τα δεδομένα μας. Πολλά faceted search συστήματα προτιμούν την επιτόπου επεξεργασία δεδομένων παρά την προ-επεξεργασία. Εδώ έχουμε να αντιμετωπίσουμε για ακόμα μια φορά το latency. Από τη μεριά του χρήστη, αυτό μεταφράζεται στο χρόνο που μεσολαβεί από το κλικ μέχρι ο χρήστης να δει τα ανανεωμένα αποτελέσματα.

Να αναφέρουμε σε αυτό το σημείο ότι η αποδοτικότητα στο faceted search δεν είναι κάτι που δεν μπορούμε να εξαντλήσουμε σε αυτήν την πτυχιακή. Η παρούσα

πτυχιακή επιχειρεί να καταστήσει κατανοητό και σαφές ότι το faceted search είναι ιδιαίτερα απαιτητικό όσον αφορά την υπολογιστική ισχύ σε σχέση με τις απλές μηχανές αναζήτησης. Οι προγραμματιστές πρέπει να το αντιμετωπίζουν με τη δέουσα προσοχή καθώς επίσης να φροντίζουν για συχνές δοκιμές απόδοσης.

3.10.4. Information overload

Μπορεί τα προβλήματα απόδοσης και υπολογιστικής ισχύς να είναι ιδιαίτερα σημαντικά, αλλά τίποτα δεν είναι πιο σημαντικό από την εμπειρία του χρήστη. Μπορεί ένα σύστημα με πλούσιο περιεχόμενο να τραβάει την προσοχή του χρήστη, αλλά δημιουργείται και η ανάγκη να καταμεριστεί αυτή η προσοχή αποδοτικά. Ο όγκος της πληροφορίας μπορεί να λειτουργήσει και αρνητικά στην εντύπωση του χρήστη για αυτό είναι απαραίτητο να θέσουμε προτεραιότητες ως προς ποια και πόση πληροφορία θα παρουσιάζουμε. Αυτό βέβαια έχει να κάνει με το interface design και εμπίπτει σε επόμενο κεφάλαιο.

Υπάρχουν δυο παράγοντες που οδηγούν σε information overload σε ένα faceted search σύστημα:

- Ο αριθμός των facets
- Ο αριθμός των τιμών ανά facet

Δεδομένων των περιορισμών δεν μπορούμε να δείξουμε στο χρήστη όλα τα facets και τις τιμές τους. Υπάρχουν φυσικά τρόποι να τα συμμαζέψουμε και να είμαστε αποδοτικοί στην απεικόνιση. Ο αριθμός των facets αντανακλά τους τρόπους που μπορεί να ταξινομηθεί ένα κείμενο. Θεωρητικά δεν υπάρχουν περιορισμοί στον αριθμό των facets, ούτε στους διάφορους συνδυασμούς. Στη πράξη φυσικά είναι πεπερασμένος αλλά μπορεί κάλλιστα να είναι μεγάλος. Εδώ παρουσιάζεται και το πρόβλημα της σχετικότητας μεταξύ των facets. Όσο ανεξάρτητα και να φαίνονται τα facets μεταξύ τους, δύσκολα είναι έτσι στην πραγματικότητα. Το δίλλημα που προκύπτει εδώ είναι πόση προσπάθεια πρέπει να καταβάλουμε στην ανεξαρτητοποίηση των facets χωρίς όμως να χαθεί το νόημα από τον χρήστη;; Όπως και να ‘χει δεν μπορούμε να απαιτήσουμε τα facets μας να είναι ανεξάρτητα. Αντί αυτού επιχειρούμε μια προσέγγιση μεγάλου αριθμού μικρών συνόλων facets που είναι ανεξάρτητα ανάμεσα σε άλλα σύνολα. Σε καμιά περίπτωση δεν μπορούμε να τα παρουσιάσουμε όλα στο χρήστη, όχι τόσο λόγω υπολογιστικού κόστους αλλά λόγω του ότι θα «πνίξουμε» το χρήστη στην πληροφορία.

Ακολουθούν μερικές συμβουλές ως προς αυτή την κατεύθυνση:

- Προτιμούμε τα facets που έχουν μεγάλη κάλυψη ανάμεσα στα έγγραφα. Συγκριμένα, προτιμούμε αυτά που έχουν τιμές σε όλα τα έγγραφα.
- Προτιμούμε facets τα οποία έχουν στα αποτελέσματα το μεγαλύτερο μέρος των τιμών τους.
- Ενοποιούμε τα facets που περιέχουν συνηθισμένες τιμές.

Μια ακόμη πρόκληση που προκύπτει από την υπερβολική ποσότητα πληροφορίας είναι τα facets που έχουν πολλές τιμές. Όπως ακριβώς υπάρχει και περιορισμός

στον αριθμό των facets που παρουσιάζουμε στο χρήστη, υπάρχει και αντίστοιχος περιορισμός στις τιμές τους. Βέβαια το μεγάλο πλήθος τιμών μπορούμε πάντα να το παρουσιάζουμε προοδευτικά ανάλογα το «δέντρο» των facets, αλλά ένα υπερβολικά μεγάλο «δέντρο» θα μειώσει την χρηστικότητα του. Πρέπει να έχουμε πάντα στο νου μας ότι τα facets πρέπει να βοηθάνε το χρήστη, όχι να του αποσπούν την προσοχή με το πλήθος τους. Επιπλέον, δεν είναι πάντα εύκολο να ιεραρχήσουμε το «δέντρο» των facets. Πολλές φορές μάλιστα είναι δυσκολότερο να ορίσουμε την ιεραρχία παρά τις τιμές αυτές καθ' αυτές. Συν τοις άλλοις, facets όπως ονόματα δεν είναι ιδιαίτερα βολικά, ίσως και καθόλου, στο να ιεραρχηθούν. Εντέλει, καταλήγουμε να θεωρούμε ότι καλύτερα να μην έχουμε ιεραρχική ταξινόμηση καθόλου, παρά να έχουμε κάποια προβληματική. Και πάντα πρέπει να είμαστε προσεκτικοί, ώστε να μην καταφύγουμε σε αυστηρή ταξινόμηση, γιατί έτσι ακυρώνουμε τα οφέλη του faceted search. Πρακτικά θα πρέπει να δεχθούμε το γεγονός ότι ορισμένα facets θα έχουν πολλές τιμές και δεν μπορούν να ταξινομηθούν ιεραρχικά με κάποια αποδοτική μέθοδο.

Εδώ θα αναφέρουμε ορισμένες τεχνικές ώστε να απορροφήσουμε τη συμφόρηση της πληροφορίας:

- Παρουσίαση μόνο των τιμών με τη μεγαλύτερη συχνότητα στα αποτελέσματα
- Παρουσίαση μόνο των τιμών που εμφανίζονται πιο συχνά στα αποτελέσματα παρά στο σύνολο
- Δημιουργία ιεραρχίας που θα βασίζεται σε υποσύνολα τιμών
- Σύμπλεξη των facets βάση τις μεταξύ τους ομοιότητες

Μεγάλοι αριθμοί facets και μεγάλο πλήθος τιμών λειτουργούν καταλυτικά στην όξυνση του προβλήματος της υπερβολικής πληροφορίας. Η καλύτερη λύση είναι, όπου είναι δυνατό, να περιορίσουμε τους αριθμούς. Όπου αυτό δεν είναι δυνατόν, οι τεχνικές που αναφέραμε παραπάνω θα φανούν ιδιαίτερα χρήσιμες στην άμβλυνση του προβλήματος.

3.10.5. Διαθεσιμότητα μεταδεδομένων

Το faceted search προϋποθέτει ότι η συλλογή των εγγράφων μας είναι οργανωμένη, με βάση faceted classification τεχνικές. Δυστυχώς συχνά υπάρχουν έγγραφα και δεδομένα που ή δεν είναι ταξινομημένα ή είναι μη δομημένα. Μπορεί το faceted search να βοηθήσει εδώ; Η γρήγορη απάντηση είναι όχι. Το faceted search χρειάζεται έγγραφα με faceted μεταδεδομένα. Διαφορετικά έχουμε απλό κείμενο. Η αναλυτική απάντηση είναι ότι υπάρχουν διάφορες τεχνικές, ώστε να εμπλουτίσουμε το μη δομημένο κείμενο ώστε να περιλαμβάνει μεταδεδομένα.

Ας αναφέρουμε μερικές στρατηγικές:

- Εντοπίζουμε και ορίζουμε μεταδεδομένα όπως πηγή, τύπος και μήκος εγγράφου. Αυτά τα δεδομένα είναι συνήθως άμεσα διαθέσιμα
- Ορίζουμε κανόνες, βάση των οποίων θα κάνουμε μια πρώτη ταξινόμηση των εγγράφων
- Με τη χρήση της τεχνικής terminology extraction ανακτούμε όρους και δημιουργούμε ένα «λεξικό», ώστε να αντιστοιχίσουμε τα έγγραφα

Εν συντομία, το faceted search προϋποθέτει τα facets να είναι χρηστικά. Η ανάκτηση facets από μη δομημένα έγγραφα δυσκολεύει τους αυτοματισμούς και απαιτεί επιπλέον εργασία όσον αφορά την οργάνωση τους. Μια προσέγγιση που δε συζητήσαμε είναι το crowd-sourcing όπου διάφοροι χρήστες/προγραμματιστές κλπ. είναι που παρέχουν τα μεταδεδομένα. Κάτι τέτοιο συνήθως επιτυγχάνεται αναλύοντας τις λέξεις που χρησιμοποιούν στις αναζητήσεις τους.

3.10.6. Το πρόβλημα του λεξιλογίου

Το 1987, δημοσιεύθηκε ένα σύγγραμμα με τίτλο “The Vocabulary Problem in Human-System Communication”, στο οποίο παρουσιάζεται το πώς οι λέξεις κλειδιά, που χρησιμοποιούνται στις βάσεις δεδομένων, λίγες φορές συμπίπτουν με αυτές που αναζητούν οι χρήστες. Η λύση που προτείνεται είναι μια προσαρμοστική δεικτοδότηση των βάσεων δεδομένων. Οι λέξεις που θα εισάγουν οι χρήστες θα συλλέγονται και θα προσαρμόζονται σε αυτές τις βάσεις. Το πρόβλημα αυτό του «λεξιλογίου», είναι πρόκληση και κίνητρο ταυτόχρονα για την βελτίωση του faceted search. Τα εργαλεία που χρησιμοποιεί για το φιλτράρισμα των αποτελεσμάτων προσφέρουν μια μορφή οδηγού για τον χρήστη, αλλά παράλληλα γίνεται και μια προσπάθεια να παρουσιάζονται αυτά που τον αφορούν.

Το πρώτο κομμάτι αυτού του προβλήματος αναφέρεται στην έλλειψη ακρίβειας, που αφορά τις εισαγωγές των χρηστών, και έγκειται περισσότερο στην κατηγορία information retrieval. Το δεύτερο κομμάτι αφορά περισσότερο το faceted search. Τα facets, ο τρόπος με τον οποίο παρουσιάζονται και γενικότερα οι δυνατότητες για φιλτράρισμα πληροφορίας, πρέπει να καθοδηγούν τον χρήστη στο που θα βρει αυτό που ψάχνει, τι «κόστος» θα έχει η εύρεση του καθώς και το ίδιο το περιεχόμενο αυτό καθαυτό. Είναι πολύ σημαντικό ο χρήστης όταν θα μελετά τις επιλογές του να βλέπει τουλάχιστον μια επιλογή προς την κατεύθυνση που θέλει να αναζητήσει δεδομένα. Ιδανικά δε θα έπρεπε να δει δύο ή περισσότερες επιλογές γιατί αυτό μπορεί να προκαλέσει σύγχυση.

Τα faceted search συστήματα μπορούν να βελτιστοποιήσουν τις μεθόδους καθοδήγησης, παρουσιάζοντας previews του περιεχομένου που σχετίζεται με την προς αναζήτηση πληροφορία και τα επιλεγμένα facets. Στην ιδανική περίπτωση τα previews πρέπει να είναι συνοπτικά αλλά να δείχνουν καθαρά τις αλλαγές που προκύπτουν από τις διάφορες επιλογές των facets.

3.10.7. Multiple entity types

Το τελευταίο θέμα που θα αναλύσουμε στο κεφάλαιο αυτό είναι οι συλλογές εγγράφων που περιέχουν πολλαπλούς τύπους οντοτήτων και πολλαπλές σχέσεις μεταξύ τους. Για όσους δεν έχουν υπόψη τους για τι ακριβώς μιλάμε θα παρουσιάσουμε ένα παράδειγμα: Μια ψηφιακή βιβλιοθήκη με άρθρα όπου κάθε άρθρο έχει έναν ή περισσότερους συγγραφείς. Υπάρχουν facets που περιγράφουν τα άρθρα, όπως θέμα, έτος έκδοσης, γλώσσα κ.α. Ας θεωρήσουμε τώρα ότι οι συγγραφείς από μόνοι τους είναι ένα facet που περιγράφει τα άρθρα. Εκεί που αρχίζουν τα προβλήματα είναι όταν θέλουμε οι συγγραφείς να κατηγοριοποιούνται σε facets όπως π.χ. εθνικότητα. Αναρωτηθείτε τι μπορεί να συμβεί όταν ένας χρήστης αναζητήσει ένα άρθρο με facets που χαρακτηρίζουν το άρθρο, αλλά ταυτόχρονα με facets που χαρακτηρίζουν και τους συγγραφείς. Και τι θα συμβεί αν ένα άρθρο έχει πολλούς συγγραφείς;;

Μια παραδοσιακή faceted search προσέγγιση προϋποθέτει ένα πιο χαλαρό μοντέλο, το οποίο στα βιβλία πληροφορικής το ονομάζουν un-normalized. Στο σύστημα μας υπάρχουν 2 ειδών αντικείμενα, τα facets και τα έγγραφα. Αν ο συγγραφέας είναι facet δεν μπορεί να είναι έγγραφο ταυτόχρονα και ως εκ τούτου δεν μπορεί να έχει δικά του facets. Το un-normalized μοντέλο έχει ορισμένους άτυπους κανόνες στα ερωτήματα που μπορούν να εισάγουν οι χρήστες. Για παράδειγμα οι χρήστες δεν μπορούν να αναζητήσουν άρθρα από συγγραφείς που είναι από τις Ηνωμένες Πολιτείες Αμερικής παρά μόνο αν αυτό το facet χαρακτηρίζει και τα άρθρα αντί μόνο τους συγγραφείς. Δυστυχώς με αυτόν τον τρόπο τα αποτελέσματα μας είναι λειψά. Για παράδειγμα θα έχουμε διαφορετικά αποτελέσματα αν αναζητήσουμε σετ από έγγραφα από αμερικάνο συγγραφέα από το Mellon University σε σχέση με το άθροισμα των σετ «αναζητούμε έγγραφα από αμερικάνο συγγραφέα» και «αναζητούμε έγγραφα συγγραφέων από το Mellon University». Αυτό συμβαίνει γιατί το άρθρο μπορεί να έχει δύο συγγραφείς που ο καθένας ικανοποιεί διαφορετικό κριτήριο. Για να υλοποιηθεί αυτός ο διαχωρισμός αποδοτικά θα πρέπει να ορίσουμε και τους συγγραφείς και τα άρθρα σαν έγγραφα ώστε το καθένα να χαρακτηρίζεται από τα δικά του facets. Εδώ να πούμε ότι αυτή η τεχνική είναι μια semantic web τεχνική σαν αυτές που χρησιμοποιεί η Endeca. Η συγκεκριμένη τακτική δυστυχώς είναι περίπλοκη και έχει μεγάλο υπολογιστικό κόστος που δυσκολεύει την σχεδίαση του interface. Αυτό έχει ως αποτέλεσμα να δυσκολευτεί ο χρήστης.

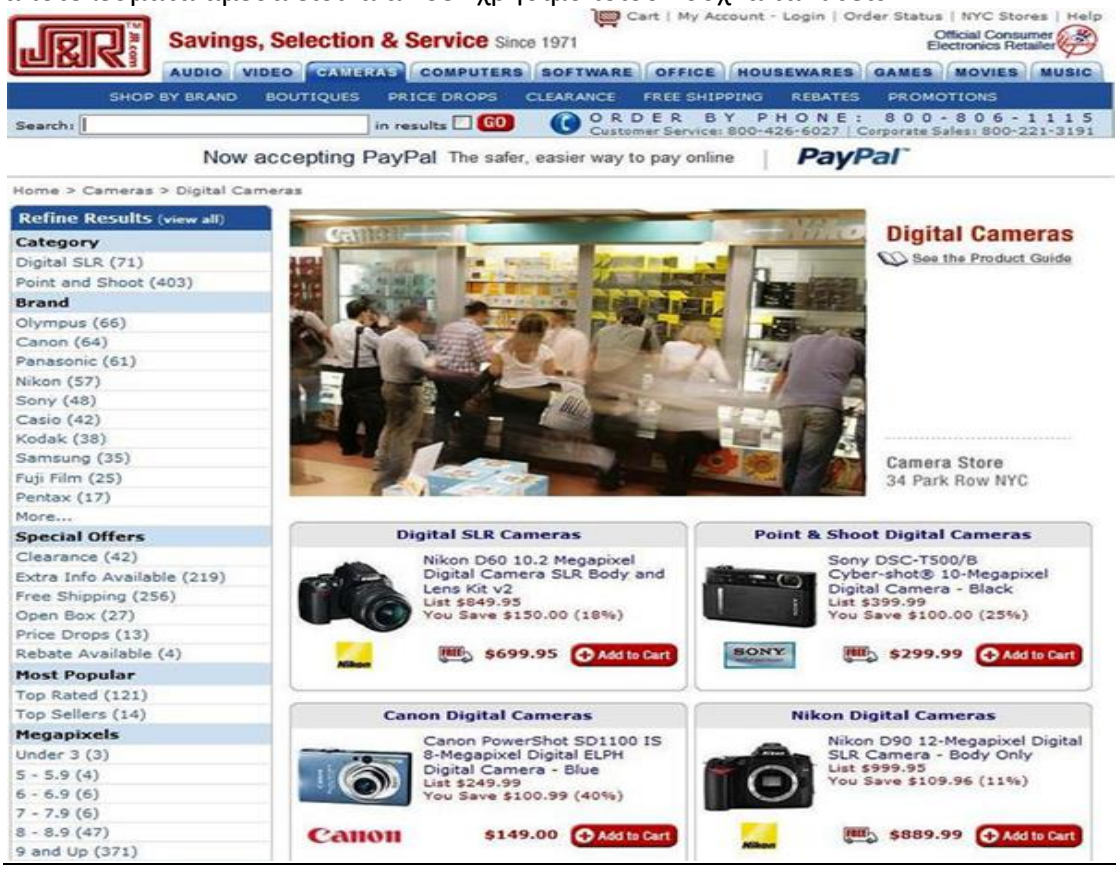
3.10.8. Front End

Στο προηγούμενο κεφάλαιο δώσαμε έμφαση στις προκλήσεις που παρουσιάζονται όταν θέλουμε να σχεδιάσουμε το back-end ενός faceted search συστήματος. Προκλήσεις όπως το τι πληροφορία να παρουσιάσουμε στον χρήστη όπως και το κόστος υπολογισμού της. Σε αυτό το κεφάλαιο θα μελετήσουμε τις αντίστοιχες προκλήσεις και τα ζητήματα που αντιμετωπίζουμε στο front end του συστήματος μας. Το front end είναι ένα εξίσου σημαντικό κομμάτι γιατί πρόκειται για αυτό που βλέπει και χρησιμοποιεί ο χρήστης. Σαν τελικός αποδέκτης που είναι, έχει τον τελευταίο λόγο για το αν το σύστημα μας θα είναι επιτυχημένο ή όχι.

3.10.9. Που και πότε παρουσιάζουμε τα facets

Όπως έχουμε ήδη πει, ένα faceted search σύστημα «απαντά» σε ένα ερώτημα παρουσιάζοντας συλλογές εγγράφων που ταιριάζουν τα κριτήρια αναζήτησης, καθώς και ένα σετ από facets τα οποία καθοδηγούν τους χρήστες στο φιλτράρισμα των αποτελεσμάτων. Κανείς δεν μας λέει όμως πώς θα τα παρουσιάσουμε αυτά στον χρήστη ώστε να είναι χρηστικά και να μην προκαλέσουν σύγχυση. Οι προεπιλογές μιας εφαρμογής μπορεί ή να παρουσιάζουν όλα τα facets και όλα τα αποτελέσματα ή μόνο ένα από τα δύο. Υπάρχει τρόπος να βρούμε μια μέση λύση; Η σύντομη απάντηση είναι όχι. Οι διάφορες εφαρμογές και οι απαιτήσεις του χρήστη απαιτούν διαφορετικά design. Παρόλα αυτά εκείνο που μπορούμε να κάνουμε είναι να απαριθμήσουμε μερικές επιλογές, να ελέγξουμε την σχέση μεταξύ τους και έπειτα να παρουσιάσουμε τα πιο σχετικά αποτελέσματα.

Ας θεωρήσουμε τώρα interfaces τα οποία παρουσιάζουν και τα facets και τα κείμενα μαζί. Υπάρχουν δυο συμβατικά layout: Τοποθετούμε τα facets σε μια στήλη αριστερά και τα αντίστοιχα αποτελέσματα δεξιά ή πάνω τα facets και κάτω τα αποτελέσματα. Μια εναλλακτική είναι να τοποθετήσουμε τα facets κάτω, αλλά σε μικρές αναλύσεις θα χρειαστεί να χρησιμοποιήσει scrolls ο χρήστης και μπορεί να μην προσέξει τα facets εξ αρχής. Από την άλλη αν τοποθετήσουμε τα facets αριστερά και στο κέντρο τα αποτελέσματα, ο χρήστης θα δώσει και πάλι έμφαση στα αποτελέσματα. Οι χρήστες προτιμούν μια οπτική η οποία του δείχνει τα αποτελέσματα άμεσα ειδικά αν δεν χρησιμοποιούν συχνά τα facets.



Εικόνα 29 - Προβολή facets αριστερά από τα αποτελέσματα

Τοποθετώντας τα facets πάνω από τα αποτελέσματα, όπως στην πιο κάτω εικόνα, τα καθιστά εύκολα ορατά από τον χρήστη. Από την άλλη όμως τα αποτελέσματα δεν παρουσιάζονται πλήρως και θα χρειαστεί μια επιπλέον προσπάθεια από τον χρήστη. Αυτό βέβαια όμως μπορεί να καταλήξει σε μια χρήσιμη επίπτωση. Ο χρήστης ίσως αναγκαστεί κατά κάποιο τρόπο να φιλτράρει τα αποτελέσματα αντί να τα κοιτάξει ευθύς αμέσως.

Εικόνα 30 - Προβολή facets πάνω από τα αποτελέσματα

Μία άλλη, λιγότερο συνηθισμένη, λύση είναι να παρουσιάζουμε τα αποτελέσματα και τα facets το καθένα στη δική του καρτέλα. Το θέμα που προκύπτει εδώ είναι πια από τις δυο καρτέλες θα παρουσιάσουμε πρώτη. Η λύση που προτιμάται τις περισσότερες φορές είναι να αναγκάζουμε κατά κάποιο τρόπο τον χρήστη να ψάξει για τα αποτελέσματα ώστε να προωθήσουμε την χρήση των facets. Ευτυχώς σπάνια οι χρήστες δυσανασχετούνε ψάχνοντας αποτελέσματα, αποτελώντας την μειοψηφία. Πιο εύκολα θα σκεφτούν να ψάξουν για αποτελέσματα παρά για facets που μπορεί να αγνοούν και την ύπαρξη τους.

3.10.10. Οργανώνοντας τα facets και τις τιμές τους

Στο προηγούμενο κεφάλαιο αντιμετωπίσαμε το πρόβλημα του information overload από την σκοπιά του back end. Το πρόβλημα που θα αντιμετωπίσουμε στο front end είναι παρόμοιας φύσεως αλλά αυτή τη φορά έχει να κάνει και με το πώς θα παρουσιάζονται τα facets. Πως λοιπόν θα οργανώσουμε αυτήν την πληροφορία;;

Προσεγγίζουμε την πρόκληση αυτή μειώνοντας τον αριθμό των facets και οργανώνοντας τα κατάλληλα με user-friendly σκοπιά.

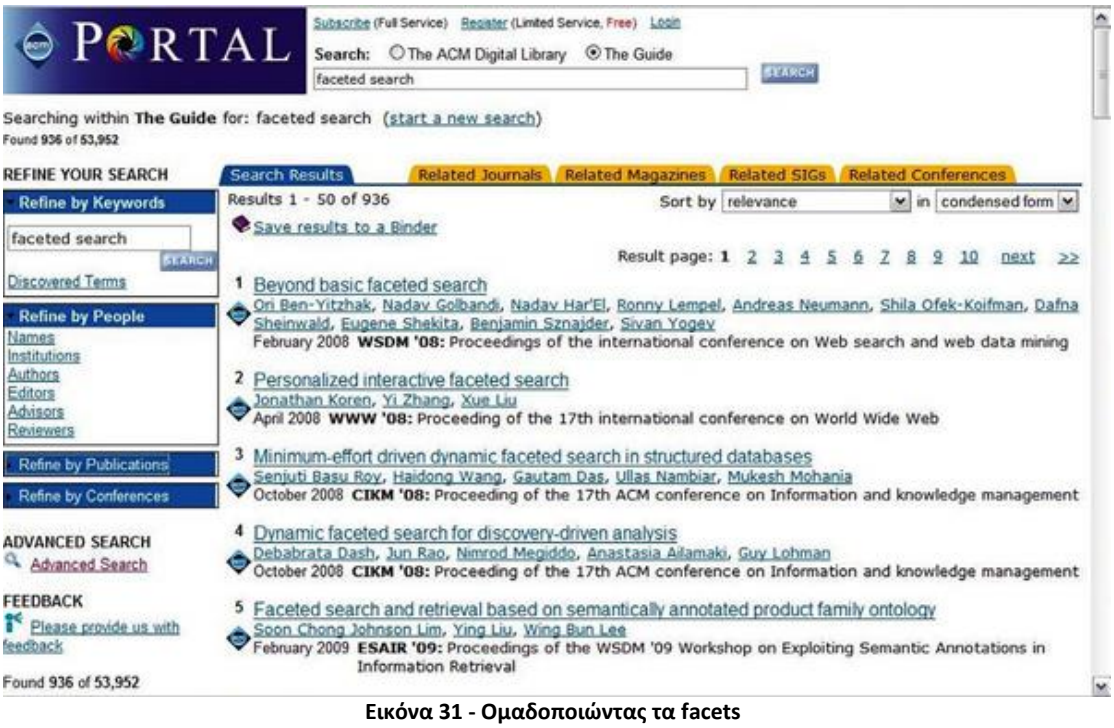
Υπάρχουν τρεις γενικές στρατηγικές οργάνωσης των facets οι οποίες μάλιστα δε διαφέρουν από αυτές που συζητήσαμε για το back-end:

- Χρήση σταθερής σειρά στα facets καθώς ο χρήστης περιηγείται
- Ταξινόμηση και εμφάνιση των facets με σειρά σχετικότητας με τα αποτελέσματα
- Οργάνωση παρόμοιων facets σε ομάδες

Η χρήση στατικών facets σε σχέση με τα ranked είναι ένα ενδιαφέρον δίλλημα. Η στατική σειρά έχει το πλεονέκτημα ότι δεν αλλάζει. Ως αποτέλεσμα δεν μπερδεύει τον χρήστη και επιπλέον διατηρεί μια ξεκάθαρη εικόνα της κατάστασης. Αυτή η τεχνική είναι πολύ χρήσιμη όταν τα facets είναι λίγα και μπορούμε να τα παρουσιάζουμε όλα μαζί.

Στον αντίποδα όταν έχουμε πλήθος facets, η στατική σειρά δεν είναι τόσο αποδοτική. Π.χ. σε μια ιστοσελίδα ηλεκτρονικού καταστήματος που πουλάει ηλεκτρονικά εξαρτήματα, ορισμένα facets σχετίζονται με πολύ λίγα από αυτά. Οπότε δεν είναι πάντα χρήσιμο να τα παρουσιάζουμε.

Ομαδοποιώντας τα σχετικά facets, όπως βλέπουμε και στην παρακάτω εικόνα, έχουμε τη δυνατότητα να παρουσιάσουμε περισσότερα facets σε λιγότερο χώρο.



Εικόνα 31 - Ομαδοποιώντας τα facets

Οι τεχνικές collapse/expand των contents, που μας προσφέρει ο προγραμματισμός ιστοσελίδων σήμερα, βοηθάει ακόμα περισσότερο αυτήν την στρατηγική. Πολλές ιστοσελίδες και διαδικτυακές εφαρμογές που υλοποιούν faceted search σχεδιάστηκαν χρησιμοποιώντας τεχνολογίες όπως AJAX και Flash δημιουργώντας ένα ιδιαίτερα αλληλεπιδραστικό περιβάλλον. Αυτό βοήθησε στο αναδειχθούν οι δυνατότης του faceted search ακόμη περισσότερο.

Οι ίδιες στρατηγικές που προτείναμε για τα facets, η ίδιες πάνω κάτω ισχύουν για την παρουσίαση των τιμών τους:

- Χρήση στατικής σειράς
- Ταξινόμηση ανάλογα με τη συχνότητα εμφάνισης
- Προοδευτική απεικόνιση των facet values ένα επίπεδο τη φορά

Η τελευταία τακτική που θα αναφέρουμε έχει να κάνει με facets που δεν είναι ιεραρχημένα. Μπορούμε να δημιουργήσουμε μια εικονική ιεραρχία με κόμβους που δεν θα έχουν περισσότερα από έναν ορισμένο αριθμό από «παιδιά». Αυτός ο αριθμός θα εξαρτηθεί από το design μας. Για παράδειγμα θεωρούμε ένα facet με πολλές τιμές, όπως «Συγγραφείς». Μπορούμε να τους ταξινομήσουμε με βάση το επώνυμο τους, χωρίζοντας τους σε υποσύνολα Α-Ε, Ζ-Κ, Λ-Ο, Π-Υ, Φ-Ω. Τα υποσύνολα αυτά μπορούν επίσης να χωριστούν στα χωριστά γράμματα. Φυσικά μπορούμε να χρησιμοποιήσουμε αυτή την τεχνική και σε αριθμητικά δεδομένα, όπου δημιουργώντας υποσύνολα έχουμε ένα πιο χρηστικό interface.

3.10.11. Το search box

Έχουμε δώσει έμφαση και έχουμε επικεντρώσει τις αναλύσεις μας στη χρήση των facets. Καλό είναι όμως να θυμηθούμε ότι το faceted search είναι search πάνω απ' όλα. Επίσης μην ξεχνάμε ότι η «είσοδος» στο faceted search από τις περισσότερες εφαρμογές είναι μέσω ενός search box. Χωρίς αυτό, διαθέτουμε μεν ένα navigation σύστημα χρήσιμο σε αρκετές εφαρμογές, αλλά περιοριστικό σε σχέση με τις δυνατότητες του faceted search.

Συνδυάζοντας search box και facets εκμεταλλευόμαστε την μέγιστη δυνατή απόδοση ενός τέτοιου συστήματος. Ο συνδυασμός επιτρέπει τους χρήστες να δημιουργούν ημι-δομημένα ερωτήματα ώστε να προσπελαύνουν δομημένο και μη δομημένο περιεχόμενο.

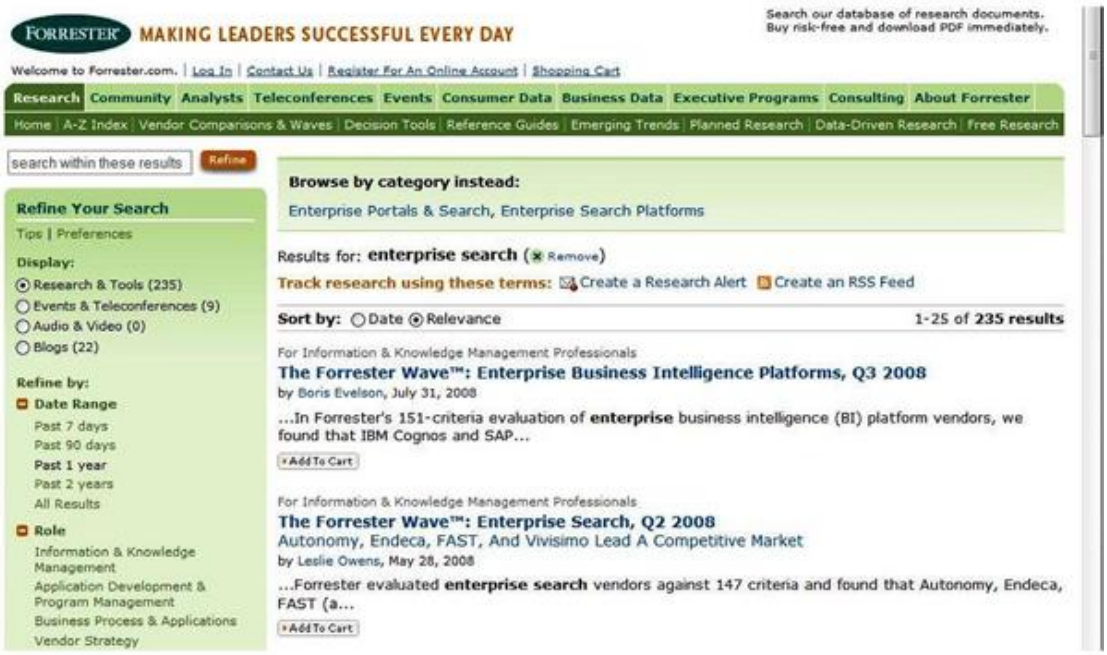
Το search box προσφέρει και αυτό δυνατές προκλήσεις στους σχεδιαστές. Ο σχεδιαστής ενός faceted search συστήματος πρέπει να πάρει αποφάσεις ως προς το πώς θα συμπεριφέρεται το search box:

- Κάθε νέο ερώτημα θα ξενικά την αναζήτηση από την αρχή ή θα ενσωματώνεται στην ήδη υπάρχουσα;
- Θα γίνεται αναζήτηση σε όλο το κείμενο ή σε συγκεκριμένα πεδία του;

- Πως θα γίνεται η επεξεργασία ερωτημάτων με πάνω από δύο λέξεις;; Θα συνδέονται με OR, AND ή θα αντιμετωπίζονται σαν φράση; Θα έχει επιλογή ο χρήστης σε αυτό;;
- Θα πρέπει τα συστήματα αυτά να παρέχουν πολλαπλά search box, ένα παραμετρικό ή κάποιο πιο προηγμένο;

Αυτά τα ερωτήματα είναι μεν ένα μέρος των προκλήσεων που έχουμε να αντιμετωπίσουμε, αλλά είναι από τα βασικότερα.

Αρχικά, θα απαντήσουμε στο αν θέλουμε κάθε νέο ερώτημα να σέβεται τα προηγούμενα. Στα περισσότερα σενάρια χρήσης, ο χρήστης εισάγει το ερώτημα στο search box και στη συνέχεια χρησιμοποιεί τα facets. Τι γίνεται όμως στην περίπτωση που αντιστρέψουμε τη διαδικασία;;
Η συμβατική και ίσως η πιο ασφαλής προσέγγιση είναι ή η χρήση του search box να ξεκινάει την αναζήτηση από την αρχή ή να κάνει αναζήτηση στα ήδη υπάρχοντα αποτελέσματα, αγνοώντας όμως τα facets.



Εικόνα 32 - Επιλογή χρήσης του search box στα αποτελέσματα

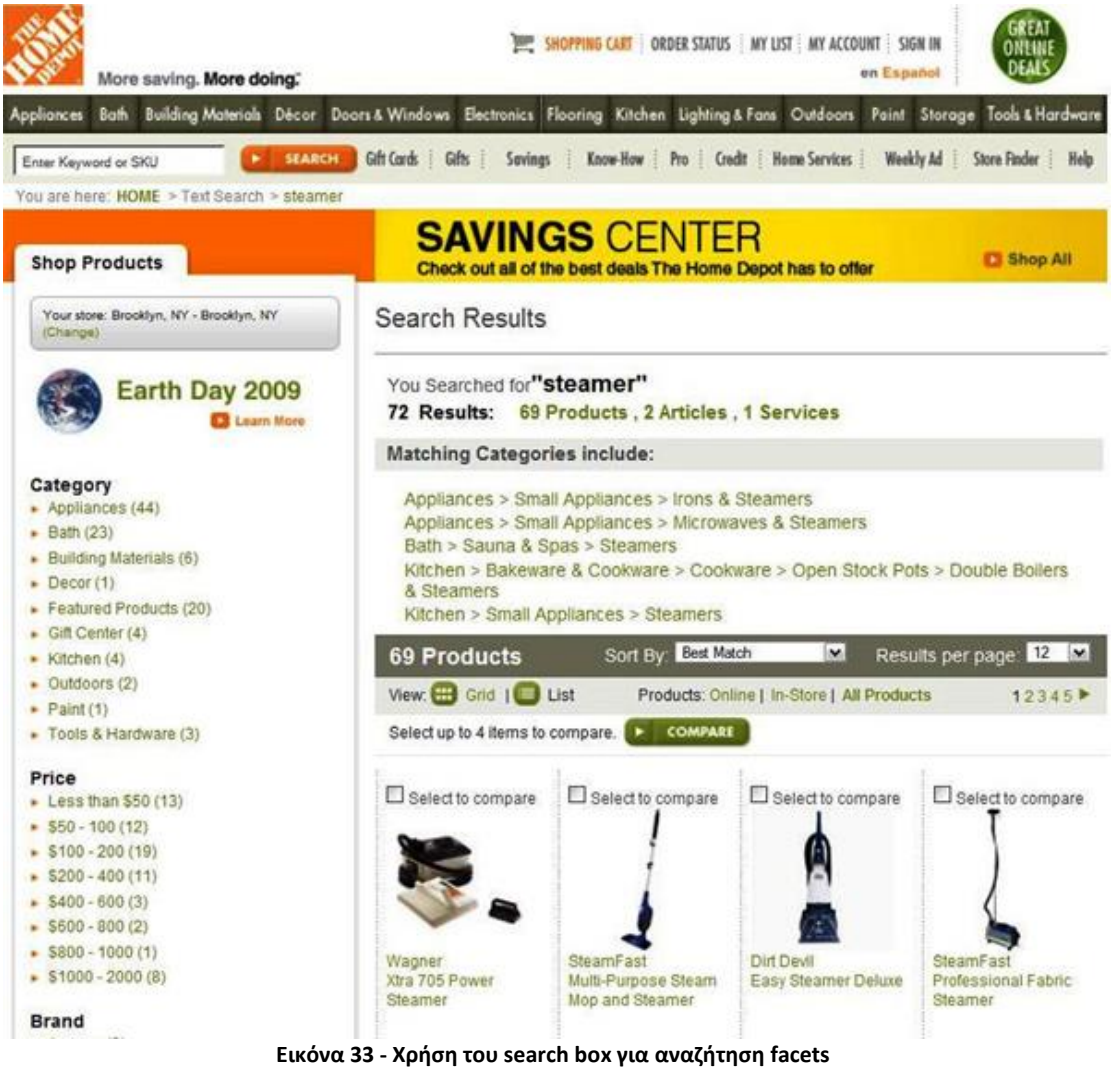
Το interface της παραπάνω εικόνας παρουσιάζει αυτή την προσέγγιση. Το search box ενημερώνει τον χρήστη για την δυνατότητα που έχει να ψάξει ανάμεσα στα αποτελέσματα.

Τώρα ας δούμε τι μπορούμε να κάνουμε με την περίπτωση που θέλουμε το ερώτημα μας να κάνει αναζήτηση σε όλο το κείμενο ή σε πεδία. Η πιο συνηθισμένη υλοποίηση είναι αυτή που ψάχνει σε ολόκληρο το κείμενο και παρουσιάζει πρώτα τα πιο σχετικά αποτελέσματα.

Η μέθοδος αυτή δουλεύει καλά σε συμβατικές μηχανές αναζήτησης αλλά υπονομεύει τις δυνατότητες του faceted search. Όπως αναλύσαμε σε προηγούμενο κεφάλαιο το faceted search υλοποιεί ένα set retrieval μοντέλο: Τα κριτήρια που ορίζουμε επιστρέφουν τις ακριβείς αντιστοιχήσεις και όχι τα πιο σχετικά αποτελέσματα. Αν τα περισσότερα αποτελέσματα δεν είναι σχετικά τότε το φιλτράρισμα που κάνουμε ίσως και να μην είναι ιδιαίτερα χρήσιμο και αποδοτικό.

Μια εναλλακτική λύση είναι να ορίσουμε ως προεπιλεγμένη συμπεριφορά την τάση προς ακρίβεια. Για παράδειγμα να γίνεται σύγκριση του ερωτήματος με τον τίτλο, εκτός και αν ο χρήστης ορίσει διαφορετικά.

Η ερώτηση κλειδί όσο αφορά το design είναι, αν το κόστος της αυξημένης ακρίβειας εκτοπίζει το κόστος της μειωμένης ανάκλησης. Φυσικά είναι εφικτό να αντισταθμίσουμε και να ισορροπήσουμε το συγκεκριμένο δίλλημα συνδυάζοντας τα ερωτήματα αρχικά, για να πετύχουμε καλή ανάκληση, και έπειτα να χρησιμοποιήσουμε τα facets για να πετύχουμε ακρίβεια. Το search box έχει επίσης τη δυνατότητα να κάνει αναζήτηση για facets και όχι μόνο για έγγραφα, όπως μας δείχνει η παρακάτω εικόνα.



Εικόνα 33 - Χρήση του search box για αναζήτηση facets

Το πλεονέκτημα αυτής της δυνατότητας είναι ότι τα αποτελέσματα θα περιλαμβάνουν έγγραφα που θα περιέχουν τις τιμές των facets που αναζητήσαμε. Ως εκ τούτου θα έχουμε μεγαλύτερη ακρίβεια στα αποτελέσματα. Ακόμη, είναι δυνατόν να αναζητήσουμε συνδυασμούς facets για ακόμα μεγαλύτερη ακρίβεια.

Τα ερωτήματα που χρησιμοποιούν πολλαπλές λέξεις, ή κάποια εργαλεία επέκτασης ερωτημάτων, μπορούν επίσης να φέρουν στην επιφάνεια το πρόβλημα της ακρίβειας και της ανάκλησης. Παρόλα αυτά λόγω οικειότητας των χρηστών με τις μηχανές αναζήτησης, έχουν συνηθίσει το γεγονός ότι οι μηχανές ερμηνεύουν τα ερωτήματα σαν να υφίσταται χρήση του AND ανάμεσα τους. Για παράδειγμα, η αναζήτηση για “faceted search” ψάχνει έγγραφα που περιέχουν και τις δυο λέξεις αλλά όχι απαραίτητα με την δοθείσα σειρά.

Τέλος, έχουμε το ερώτημα για το αν πρέπει να προσφέρουμε στους χρήστες πολλαπλά ή παραμετρικά search box ή κάποιο πιο προχωρημένο interface. Δεν υπάρχουν αυστηροί κανόνες εδώ, αλλά όπως και να έχει οι υλοποιήσεις αυτές μάλλον θα μπερδέψουν τους χρήστες περισσότερο. Αρκετοί ίσως παραμετροποιήσουν το search box, αλλά οι περισσότεροι θα αρκεστούν στις προεπιλεγμένες λειτουργίες του. Για αυτό το λόγο τις περισσότερες φορές οι προεπιλεγμένες ρυθμίσεις συνήθως είναι οι πιο αποδοτικές για το σύνολο των αναζητήσεων και οι προγραμματιστές τις περισσότερες φορές χρησιμοποιούν αυτή την τακτική. Παρομοίως, ενώ μια μειοψηφία χρηστών θα εκτιμήσει μια προχωρημένη υλοποίηση οι περισσότεροι από αυτούς ίσως και να μην καταλάβουν ότι υπάρχει.

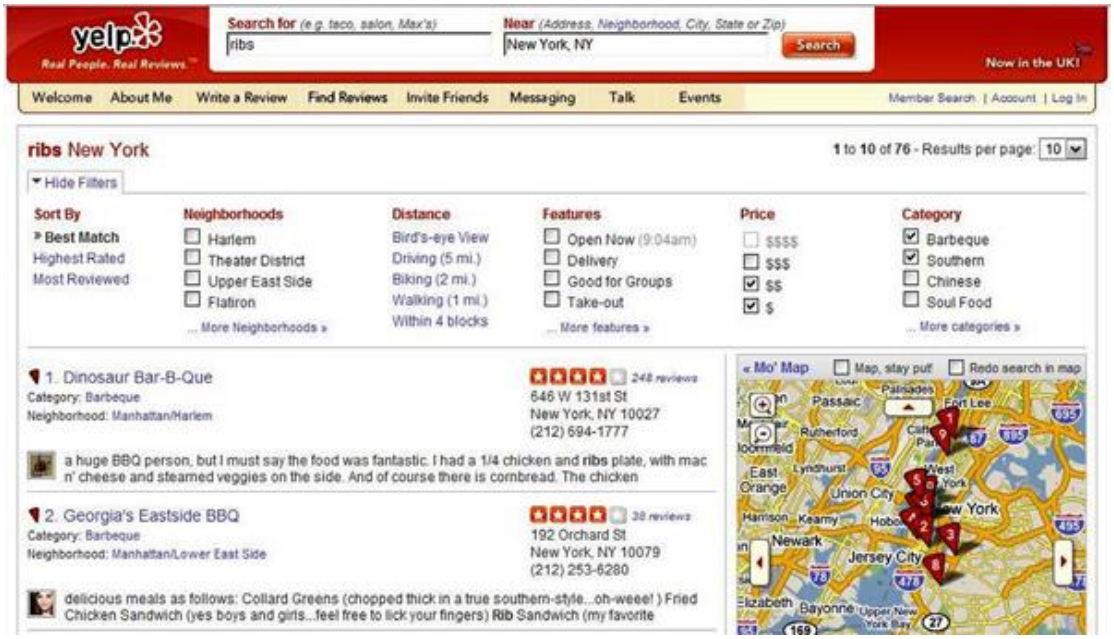
Για να αποφευχθεί η σύγχυση στους χρήστες, πολλές εφαρμογές τοποθετούν τα advanced search box τους σε ξεχωριστή σελίδα.

3.10.12. Πολλαπλές επιλογές από facets

Το πιο συνηθισμένο σενάριο χρήσης στις εφαρμογές faceted search είναι ο χρήστης να επιλέξει το πολύ μια τιμή ανά facet. Υπάρχουν όμως τουλάχιστον δύο τρόποι να επιλέξουμε πολλαπλές τιμές από το ίδιο facet:

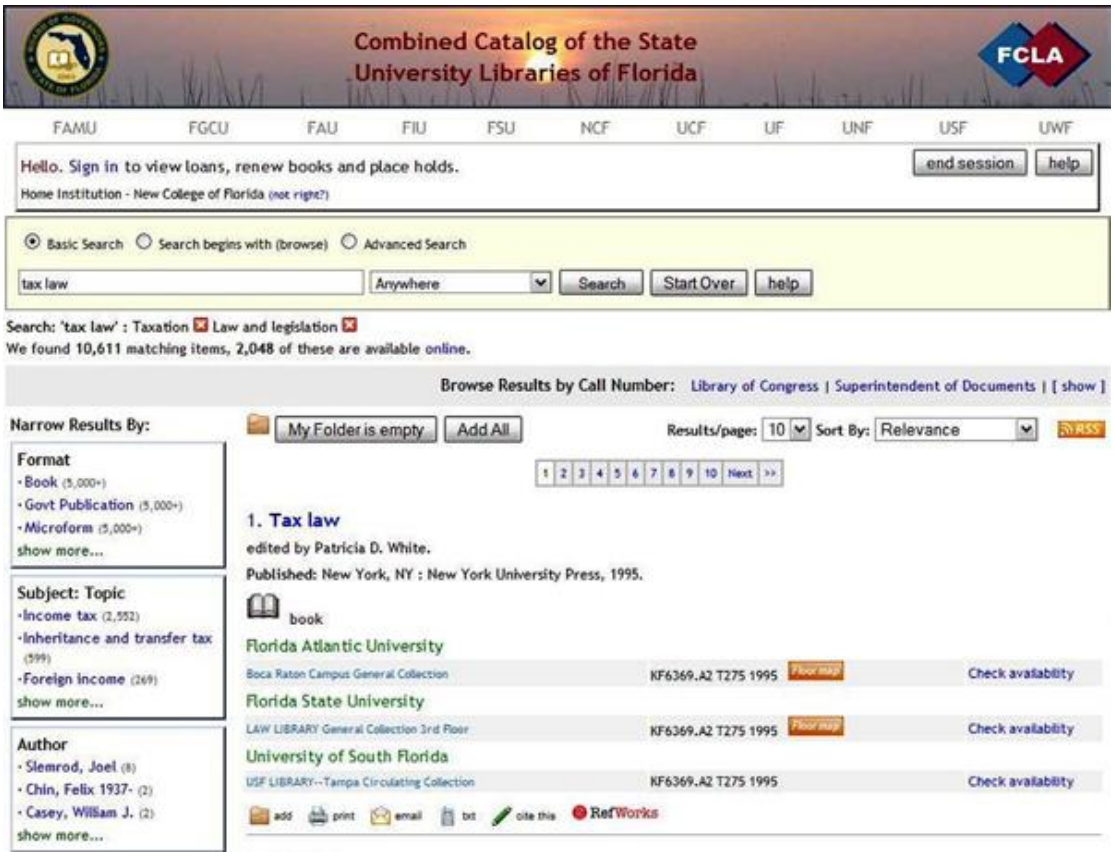
- Χρήση του τελεστή OR ή επιλογή εύρους τιμών.
- Χρήση του τελεστή AND

Η σχεδιαστική πρόκληση που αντιμετωπίζουμε εδώ είναι να γίνει σαφές στον χρήστη ποιος από τους δύο τελεστές είναι εν ενεργεία ανά πάσα στιγμή, ειδικά αν η εφαρμογή προσφέρει και τους δυο τρόπους. Οι χρήστες από τη φύση τους δεν κατέχουν καλά την Boolean λογική. Για αυτό είναι πολύ σημαντικό να σχεδιάσουμε το interface μας με τέτοιο τρόπο, ώστε να υλοποιεί και οικία στοιχεία στους χρήστες. Μην ξεχνάμε ότι αποκρινόμαστε σε χρήστες κάθε επιπέδου.



Εικόνα 34 - Συμβατική σχεδίαση faceted search layout

Στην παραπάνω εικόνα για παράδειγμα, τα check boxes ωθούν προς την συμβατική κατεύθυνση. Υπάρχουν λίγα interface που επιτρέπουν συνδετικούς αλγορίθμους αλλά για όσα το κάνουν, χρησιμοποιούν συμβατικά design με links ώστε να είναι πιο οικείο το περιβάλλον όπως στην επόμενη εικόνα.



Εικόνα 35 - Συμβατική σχεδίαση faceted search layout II

3.10.13. Design Patters

Στις προηγούμενες παραγράφους δώσαμε έμφαση σε συγκεκριμένα κομμάτια του design ενός faceted search interface, όπου συνήθως προκύπτουν θέματα χρηστικότητας. Μα πιο σφαιρική προσέγγιση σε αυτά τα θέματα είναι η χρήση design patterns. Τα design patterns πρωτοεμφανίστηκαν στην τεχνολογία λογισμικού και πρόκειται ουσιαστικά για μία λύση που μπορούμε να χρησιμοποιήσουμε σαν θεμέλιο στο σχεδιασμό μας. Παρόλο που η τεχνολογία του faceted search είναι καινούρια, υπάρχουν ήδη design patters για αυτό.

Το Front End design είναι κάτι ιδιαίτερα πολύπλοκο. Το γεγονός ότι είναι το κομμάτι της εφαρμογής που έρχεται σε επαφή ο χρήστης το κάνει πάρα πολύ σημαντικό. Η παραμικρή δυσκολία που θα παρουσιαστεί ή οτιδήποτε μπερδέψει το χρήστη καθιστά και ένα μειονέκτημα σε σχέση με τον ανταγωνισμό.

3.11. Υλοποίηση Faceted Search εφαρμογής με τη χρήση του Solr

Σε αυτό το κεφάλαιο θα παρουσιάσουμε ένα tutorial για το πώς να φτιάξουμε μια faceted search εφαρμογή χρησιμοποιώντας τη μηχανή της Solr. Θα ξεκινήσουμε από τα απολύτως βασικά και θα φτάσουμε ως την υλοποίηση.

- Download όλων των βασικών libraries/components
- Καταμερισμός των αρχείων
- Directory layouts
- IDE setup
- Ρυθμίσεις
- Συμβουλές
- HTML setup
- Apache Proxying
- Ιστοσελίδας αναφοράς και manuals

3.11.1. Εν συντομία

Η διαδικασία αυτή θα μας οδηγήσει στην υλοποίηση μιας πραγματικής εφαρμογής faceted search.

Ο συγκεκριμένος οδηγός καλύπτει τις απαιτήσεις του γράφοντα οι οποίες μπορεί να διαφέρουν από άλλων προγραμματιστών αλλά θα γίνει προσπάθεια να καλυφθούν τα βασικά ώστε η παραμετροποίηση να είναι εύκολη αργότερα.

3.11.2. Λίστα απαιτήσεων

Για την υλοποίηση της εφαρμογής θα χρησιμοποιηθούν τα παρακάτω περιβάλλοντα:

- Java: Ο Solr server είναι βασισμένος σε java οπότε είναι το αναγκαίο κακό στην προκειμένη περίπτωση
- Native java connection client: Δε θα μας απασχολήσει το XML format λόγο αυτού. Είναι μάλιστα ο κύριος λόγος που θα πρέπει να είναι η ιστοσελίδα μας βασισμένη σε Java.
- Tomcat: Θα χρησιμοποιήσουμε tomcat γιατί θα μας ευκολύνει πολύ με τη java.
- Max OS X deployment: Άμεσος στόχος της εφαρμογής είναι να υλοποιεί μια multiplatform προσέγγιση.

3.11.3. Τι θα κατασκευάσουμε;

Σε πρώτη φάση θα δούμε πάνω κάτω τι εφαρμογές μπορούν να κατασκευαστούν με το Solr:

- iTunes library faceted search
- Recipe Search
- Generic Database search
- RSS feeds
- Images
- Local directory search
- Web based search
- Πολλές άλλες…

Ίσως η παραπάνω λίστα να δείχνει ότι απαιτείται πολύ δουλειά αναλογιζόμενοι το πώς δουλεύει το Solr αλλά στην πραγματικότητα ο χρόνος υλοποίησης που δίνει η κατασκευάστρια εταιρία δεν ξεπερνά τις δύο μέρες.

Επίσης θα καλύψουμε και τα εξής:

- Logging
- User tagging
- Privacy/Authentication result filtering
- Plugins
- Solr PanlScalability/Replication

3.11.4. Εγκατάσταση του Solr

Σε αυτό το κομμάτι θα δούμε πως θα εγκαταστήσουμε το Solr ώστε να το έχουμε ετοιμοπόλεμο για οποιαδήποτε μετέπειτα εφαρμογή φτιάξουμε. Την πρώτη μας εγκατάσταση θα την ονομάσουμε Base Install. Επίσης θα ασχοληθούμε και με source code editing ώστε να είμαστε έτοιμοι για όλες τις εφαρμογές.

3.11.5. Solr download

Μπορείτε να κατεβάσετε την πλατφόρμα του Solr από την εξής διεύθυνση: http://www.apache.org/dyn/closer.cgi/lucene/solr/

Μέχρι στιγμής έχουμε τα εξής από αρχεία:

- Parent Directory/
- 1.3.0/
- *HEADER.html*
- *KEYS*

Στην παρακάτω διεύθυνση μπορείτε να κατεβάσετε τον Apache. http://mirror1.smudge-it.co.uk/apache/lucene/solr/1.3.0/apache-solr-1.3.0.tgz.

Αποσυμπιέστε τους φακέλους και θα έχετε τους εξής φακέλους και αρχεία:

- *CHANGES.txt*
- *KEYS.txt*
- *LICENSE.txt*
- *NOTICE.txt*
- *README.txt*
- *build.xml*
- client/
- *common-build.xml*
- contrib/
- dist/
- docs/
- example/
- lib/
- src/

Τα αρχεία που μας ενδιαφέρουν εδώ είναι τα εξής:

- dist/apache-solr-1.3.0.war
- dist/apache-solr-solrj-1.3.0.jar
- example/solr/conf/solrconfig.xml
- example/solr/conf/schema.xml
- example/solr/conf/protwords.txt
- example/solr/conf/synonyms.txt
- example/solr/conf/stopwords.txt

Το apache-solr-1.3.0.war αρχείο είναι το webapp που θα εγκατασταθεί στον tomcat.

Το apache-solr-solrj-1.3.0.jar θα το χρειαστούμε για την αλληλεπίδραση με τον solr server.

Τα υπόλοιπα αρχεία θα χρειαστούν για την ρύθμιση του Solr server.

3.11.6. Εγκατάσταση Tomcat

Το πρώτο πράγμα που έχουμε να κάνουμε είναι η εγκατάσταση του Apache Tomcat Server.

Εδώ θα χρησιμοποιήσουμε την έκδοση 6.0.18. (http://tomcat.apache.org/download-60.cgi#6.0.18)

Αποσυμπιέζουμε τα αρχεία και μετονομάζουμε τον φάκελο apache-tomcat-6.0.18 σε tomcat-solr. Τοποθετήστε τα αρχεία σε φάκελο της προτίμησης σας. Από εδώ και στο εξής θα αναφερόμαστε σε αυτόν ως **{BASE_TOMCAT_DIRECTORY}**.

3.11.7. Μια ματιά στη δομή των καταλόγων

Κοιτώντας τον **{BASE_TOMCAT_DIRECTORY}** βλέπουμε την εξής δομή καταλόγων ια αρχείων:

- bin/
- conf/
- lib/
- *LICENSE*
- logs/
- *NOTICE*
- *RELEASE-NOTES*
- *RUNNING.txt*
- temp/
- webapps/
- work/

Οι δυο φάκελοι που μας ενδιαφέρουν εδώ είναι οι bin και webapps.

Webapps: Αρχικά ο φάκελος περιέχει τα εξής

- docs/
- examples/
- host-manager/
- manager/
- ROOT/

Σβήνουμε τους φακέλους ώστε να αδειάσει ο φάκελος

Bin: Ο φάκελος εκκίνησης του Tomcat και ο φάκελος που θα τοποθετήσουμε τα αρχεία ρυθμίσεων του Solr. Υπάρχουν διάφοροι τρόποι να ρυθμίσουμε το Solr αλλά εμείς θα προτιμήσουμε αυτόν.

3.11.8. Βασική εγκτάσταση Solr

Σε αυτό το σημείο είμαστε έτοιμοι να συνδυάσουμε τις ρυθμίσεις ώστε να ετοιμάσουμε τον Solr server.

Βήμα 1: Από τον κατάλογο του Solr αντιγράφουμε το αρχείο dist/apache-solr-1.3.0.war στον φάκελο {BASE_TOMCAT_DIRECTORY}/webapps/ directory και μετονομάζουμε τον αρχείο σε solr.war

Βήμα 2: Δημιουργούμε τους παρακάτω φακέλους:

- {BASE_TOMCAT_DIRECTORY}/bin/solr/conf/
- {BASE_TOMCAT_DIRECTORY}/bin/solr/data/

Βήμα 3: Αντιγράφουμε τα παρακάτω αρχεία στον φάκελο /java_servers/tomcat-solr/bin/solr/conf/

- example/solr/conf/solrconfig.xml
- example/solr/conf/schema.xml
- example/solr/conf/protwords.txt
- example/solr/conf/synonyms.txt
- example/solr/conf/stopwords.txt

Σημαντικό: Τα αρχεία που μόλις αντιγράψατε συνιστούν την βασική ρύθμιση του Solr και ανταποκρίνονται σε όλες τις Solr εγκαταστάσεις.

Ο φάκελος του tomcat θα πρέπει να είναι τώρα κάπως έτσι:

- /java_servers/tomcat-solr/LICENSE
- /java_servers/tomcat-solr/NOTICE
- /java_servers/tomcat-solr/RELEASE-NOTES
- /java_servers/tomcat-solr/RUNNING.txt
- /java_servers/tomcat-solr/bin
- /java_servers/tomcat-solr/bin/bootstrap.jar
- /java_servers/tomcat-solr/bin/catalina-tasks.xml
- /java_servers/tomcat-solr/bin/catalina.bat
- /java_servers/tomcat-solr/bin/catalina.sh
- /java_servers/tomcat-solr/bin/commons-daemon.jar
- /java_servers/tomcat-solr/bin/cpappend.bat
- /java_servers/tomcat-solr/bin/digest.bat
- /java_servers/tomcat-solr/bin/digest.sh
- /java_servers/tomcat-solr/bin/jsvc.tar.gz
- /java_servers/tomcat-solr/bin/service.bat

- /java_servers/tomcat-solr/bin/setclasspath.bat
- /java_servers/tomcat-solr/bin/setclasspath.sh
- /java_servers/tomcat-solr/bin/shutdown.bat
- /java_servers/tomcat-solr/bin/shutdown.sh
- **/java_servers/tomcat-solr/bin/solr**
- **/java_servers/tomcat-solr/bin/solr/conf**
- **/java_servers/tomcat-solr/bin/solr/conf/elevate.xml**
- **/java_servers/tomcat-solr/bin/solr/conf/protwords.txt**
- **/java_servers/tomcat-solr/bin/solr/conf/schema.xml**
- **/java_servers/tomcat-solr/bin/solr/conf/solrconfig.xml**
- **/java_servers/tomcat-solr/bin/solr/conf/spellings.txt**
- **/java_servers/tomcat-solr/bin/solr/conf/stopwords.txt**
- **/java_servers/tomcat-solr/bin/solr/conf/synonyms.txt**
- **/java_servers/tomcat-solr/bin/solr/data**
- /java_servers/tomcat-solr/bin/startup.bat
- /java_servers/tomcat-solr/bin/startup.sh
- /java_servers/tomcat-solr/bin/tomcat-juli.jar
- /java_servers/tomcat-solr/bin/tomcat-native.tar.gz
- /java_servers/tomcat-solr/bin/tomcat6.exe
- /java_servers/tomcat-solr/bin/tomcat6w.exe
- /java_servers/tomcat-solr/bin/tool-wrapper.bat
- /java_servers/tomcat-solr/bin/tool-wrapper.sh
- /java_servers/tomcat-solr/bin/version.bat
- /java_servers/tomcat-solr/bin/version.sh
- /java_servers/tomcat-solr/conf
- /java_servers/tomcat-solr/conf/catalina.policy
- /java_servers/tomcat-solr/conf/catalina.properties
- /java_servers/tomcat-solr/conf/context.xml
- /java_servers/tomcat-solr/conf/logging.properties
- /java_servers/tomcat-solr/conf/server.xml
- /java_servers/tomcat-solr/conf/tomcat-users.xml
- /java_servers/tomcat-solr/conf/web.xml
- /java_servers/tomcat-solr/lib
- /java_servers/tomcat-solr/lib/annotations-api.jar
- /java_servers/tomcat-solr/lib/catalina-ant.jar
- /java_servers/tomcat-solr/lib/catalina-ha.jar
- /java_servers/tomcat-solr/lib/catalina-tribes.jar
- /java_servers/tomcat-solr/lib/catalina.jar
- /java_servers/tomcat-solr/lib/el-api.jar
- /java_servers/tomcat-solr/lib/jasper-el.jar
- /java_servers/tomcat-solr/lib/jasper-jdt.jar
- /java_servers/tomcat-solr/lib/jasper.jar
- /java_servers/tomcat-solr/lib/jsp-api.jar
- /java_servers/tomcat-solr/lib/servlet-api.jar
- /java_servers/tomcat-solr/lib/tomcat-coyote.jar
- /java_servers/tomcat-solr/lib/tomcat-dbcp.jar
- /java_servers/tomcat-solr/lib/tomcat-i18n-es.jar
- /java_servers/tomcat-solr/lib/tomcat-i18n-fr.jar

- /java_servers/tomcat-solr/lib/tomcat-i18n-ja.jar
- /java_servers/tomcat-solr/logs
- /java_servers/tomcat-solr/temp
- /java_servers/tomcat-solr/temp/safeToDelete.tmp
- /java_servers/tomcat-solr/webapps
- **/java_servers/tomcat-solr/webapps/solr.war**
- /java_servers/tomcat-solr/work

Με bold είναι φάκελοι και αρχεία που μας ενδιαφέρουν.

Συγχαρητήρια

Αυτή είναι η βασική εγκατάσταση Solr η οποία μπορεί να χρησιμοποιηθεί ξανά και ξανά σε μετέπειτα εφαρμογές. Θα την ονομάσουμε **{BASE_SOLR_INSTALL}.** Το μόνο που θα χρειαστεί να αλλάξουμε είναι τα αρχεία στον φάκελο /java_servers/tomcat-solr/bin/solr/conf directory.

3.11.9. Η εφαρμοφή

Πριν ξεκινήσουμε καλό θα ήταν να εξηγήσουμε κάτι το οποίο θα χρειαστεί να κατανοήσουμε για την αρχιτεκτονική του Solr. Το Solr χρησιμοποιεί ένα μοντέλο client-server. Ο server είναι μια web εφαρμογή. Η σελίδα μας κάνει ένα αίτημα στον server και ανακτά τα αποτελέσματα. Αυτό σημαίνει ότι έχουμε ένα αντίγραφο Solr που μπορεί τρέχει σε διαφορετικό server από ότι ι σελίδα μας. Αξίζει να αναφέρουμε ότι εδώ τίθενται θέματα ασφαλείας αλλά δεν είναι της παρούσης.

Τα αποτελέσματα ανάμεσα στον client και στον server μεταδίδονται με δύο τρόπους.

- XML
- solr-j client

Τα περισσότερα παραδείγματα χρησιμοποιούν XML μηχανισμό. Εμείς στο παράδειγμα μας θα χρησιμοποιήσουμε τον δεύτερο τρόπο ο οποίος είναι και γρηγορότερος.

Για το πρώτη αυτή υλοποίηση μας θα φτιάξουμε μια εφαρμογή που θα δημιουργεί δύο jsp σελίδες μία για να δημιουργεί δεδομένα στον server και μία για να ανακτά faceted search αιτήματα. Αρχικά δεν θα μας απασχολήσει το keyword search αλλά το faceted κομμάτι.

3.11.10. Βασικές ρυθμίσεις Solr

Τώρα που έχουμε μια επίγνωση του τι περίπου κάνουμε, στον φάκελο {BASE_SOLR_INSTALL}/bin/conf/ και θα κάνουμε edit το αρχείο schema.xml. Αυτό το αρχείο ρυθμίζει τα πεδία για την μηχανή αναζήτησης. Θα χρησιμοποιήσουμε μια αρκετά «κομμένη» έκδοση στο παράδειγμα μας.

Ανοίξτε το αρχείο {BASE_SOLR_INSTALL}/bin/conf/schema.xml και διαγράψτε τα πάντα. Θα ορίσουμε τις ρυθμίσεις από την αρχή. Παράλληλα θα εξηγήσουμε την λειτουργία της κάθε γραμμής κώδικα.

Το αρχείο θα είναι χωρισμένο σε τρία μέρη:

1. Field Types,
2. Fields
3. Configuration

Οι παρακάτω γραμμές είναι η πιο απλές ρυθμίσεις που μπορούμε να ορίσουμε:

```
01:<?xml version="1.0" encoding="UTF-8" ?>
02:<schema name="example" version="1.1">
03:  <types>
04:   <fieldType name="string" class="solr.StrField" sortMissingLast="true"
omitNorms="true"/>
05: </types>
06:
07:
08: <fields>
09:   <field name="id" type="string" indexed="true" stored="true" required="true" />
10:   <field name="category" type="string" indexed="true" stored="true"
multiValued="true" omitNorms="true"/>
11:   <field name="size" type="string" indexed="true" stored="true"/>
12:  <field name="text" type="string" indexed="true" stored="false"
multiValued="true"/>
13: </fields>
14:
15: <uniqueKey>id</uniqueKey>
16: <defaultSearchField>text</defaultSearchField>
17: <solrQueryParser defaultOperator="AND"/>
18:
19: <copyField source="id" dest="text"/>
20: <copyField source="category" dest="text"/>
21: <copyField source="size" dest="text"/>
22:
23:</schema>
```

01: Περιγραφή του xml αρχείου
02: Η αρχή του schema element
03: Η αρχή όπου ορίζουμε τους τύπους των πεδίων μας
04: Ορισμός του τύπου string
05: Τέλος του ορισμού των τύπων πεδίων
08: Η αρχή του ορισμού των πεδίων
09: Ο ορισμός του ID field ως string
10: Ο ορισμός του πεδίου category ως string
11: Ο ορισμός του πεδίου size ως string

12: Ο ορισμός του πεδίου text ως string
13: Τέλος ορισμού των πεδίων
15: Το μοναδικό αναγνωριστικό των εγγράφων που θα αποθηκεύεται στο Solr index
16: Το προεπιλεγμένο πεδίο αναζήτησης
17: Ορίζει τον προεπιλεγμένο τελεστή AND ή OR.
19-21: Το element copyField που αντιγράφει πληροφορία από το ένα πεδίο στο άλλο.
23: Το τέλος του schema element.

3.11.11. Εκκινώντας τον server

Τώρα που τα έχουμε όλα έτοιμα είναι ώρα να ξεκινήσουμε τον server. Πηγάινουμε στον φάκελο {BASE_SOLR_INSTALL}/bin/ directory και τρέχουμε το αρχείο catalina.bat
Το παράθυρο θα τυπώσει ανάμεσα σε άλλα και τα εξής:

start up configuration

XX:XX:XX XXX XXX XX $ ./catalina.sh run
Using CATALINA_BASE: /java_servers/tomcat-solr
Using CATALINA_HOME: /java_servers/tomcat-solr
Using CATALINA_TMPDIR: /java_servers/tomcat-solr/temp
Using JRE_HOME: /System/Library/Frameworks/JavaVM.framework/Home
...

Deploying the solr server

...
XXX XX, XXXX XX:XX:XX XX org.apache.catalina.startup.HostConfig deployWAR
INFO: Deploying web application archive solr.war
XXX XX, XXXX XX:XX:XX XX org.apache.solr.servlet.SolrDispatchFilter init
INFO: SolrDispatchFilter.init()
XXX XX, XXXX XX:XX:XX XX org.apache.solr.core.SolrResourceLoader locateInstanceDir
INFO: No /solr/home in JNDI
XXX XX, XXXX XX:XX:XX XX org.apache.solr.core.SolrResourceLoader locateInstanceDir
INFO: solr home defaulted to 'solr/' (could not find system property or JNDI)
XXX XX, XXXX XX:XX:XX XX org.apache.solr.core.CoreContainer$Initializer initialize
INFO: looking for solr.xml: /java_servers/tomcat-solr/bin/solr/solr.xml
XXX XX, XXXX XX:XX:XX XX org.apache.solr.core.SolrResourceLoader <init>
INFO: Solr home set to 'solr/'
...

Setting of the solr home

...

XXX XX, XXXX XX:XX:XX XX org.apache.solr.core.SolrResourceLoader locateInstanceDir
INFO: No /solr/home in JNDI
XXX XX, XXXX XX:XX:XX XX org.apache.solr.core.SolrResourceLoader locateInstanceDir
INFO: solr home defaulted to 'solr/' (could not find system property or JNDI)
XXX XX, XXXX XX:XX:XX XX org.apache.solr.core.SolrResourceLoader <init>
INFO: Solr home set to 'solr/'
...

setting up the data directory and configuration

...
XXX XX, XXXX XX:XX:XX XX org.apache.solr.core.SolrConfig
INFO: Loaded SolrConfig: solrconfig.xml
XXX XX, XXXX XX:XX:XX XX org.apache.solr.core.SolrCore
INFO: Opening new SolrCore at solr/, dataDir=./solr/data/
...

Starting the http listener

...
XXX XX, XXXX XX:XX:XX XX org.apache.coyote.http11.Http11Protocol start
INFO: Starting Coyote HTTP/1.1 on http-8080
...

The server has started and is ready to service incoming requests.

...
XXX XX, XXXX XX:XX:XX XX org.apache.catalina.startup.Catalina start
INFO: Server startup in 2056 ms

Επίσης θα πρέπει να έχουν δημιουργηθεί οι παρακάτω φάκελοι στον {BASE_SOLR_INSTALL}/bin/solr/data

- index
- spellchecker1
- spellchecker2
- spellcheckerFile

3.11.12. Indexing δεδομένων

Τώρα που εκκινήσαμε τον server είναι ώρα να προσθέσουμε πληροφορίες στο Solr index. Για να το κάνουμε αυτό θα δημιουργήσουμε ένα jsp αρχείο και εισάγουμε μερικά snippets μέσα

3.11.13. Ρυθμίζοντας το πλαίσιο ROOT

- Κλείνουμε τον server από το τερματικό

- Κατεβάζουμε αυτό το αρχείο synapticloop.com/static/tomes/solr/tutorial/my-first-faceted-example/ROOT.war και το τοποθετούμε στον φάκελο {BASE_SOLR_INSTALL}/webapps/
- Επανεκκινούμε τον server.

3.11.14. Προσθήκη δεδομένων για indexing

Το αρχείο ROOT.war έχει πολλά αρχεία και φακέλους αλλά για την ώρα θα ασχοληθούμε με το add-data-1.jsp. Πέρα από τις κλάσεις, τις μεθόδους και τον κώδικα HTML μας ενδιαφέρουν οι παρακάτω 21 γραμμές κώδικα:

```
01: SolrServer solrServer = new
CommonsHttpSolrServer("http://localhost:8080/solr/");
02:
03: out.print("<ul>");
04:
05: for(int i = 0; i < 100; i++) {
06:         SolrInputDocument solrInputDocument = new SolrInputDocument();
07:         solrInputDocument.addField("id", new String("widget " + i));
08:         // add three random categories
09:         for(int j = 0; j < 3; j++) {
10:                 solrInputDocument.addField("category", getRandomCategory());
11:         }
12:         solrInputDocument.addField("size", getRandomSize());
13:         // this is cheating below - but saves us from the query string...
14:         solrInputDocument.addField("text", "a");
15:         solrServer.add(solrInputDocument);
16:         out.print("<li> Adding: " + solrInputDocument + "</li>\n");
17: }
18:
19: out.print("</ul>");
20:
21: solrServer.commit();
```

01: Συνδεόμαστε μέσω HTTP στον server.
03: Snippet που έχει σαν έξοδο HTML κώδικα.
05: Θα εισάγουμε 100 έγγραφα στο ευρετήριο χρησιμοποιώντας loops.
06: Δημιουργία νέου SolrInputDocument ώστε να αρχίσουμε να εισάγουμε πληροφορία
07: Προσθήκη πεδίου στο έγγραφο. Συγκεκριμένα θα εισάγουμε το πεδίο ID
08-11: Προσθήκη κατηγοριών στο έγγραφο. Εδώ προσθέτουμε τρείς τυχαίες κατηγορίες που μας επιτρέπουν επιπρόσθετη αλληλεπίδραση με το περιβάλλον αναζήτησης
12: Προσθήκη πεδίου size
13-14: Εδώ «κλέβουμε» λίγο για να γλιτώσουμε την παραμετροποίηση του query. Προσθέτουμε ένα χαρακτήρα στο έγγραφο μας και συγκεκριμένα στο πεδίο text.

15: Προσθήκη του εγγράφου στο SolrServer
16-19: Περισσότερη HTML
21: Αποστολή αποτελεσμάτων στο Server.

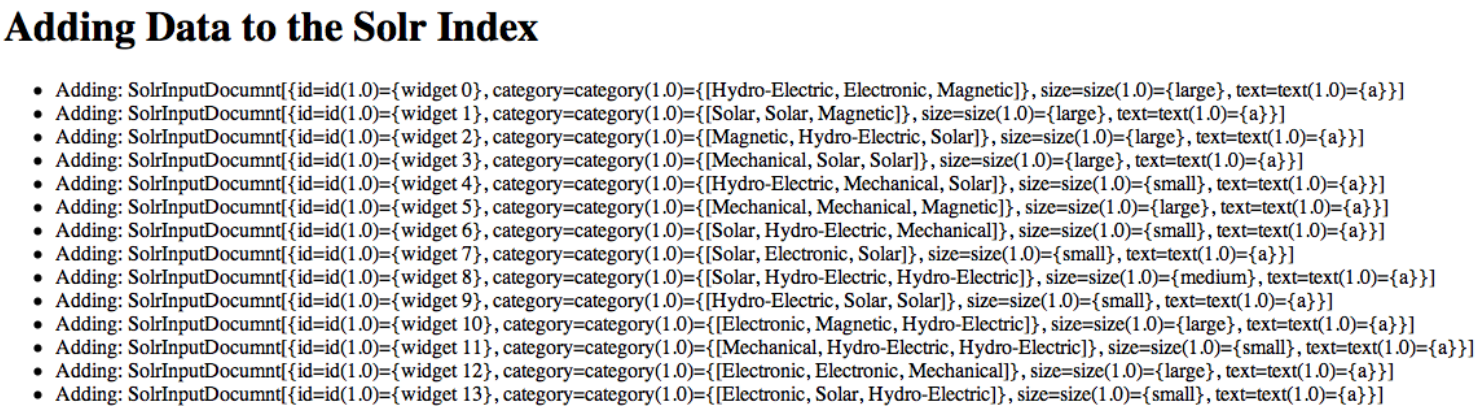
Ούτε λίγο ούτε πολύ με αυτές τις 21 γραμμές κώδικα έχουμε προσθέσει δεδομένα στο ευρετήριο μας και είμαστε έτοιμοι να τα αναζητήσουμε.

3.11.15. Αναζήτηση δεδομένων

Θα ρίξουμε μια ματιά τώρα σε μια σελίδα με απλά αποτελέσματα. Αυτά τα αποτελέσματα μπορούμε να τα βρούμε στο αρχείο search-1.jsp. Αυτό το αρχείο είναι πιο πολύπλοκο για αυτό θα αρκεστούμε στην σελίδα.

Ο κώδικας είναι λίγο «δύσκολος» αλλά σχεδιάστηκε για να είναι ταχύς και εύκολα προσβάσιμος.

Βεβαιωνόμαστε ότι ο server είναι σε λειτουργία και ανοίγουμε την εξής σελίδα: http://localhost:8080/add-data-1.jsp. Η σελίδα θα φαίνεται κάπως έτσι: (οι κατηγορίες ίσως είναι διαφορετικές ανά υλοποίηση γιατί δημιουργούνται τυχαία)



Εικόνα 36 - Υλοποίηση faceted search εφαρμογής με τη χρήση του Solr

3.11.16. Queries και Server

26 γραμμές κώδικα για τα ερωτήματα μας στον server. Δεν είναι τόσο άσχημα αν αναλογιστούμε τις πολλές επιλογές που έχουμε για το πώς θα επιστρέφονται τα αποτελέσματα.

```
01: SolrQuery solrQuery = new SolrQuery();
02: solrQuery.setFacet(true);
03: solrQuery.setFacetMinCount(1);
04: solrQuery.setRows(new Integer(100));
05: solrQuery.setStart(new Integer(0));
```

```
06: solrQuery.addFacetField("category");
07: solrQuery.addFacetField("size");
08:
09: // here I am definitely cheating!
10: solrQuery.setQuery("a");
11:
12: Map parameterMap = request.getParameterMap();
13: Set keySet = parameterMap.keySet();
14: Iterator parameterMapIterator = keySet.iterator();
15: while(parameterMapIterator.hasNext()) {
16:         String key = (String)parameterMapIterator.next();
17:         String[] values = (String[])parameterMap.get(key);
18:         for(int i = 0; i < values.length; i++) {
19:                 solrQuery.addFilterQuery(key + ":" + values[i]);
20:         }
21: }
22:
23: SolrServer solrServer = new
CommonsHttpSolrServer("http://localhost:8080/solr/");
24:
25: QueryResponse queryResponse = solrServer.query(solrQuery);
26: pageContext.setAttribute("queryResponse", queryResponse);
```

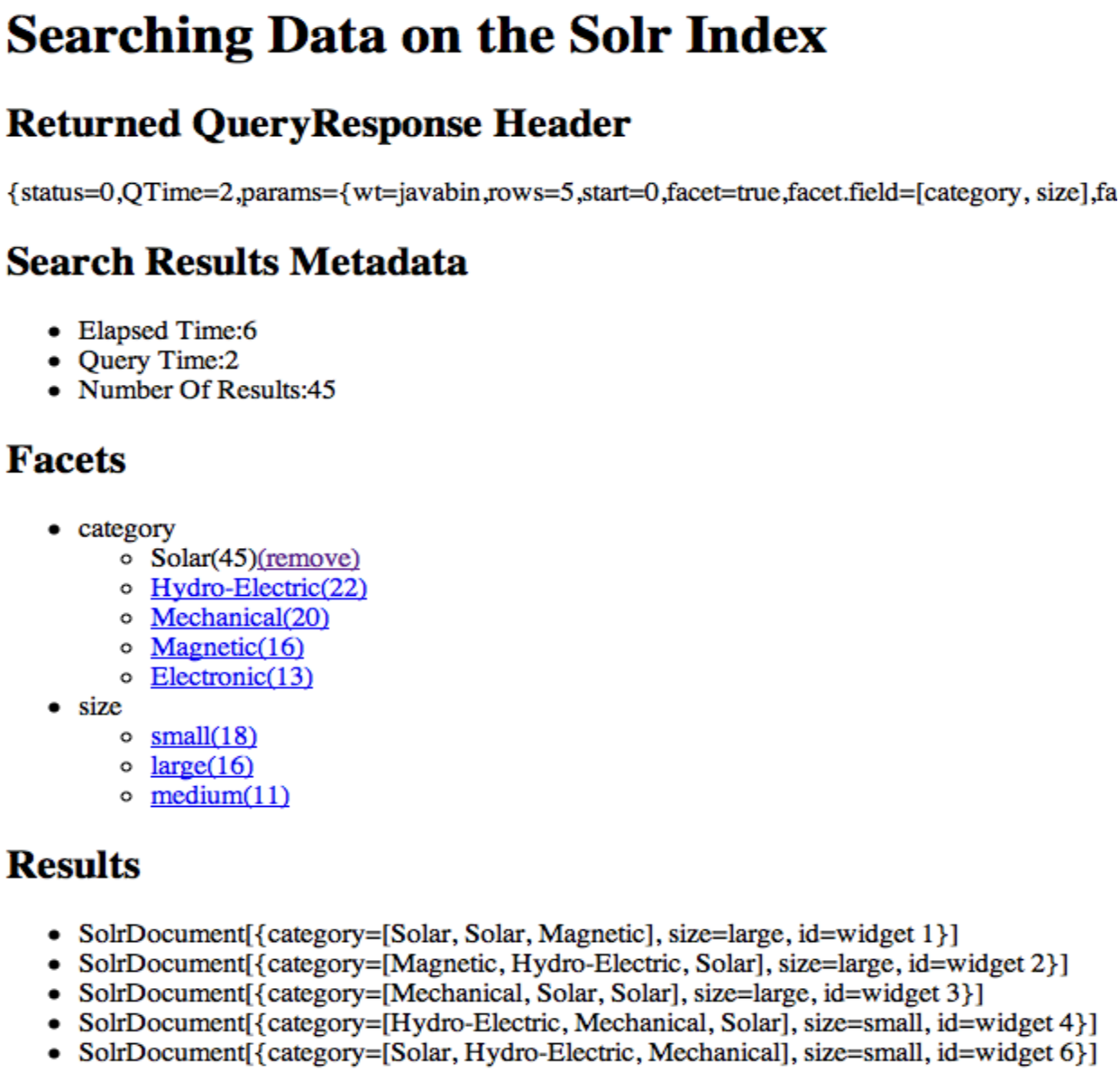
01:Δημιουργία νέου SolrQuery αντικειμένου
02: Ενεργοποίηση faceting
03: Ορισμός ελάχιστου αριθμού facet hits που απαιτούνται ώστε να επιστραφεί το facet. Ακούγεται περίπλοκο αλλά ουσιαστικά αυτή η γραμμή μας λέει με λίγα λόγια ότι αν το facet εμφανίζεται μηδέν φορές να μην εμφανιστεί.
04: Ορισμός του αριθμού των γραμμών που θα επιστραφούν ανά ερώτημα.
05: Ορισμός του αποτελέσματος που θα επιστραφεί πρώτο.
06-07: Ορισμός των facet πεδίων που θέλουμε να επιστραφούν μαζί με τα αποτελέσματα.
09-10: Εδώ βρίσκει εφαρμογή το «κλέψιμο» που εξηγούσαμε πριν. Υπάρχουν αρκετές λύσεις για αυτό αλλά εδώ προτιμούμε την συγκεκριμένη.
12-21: Ορισμός των παραμέτρων του ερωτήματος.
23: Σύνδεση με τον Solr server
25: Η ανταπόκριση στο ερώτημα μας.
26: Αποθήκευση του αποτελέσματος για μελλοντική χρήση.

Όπως είδαμε πρόκειται για πολύ απλές και άμεσες γραμμές κώδικα που εξυπηρετούν τα ερωτήματα προς τον Server. Επόμενη μας κίνηση θα είναι να παρουσιάσουμε αυτά τα δεδομένα.

Τώρα που είδαμε τον κώδικα ανοίγουμε την σελίδα http://localhost:8080/search-data-1.jsp

Η σελίδα θα πρέπει να είναι σαν την παρακάτω εικόνα. Έχει ήδη επιλεχτεί η κατηγορία "Solar" η οποία έχει και ένα σύνδεσμο που την αφαιρεί δίπλα της. Κάντε κλικ στα διάφορα links και παρατηρήστε πως αλλάζουν τα αποτελέσματα.

Έχουμε φτιάξει μια πρώτη σελίδα που υλοποιεί το faceted search πάντα με τη βοήθεια της Solr πλατφόρμας. Λείπουν πάρα πολλά κομμάτια ακόμα αλλά μπορείτε να πάρετε μια ιδέα για το πώς περίπου λειτουργεί η μηχανή και με τη σειρά σας να δοκιμάσετε δικές σας λύσεις. Παρακάτω παρατίθεται μια λίστα μεθόδων πολύ χρήσιμων ιδιαίτερα στην ανάκτηση δεδομένων.



Εικόνα 37 - Υλοποίηση εφαρμογής με τη χρήση του Solr

3.12. Παρουσίαση μεθόδων για την εξαγωγή δεδομένων

3.12.1. Parsing QueryResponse Object

Θεωρούμε ότι έχουμε ένα αντικείμενο QueryResponce που το ονομάζουμε queryResponce. Οι παρακάτω μέθοδοι χρησιμοποιούνται για να ανακτήσουμε αντικείμενα.

Μέθοδος	**Τύπος**	**Παράδειγμα**
getElapsedTime()	long	186
Σημείωση: Ο χρόνος για την πληρης εκτέλεση του ερωτήματος.		
getQTime()	int	97
Σημείωση: Ο χρόνος που χρειάζεται το Solr για την εκτέλεση του ερωτήματος.		
getStatus()	int	0
Σημείωση: Το 0 σημαίνει ότι όλα οκ ειδάλλως το 500 θα σημαίνει ότι υρήρξε σφάλμα.		
getFacetDates()	List<FacetField>	n/a
Σημείωση: Επιστρέφει όλα τα facets που βασίζονται σε ημερομηνία.		
getFacetDate(String name)	FacetField	n/a
Σημείωση: Η μέθοδος επιστρέφει ένα συγκεκριμένο date facet με όνομα "name" ή null αν δεν υπάρχει.		
getFacetFields()	List<FacetField>	[category:[Magnetic (58), Mechanical (25), Solar (24), Hydro-Electric (22), Electronic (18)], size:[large (21), medium (20), small (17)]]
Σημείωση: Επιστρέφει όλα τα διαθέσιμα facets ακόμα και αυτά που είναι εν ενεργεία σε φίλτρα.		
getFacetField(String	FacetField	category:[Magnetic (58), Mechanical (25), Solar (24), Hydro-Electric (22), Electronic

Μέθοδος	**Τύπος**	**Παράδειγμα**
name)		(18)]
Σημείωση: Επιστρέφει επιστρέφει ένα συγκεκριμένο facet με όνομα "name" ή null αν δεν υπάρχει.		
getLimitingFacets()	List<FacetField>	[category:[Mechanical (25), Solar (24), Hydro-Electric (22), Electronic (18)], size:[large (21), medium (20), small (17)]]
Σημείωση: Επιστρέφει όλα τα πεδία που απομένουν και μπορούν να περιορίσουν τα αποτελέσματα του ερωτήματος.		
getResponse()	NamedList<Object>	{responseHeader={status=0, QTime=1, params={... **rows=5**, **start=0**, facet=true, facet.field=[category, size], facet.mincount=1, **fq=category:Magnetic**, q=a, ...}}, response={numFound=58, start=0, docs=[...}, facet_counts={...}}
Σημείωση: Αυτή η μέθοδος χρησιμοποιείται για debugging.		
getResponseHeader()	NamedList	{status=0, QTime=1, params={wt=javabin, rows=5, start=0, facet=true, facet.field=[category, size], facet.mincount=1, fq=category:Magnetic, q=a, version=2.2}}
Σημείωση: Επιστρέφει την κεφαλίδα		
getResults()	SolrDocumentList	{numFound=58, start=0, docs=[SolrDocument[{category=[Magnetic, Solar, Mechanical], size=medium, id=widget 1}], SolrDocument[{category=[Solar, Magnetic, Solar], size=large, id=widget 3}],

Μέθοδος	Τύπος	Παράδειγμα
		SolrDocument[{category=[Solar, Magnetic, Mechanical], size=large, id=widget 5}], SolrDocument[{category=[Magnetic, Magnetic, Hydro-Electric], size=small, id=widget 7}]

Σημείωση: Επιστρέφει όλα τα έγγραφα των αποτελεσμάτων καθώς και ορισμένα metadata.

3.12.2. Parsing FacetField Object

Εδώ θα χρησιμοποιήσουμε την μέθοδο queryResponce.getFacetField("category") για να εξάγουμε ένα facet από μια κατηγορία. Θεωρώντας λοιπόν ότι έχουμε ένα FacetField αντικείμενο με όνομα facetField οι ακόλουθες μέθοδοι είναι διαθέσιμες προς χρήση.

Μέθοδος	Τύπος	Παράδειγμα
getName()	String	category

Σημείωση: Το όνομα του facet πεδίου.

getValueCount()	int	5

Σημείωση: Ο αριθμός των τιμών για το εν λόγω facet.

getValues()	List<FacetField.Count>	[Magnetic (58), Mechanical (25), Solar (24), Hydro-Electric (22), Electronic (18)]

Σημείωση: Επιστρέφει όλα τα ονόματα και hits που είναι αποθηκευμένα σε ένα Facet.Count

---

3.12.3. Parsing Facet.Count Object

Το αντικείμενο Facet.Count περιέχει λεπτομέρειες για το επιλεγμένο facet. Μια απλή χρήση του γίνεται με τις παρακάτω γραμμές κώδικα:

```
List<FacetField.Count> facetFieldCounts = facetField.getValues();
```

```
Iterator<FacetField.Count> facetFieldCountIterator = facetFieldCounts.iterator();
while(facetFieldCountIterator.hasNext()) {
        FacetField.Count count = facetFieldCountIterator.next();

        // do something with the information
        String name = count.getName();
        String count = count.getCount();
}
```

3.12.4. Parsing SolrDocumentList Object

Το αντικείμενο SolrDocumentList περιέχει όλα τα έγγραφα τα οποία ικανοποιούν τα κριτήρια μας.

```
SolrDocumentList solrDocumentList = queryResponse.getResults();
Iterator<SolrDocument> solrDocumentIterator =  solrDocumentList.iterator();
while(solrDocumentIterator.hasNext()) {
        SolrDocument solrDocument = solrDocumentIterator.next();
        // do something useful here
}
```

Ορισμένες μέθοδοι που μπορείτε να χρησιμοποιήσετε με το αντικείμενο SolrDocument:

Μέθοδος	**Τύπος**	**Παράδειγμα**
getFieldNames()	Collection<String>	[category, size, id]

Σημείωση: Επιστρέφει όλα τα αντικείμενα που επιστράφηκαν με τα αποτελέσματα.

---

getFieldValue(String name)	Object	[Hydro-Electric, Mechanical, Magnetic]

Σημείωση: Επιστρέφει ένα java.util.ArrayList αντικείμενο με όλες τις τιμές για ένα συγκεκριμένο πεδίο

Σε αυτό το σημείο να αναφέρουμε ότι οι δυνατότητες που μας παρέχει η πλατφόρμα του Solr και γενικότερα οποιαδήποτε πλατφόρμα κυκλοφορεί αυτή τη στιγμή, είναι πάρα πολλές. Με λίγη φαντασία και προσεκτική κατανομή πόρων και design πετυχαίνουμε πάρα πολύ καλά αποτελέσματα. Ισχύει και εδώ όπως και σε κάθε εφαρμογή το ότι πρέπει πάντα να κοιτάμε την κατασκευή από την σκοπιά του

χρήστη. Τι θα κάναμε εμείς αν μας δινόταν να χρησιμοποιήσουμε μια τέτοια εφαρμογή; Τι σενάρια χρήσης θα παρουσιαζόντουσαν; Τα έχουμε προβλέψει όλα; Τέτοιου τύπου ερωτήσεις μας βοηθάνε να προβλέψουμε καταστάσεις ώστε να αποφύγουμε να τις αντιμετωπίσει ο χρήστης.

4. Drupal και Faceted Search

Η πτυχιακή συνοδεύεται από μια υλοποίηση ιστοσελίδας με βάση το γνωστό CMS (Content Management System) Drupal. Το εν λόγω CMS παρέχει ένα από τα πιο ολοκληρωμένα components για faceted search το οποίο παρέχει πολλές παραμετροποιήσεις. Αυτοί ήταν οι λόγοι που οδήγησαν στην προτίμηση του. Παρακάτω θα γίνει μια αναφορά στον τρόπο λειτουργίας τους και στην καθαυτή λειτουργία του component.

4.1. Drupal: Γενικά

Το **Drupal** είναι ένα αρθρωτό σύστημα διαχείρισης περιεχομένου (Content Management System, CMS) ανοικτού/ελεύθερου λογισμικού, γραμμένο στη γλώσσα προγραμματισμού PHP. Το Drupal, όπως πολλά σύγχρονα CMS, επιτρέπει στο διαχειριστή συστήματος να οργανώνει το περιεχόμενο, να προσαρμόζει την παρουσίαση, να αυτοματοποιεί διαχειριστικές εργασίες και να διαχειρίζεται τους επισκέπτες του website και αυτούς που συνεισφέρουν. Παρόλο που υπάρχει μια πολύπλοκη προγραμματιστική διεπαφή, οι περισσότερες εργασίες μπορούν να γίνουν με λίγο ή και καθόλου προγραμματισμό. Το Drupal ορισμένες φορές περιγράφεται ως "υποδομή για εφαρμογές ιστού", καθώς οι δυνατότητές του προχωρούν παραπέρα από τη διαχείριση περιεχομένου, επιτρέποντας ένα μεγάλο εύρος υπηρεσιών και συναλλαγών.

Το Drupal μπορεί να εκτελεστεί σε διάφορες πλατφόρμες, συμπεριλαμβανομένων των λειτουργικών συστημάτων Windows, Mac OS X, Linux, FreeBSD, ή οποιασδήποτε πλατφόρμας που υποστηρίζει είτε το διακομιστή ιστοσελίδων Apache HTTP Server (έκδοση 1.3+), είτε το Internet Information Services (έκδοση IIS5+), καθώς επίσης και τη γλώσσα προγραμματισμού PHP (έκδοση 4.3.3+). Το Drupal απαιτεί μια βάση δεδομένων όπως η MySQL και η PostgreSQL για την αποθήκευση του περιεχομένου και των ρυθμίσεών του.

4.1.1. Ιστορία

Αρχικά γραμμένο από τον Dries Buytaert ως σύστημα πίνακα ανακοινώσεων (*BBS, bulletin board system*), το Drupal μετατράπηκε σε εγχείρημα ανοικτού κώδικα το 2001. *Drupal* είναι η διατύπωση στην Αγγλική γλώσσα της Ολλανδικής λέξης "druppel", που σημαίνει "σταγόνα". Το όνομα πάρθηκε από το ξεπερασμένο πλέον website Drop.org, του οποίου ο κώδικας εξελίχθηκε στο Drupal. Ο Buytaert ήθελε να ονομάσει τον ιστοτόπο "dorp" (στα Ολλανδικά σημαίνει "χωριό", αναφερόμενος στη διάσταση της κοινότητας), αλλά έκανε ένα ορθογραφικό λάθος κατά τη διαδικασία ελέγχου του ονόματος χώρου (domain name) και τελικά σκέφτηκε ότι ακούγεται καλύτερα.

Από το Μάιο του 2006 ως τον Απρίλιο του 2007, χρήστες κατέβασαν το Drupal από το επίσημο website περισσότερες από 600.000 φορές. Μια μεγάλη κοινότητα χρηστών λαμβάνει πλέον μέρος στη συνεχή εξέλιξη του Drupal.

4.1.2. Σχεδίαση

Το Drupal έχει λάβει επαίνους από τους διαχειριστές ιστοσελίδων, σχεδιαστές και προγραμματιστές για τον αρθρωτό σχεδιασμό του, που παρέχει το βασικό του στρώμα, ή "πυρήνα", να παρέχει τα βασικά χαρακτηριστικά του Drupal στην προεπιλεγμένη εγκατάστασή του. Πρόσθετα χαρακτηριστικά λειτουργικότητας και παρουσίασης μπορούν να επεκταθούν στον πυρήνα με την πρόσθεση προσαρτώμενων μονάδων και θεματικών παραλλαγών.

Οι μονάδες του Drupal χρησιμοποιούνται για να "υπερβούν" τα ενσωματωμένα χαρακτηριστικά του πυρήνα, επεκτείνοντας έτσι ή και αντικαθιστώντας την εξ' ορισμού συμπεριφορά του Drupal, χωρίς την επέμβαση στον αυτούσιο κώδικα των αρχείων του πυρήνα του Drupal. Η δυνατότητα αυτή της τροποποίησης της λειτουργικότητας του πυρήνα έχει επίπτωση στην προσαρμοστικότητα του Drupal καθώς και στην ασφάλειά του, ειδικότερα σε θέματα ασφαλείας, όπως η έγχυση εντολών SQL (SQL injection).

Προσαρμοσμένες θεματικές παραλλαγές, που μπορούν να προστεθούν χωρίς να επηρεάζουν τον πυρήνα του Drupal, χρησιμοποιούν προτυποποιημένες μορφές που μπορούν να δημιουργηθούν από μηχανές σχεδίασης θεματικών παραλλαγών τρίτων.

Το **Drupal** έχει ένα βασικό στρώμα, ή πυρήνα, που παρέχει τα βασικά χαρακτηριστικά του Drupal και υποστηρίζει αρθρωτές μονάδες που προσθέτουν επιπλέον λειτουργικότητα ή χαρακτηριστικά.

Οι μονάδες που περιέχονται στον πυρήνα του Drupal παρέχουν στους χρήστες τη δυνατότητα να υποβάλλουν, να αναθεωρούν, να κατηγοριοποιούν ύλη, να εκτελούν αναζητήσεις, να υποβάλλουν σχόλια, να λαμβάνουν μέρος σε φόρουμ συζητήσεων, να ψηφίζουν σε ψηφοφορίες και να δουλεύουν σε συνεργατικά έργα, χωρίς την απαίτηση να γνωρίζουν HTML. Οι μονάδες του πυρήνα επιτρέπουν επίσης στους χρήστες να υποβάλλουν και να βλέπουν προσωπικά προφίλ, να επικοινωνούν μεταξύ τους ή και με τους διαχειριστές του website.

Το σύστημα διαχείρισης εκδόσεων του Drupal, επίσης ένα χαρακτηριστικό του πυρήνα, παρακολουθεί τις αλλαγές του περιεχομένου της ύλης, το ποιος άλλαξε κάτι, τι άλλαξε, την ημερομηνία και ώρα της αλλαγής κ.ο.κ. Το σύστημα παρέχει ένα ημερολόγιο με σχόλια αλλαγών και παρέχει τη δυνατότητα για μετάβαση του περιεχομένου σε προηγούμενη έκδοση.

Πρόσθετες επίσης στον πυρήνα του Drupal είναι μονάδες που επιτρέπουν τους διαχειριστές του website την αλλαγή της εμφάνισής του με έτοιμες ή φτιαγμένες με το χέρι θεματικές παραλλαγές, τη δημιουργία μενού με πολλά επίπεδα και την παροχή στους χρήστες μιας διεπαφής στη μητρική τους γλώσσα. Ακόμη, ο πυρήνας του Drupal επιτρέπει στους διαχειριστές να παρέχουν ροές RSS, καθώς και τη συλλογή περιεχομένου από ροές RSS άλλων websites.

Άλλες μονάδες του πυρήνα παρέχουν την εγγραφή χρηστών, τον καθορισμό ρόλων χρηστών από τους διαχειριστές, με τον ορισμό αδειών (permissions) στους χρήστες

για τη χρησιμοποίηση επιλεγμένων χαρακτηριστικών του website. Οι διαχειριστές μπορούν επίσης να χρησιμοποιούν κανόνες πρόσβασης για την άρνηση πρόσβασης στο website σε συγκεκριμένα ονόματα χρηστών, διευθύνσεις ηλεκτρονικού ταχυδρομείου και διευθύνσεις IP.

Ο πυρήνας του Drupal περιλαμβάνει το χαρακτηριστικό "ψευδώνυμο URL" που επιτρέπει τη δημιουργία φιλικών στο χρήστη, εύκολων προς απομνημόνευση διευθύνσεων URL, είτε με αυτόματο τρόπο, είτε ένας χρήστης να καθορίζει διευθύνσεις URL με την ιδιότητα του συντάκτη ή διαχειριστή.

Οι μονάδες του πυρήνα παρέχουν στατιστικά και αναφορές για τους διαχειριστές, ενώ τους επιτρέπουν να χειρίζονται θέματα λανθάνουσας μνήμης και απόπνιξης ώστε να βελτιώσουν την απόδοση του website σε περιόδους μεγάλης κίνησης.

Οι διαχειριστές μπορούν να κατασκευάζουν και να καθορίζουν διάφορα φίλτρα εισόδου και μορφότυπους ύλης.

Οι χρήστες και οι διαχειριστές μπορούν να εκμεταλλεύονται τα χαρακτηριστικά αυτά, χωρίς να χρειάζεται να γνωρίζουν PHP ή HTML.

4.1.3. Μονάδες

Ο πυρήνας του Drupal έχει σχεδιαστεί βάση ενός συστήματος από hook, ή callback, που επιτρέπει στις γραμμένες από την κοινότητα μονάδες να εισάγουν συναρτήσεις στο μονοπάτι εκτέλεσης του Drupal.

Οι μονάδες που περιέχονται στον πυρήνα του Drupal παρέχουν στους χρήστες τη δυνατότητα να:

- υποβάλλουν, αναθεωρούν και να κατηγοριοποιούν την ύλη
- εκτελούν αναζητήσεις
- υποβάλλουν σχόλια
- λαμβάνουν μέρος σε φόρουμ συζητήσεων
- ψηφίζουν σε ψηφοφορίες
- δουλεύουν σε συνεργατικά συγγραφικά έργα
- τροποποιούν και να βλέπουν προσωπικά προφίλ
- επικοινωνούν μεταξύ τους ή και με τους διαχειριστές του website
- αλλάζουν την εμφάνιση του website με έτοιμες ή φτιαγμένες με το χέρι θεματικές παραλλαγές
- δημιουργούν μενού με πολλαπλά επίπεδα
- βλέπουν τη γραφική διεπαφή και τα μηνύματα στην μητρική τους γλώσσα
- παρέχουν ροές RSS
- συλλέγουν περιεχόμενο από ροές RSS άλλων websites
- εγγράφονται ως χρήστες και να διαχειρίζονται τους λογαριασμούς τους
- καθορίζουν ρόλους χρηστών, με τον ορισμό αδειών (permissions) στους χρήστες για τη χρησιμοποίηση επιλεγμένων χαρακτηριστικών του ιστοτόπου

- χρησιμοποιούν κανόνες πρόσβασης για την άρνηση πρόσβασης στο website σε συγκεκριμένα ονόματα χρηστών, διευθύνσεις ηλεκτρονικού ταχυδρομείου και διευθύνσεις IP
- λαμβάνουν στατιστικά και αναφορές για τους διαχειριστές
- χειρίζονται θέματα λανθάνουσας μνήμης και απόπνιξης ώστε να βελτιώσουν την απόδοση του website σε περιόδους μεγάλης κίνησης
- δημιουργούν και καθορίζουν διάφορα φίλτρα εισόδου και μορφότυπους ύλης
- δημιουργούν φιλικές στο χρήστη, εύκολες προς απομνημόνευση, διευθύνσεις URL (πχ. "www.mysite.com/products" αντί για "www.mysite.com/?q=node/432")

Επιπρόσθετα, το website του Drupal παρέχει εκατοντάδες δωρεάν μονάδες γραμμένες από την κοινότητα του Drupal, που παρέχουν:

- δυνατότητες συστημάτων ηλεκτρονικού εμπορίου (e-commerce)
- χαρακτηριστικά ροής εργασιών
- γκαλερί φωτογραφιών
- σελίδες ομάδων ατόμων (οργανικά γκρουπ)
- χάρτες website για το Google
- αντικείμενα Amazon
- διαχείριση λιστών ηλεκτρονικού ταχυδρομείου
- ένα σύστημα διαχείρισης των συστατικών στοιχείων σχέσεων (Customer relationship management, CRM), το CiviCRM
- ενσωμάτωση με ένα "Concurrent Versions System" (CVS).

4.1.4. Παραδείγματα χρήσης του Drupal

Μερικοί από τους ρόλους που έχει αναλάβει το Drupal είναι εταιρικά intranet, on-line τάξεις, κοινότητες με θέμα τις τέχνες και διαχείριση έργων:

- Η βιβλιοθήκη Ann Arbor District Library χρησιμοποίησε το Drupal για τη δημιουργία ενός βραβευμένου website, συμπεριλαμβάνοντας προσαρμοσμένα χαρακτηριστικά που προστέθηκαν, όπως η δημιουργία προσωποποιημένων καρτών καταλόγων από τους υπαλλήλους της βιβλιοθήκης.
- Aspedia - the Web Company, μια αυστραλιανή εταιρία ανάπτυξης σε Drupal και παροχής φιλοξενίας ιστοσελίδων.
- Διάφορες καμπάνιες πολιτικών, όπως αυτή του *Jack Carter για Γερουσιαστής* στη Νεβάδα, για τις οποίες δημιουργήθηκαν αρκετά website βασισμένοι σε Drupal.
- CiviCRM, ένα συστατικό στοιχείο για ένα σύστημα διαχείρισης σχέσεων πελατών που ενσωματώνεται με το Drupal και το έχει μετατρέψει σε μια δημοφιλή πλατφόρμα για μη κερδοσκοπικούς οργανισμούς.
- Το Drupal χρησιμοποιήθηκε για τη δημιουργία τοπικών δημοσιογραφικών ιστοτόπων για πόλεις όπως το Bluffton, South Carolina και το Watertown, Massachusetts.

- Το Drupal χρησιμοποιήθηκε για τη δημιουργία του τύπου 43things ιστοτόπου Change Everything.
- Με τον ερχομό της μονάδας Revision Moderation, η δημιουργία εφαρμογών όπως σχολικοί ιστοτόποι που έχουν τάξεις τύπου "Advanced Web" γίνεται πιο εφικτή.

4.2. Το faceted search component

Το Faceted search component του Drupal παρέχει μια εφαρμογή αναζήτησης παράλληλα με ένα interface το οποίο επιτρέπει να πλοηγηθούν στην ιστοσελίδα εκμεταλλευόμενοι όλα τα πλεονεκτήματα που προσφέρει το faceted search. Το συγκεκριμένο εργαλείο μάλιστα, είναι από τα πιο ολοκληρωμένα κάνοντας πολύ εύκολη τη ζωή του προγραμματιστή καθώς και του χρήστη.

Το interface δομεί τα μεταδεδομένα με τέτοιο τρόπο έτσι ώστε οι χρήστες να μπορούν να δημιουργήσουν τα ερωτήματα τους κατά τη διάρκεια της πλοήγησης τους ή να επεκτείνουν τα ήδη ενεργά. Τα αποτελέσματα αντανακλούν το τρέχον ερώτημα καθ' όλη τη διάρκεια της πλοήγησης. Η εφαρμογή παρέχει φυσικά και αναζήτηση με keywords σε απλό κείμενο, κάνοντας χρήση της μηχανής του Drupal. Γενικότερα έχουν αποφευχθεί πολύπλοκες φόρμες και ποτέ δε θα δούμε facets που οδηγούν σε μηδενικά αποτελέσματα.

Τα πιο προφανή μεταδεδομένα παρέχονται από το taxonomy module του Drupal. Παρόλα αυτά το ίδιο το component παρέχει στους προγραμματιστές την δυνατότητα να χειριστούν και να δημιουργήσουν τα δικά τους μεταδεδομένα.

4.2.1. Γιατί να χρησιμοποιήσουμε faceted search με το Drupal

Έχουμε αναφερθεί επανειλημμένες φορές στα πλεονεκτήματα του Faceted search. Αυτή τη φορά όμως θα του πλέξουμε ξανά το εγκώμιο δικαιολογώντας γιατί προτιμούμε τη χρήση του με το Drupal.

Παρακάτω αναφέρονται ορισμένοι λόγοι που θα σας ωθήσουν στη χρήση του:

- Οι χρήστες θέλουν να φιλτράρουν περιεχόμενο χρησιμοποιώντας πολλά facets ταυτόχρονα
- Οι χρήστες θέλουν να συνδυάσουν αναζήτηση κειμένου, facets και άλλων κριτηρίων αναζήτησης ταυτόχρονα
- Οι χρήστες ίσως να μην ξέρουν ακριβώς τι μπορούν να βρούνε στην ιστοσελίδα σας ή τι να ψάξουν
- Επιθυμείτε να ωθήσετε τους χρήστες στην αναζήτηση περιεχομένου το οποίο αρχικά δεν είχαν προσέξει αλλά ίσως τελικά τους ενδιαφέρει.
- Επιθυμείτε να δείξετε τις θεματικές ενότητες ή τα είδη των προιόντων που περιέχει η ιστοσελίδα σας.

- Θέλετε να ανακαλύψετε και να συνδυάσετε σχέσεις μεταξύ των διάφορων κειμένων σας.
- Υπερβολικός όγκος περιεχομένου στην ιστοσελίδα. Θυμίζουμε εδώ ότι η ομαδοποίηση των facets βοηθάει πάρα πολύ στη λύση του information overload.
- Χρήση faceted classification.
- Αποφυγή μηδενικών αποτελεσμάτων
- Αποφυγή πολύπλοκων φορμών αναζήτησης που δυσκολεύουν τους χρήστες.

4.2.2. Περιεχόμενα του Component

Το Faceted search component αποτελείται ουσιαστικά από διάφορες μονάδες (modules):

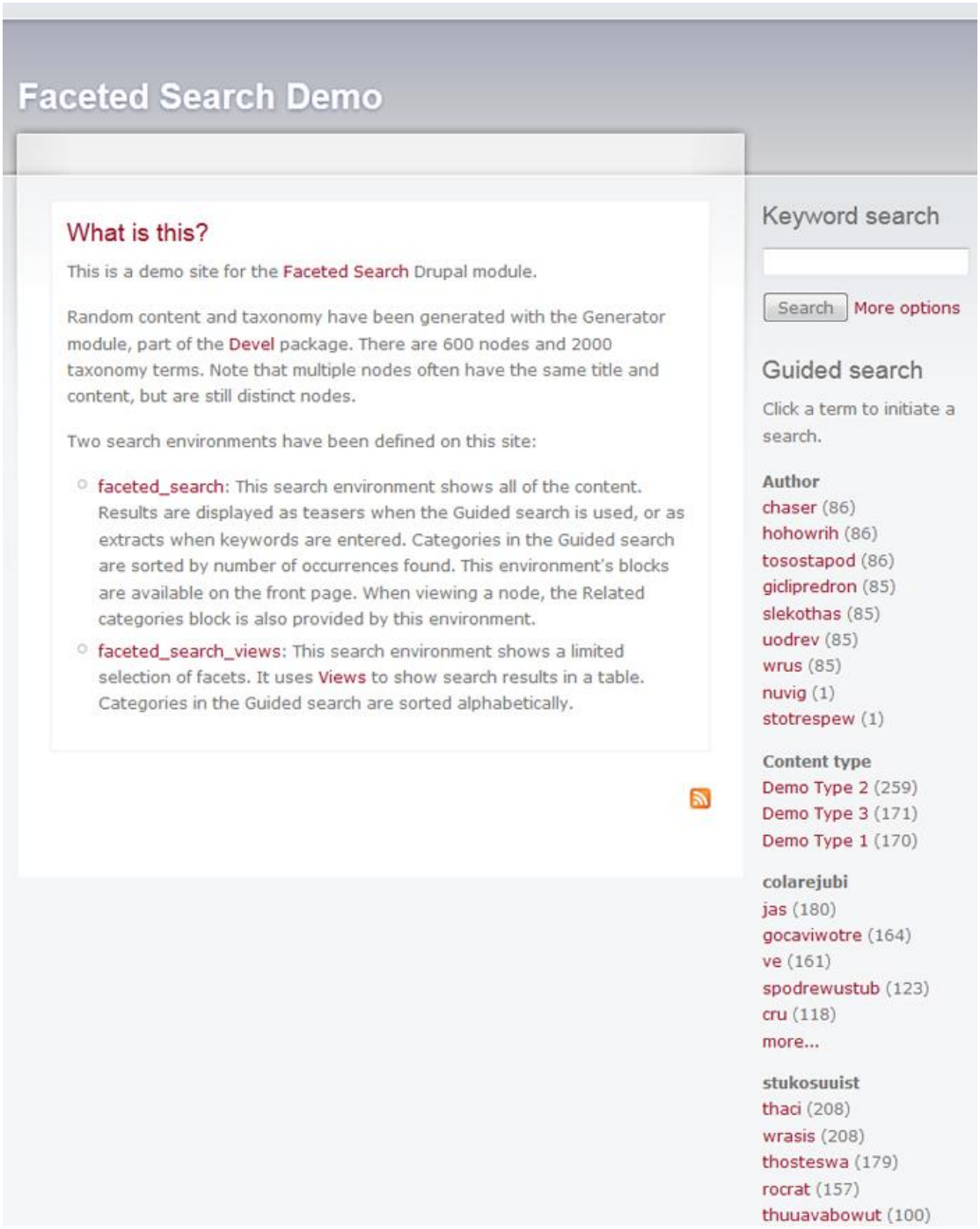
- Faceted search: Το βασικό API του faceted search
- Faceted search UI: Το search interface
- Faceted Search Views: Επιτρέπει τη χρήση όψεων για την προβολή των αποτελεσμάτων
- Author Facet: Επιτρέπει τους χρήστες να κάνουν αναζητήσεις με βάση τον συγγραφέα ενός άρθρου.
- Content type Facet: Επιτρέπει τους χρήστες να κάνουν αναζήτηση με βάση τον τύπο του εγγράφου.
- Date Authored Facet: Επιτρέπει στους χρήστες να φιλτράρουν τα δεδομένα με βάση την ημερομηνία δημιουργίας του εγγράφου.
- Taxonomy Facets: Επιτρέπει τους χρήστες τη χρήση της ταξινόμησης για τη βοήθεια των αναζητήσεων τους.
- Field Keyword Filter: Επιτρέπει τους χρήστες να αναζητήσουν περιεχόμενο με τη χρήση keyword με περιορισμό ανά πεδίο

Τη στιγμή που γράφονται αυτές οι γραμμές έχουν αναπτυχθεί και άλλα default facets. Μην ξεχνάμε ότι είναι στο χέρι μας να δημιουργήσουμε και μόνοι μας όσα πραγματικά χρειαζόμαστε.

4.2.3. Demo

Πριν δούμε την εφαρμογή που αναπτύχθηκε ας ρίξουμε μια ματιά στην default εφαρμογή του Drupal. Μπορείτε και μόνοι σας να ρίξετε μια ματιά στην διεύθυνση

http://facetedsearch.davidlesieur.com/



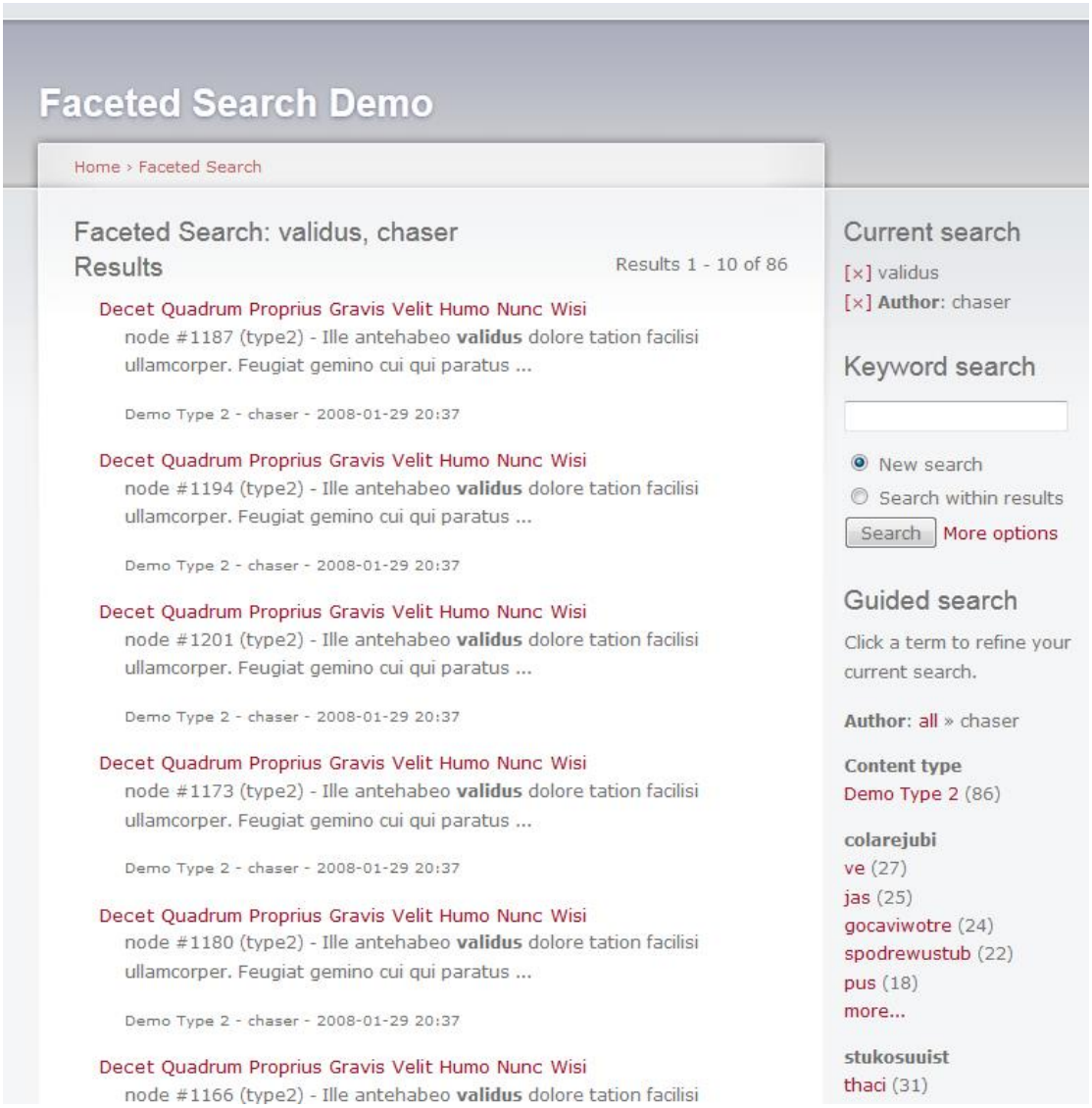
Εικόνα 38 - Faceted Search Demo

Στην πραγματικότητα η ιστοσελίδα έχει περισσότερες τιμές αλλά δεν χωρούσαν όλες οπότε αρκούμαστε να σας παρουσιάσουμε όσες βλέπετε στην παραπάνω εικόνα.

Βλέπουμε ότι το layout ακολουθεί την τακτική, δεξιά πάνω το search box από κάτω τα facets και αριστερά το περιεχόμενο/αποτελέσματα. Για περισσότερες πληροφορίες ανατρέξτε στην παράγραφο «Faceted search και κατασκευή ιστοσελίδων» (σελ. 47).

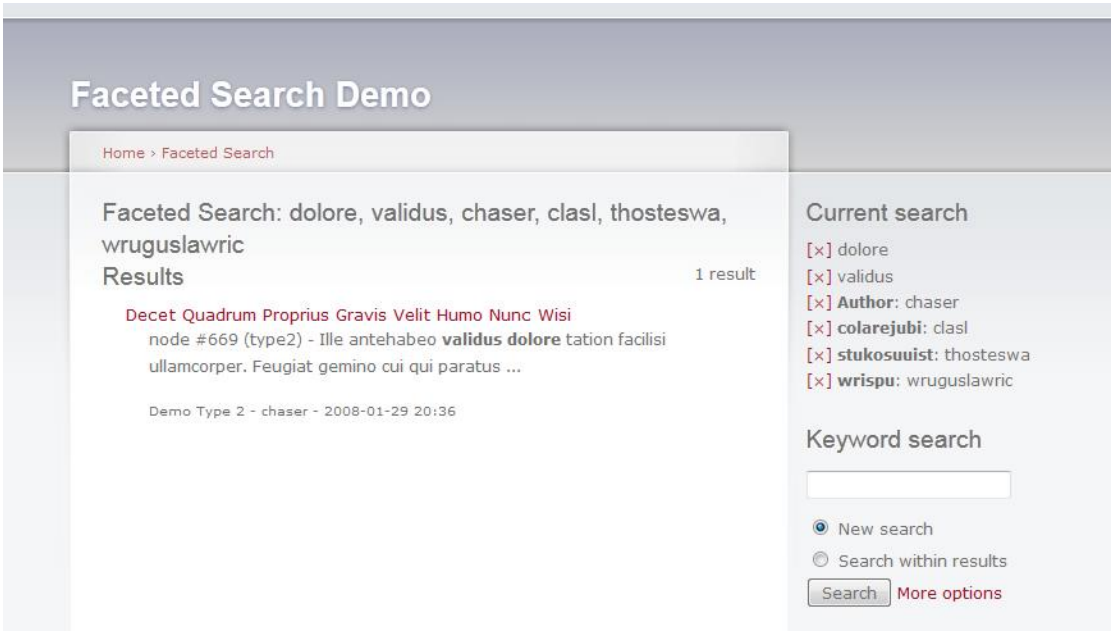
Με ένα απλό σενάριο χρήσης μπορούμε να έχουμε μια πάρα πολύ καλή εικόνα των δυνατοτήτων του component. Παρατηρούμε ότι τα facets είναι ταξινομημένα με βάση των πλήθος των τιμών τους.

Ας κάνουμε μια αναζήτηση κλικάροντας τον συγγραφέα “Chaser” ενώ παράλληλα αναζητήσουμε στα αποτελέσματα το “validus”. Θα έχουμε το εξής αποτέλεσμα:



Εικόνα 39 - Faceted Search Demo II

Στην πράξη μπορείτε να δείτε ότι δημιουργήσαμε ένα προσωρινό facet το οποίο μπορούμε να διαγράψουμε από τα κριτήρια μας κατά βούληση ή να προσθέσουμε κάποιο καινούριο. Φυσικά τα κριτήρια μας μπορούν να γίνουν ακόμα πιο συγκριμένα για μεγαλύτερη ακρίβεια στα αποτελέσματα όπως εδώ:



Εικόνα 40 - Faceted Search Demo III

Όπως παρατηρούμε με τη βοήθεια στην καθοδήγηση που μας προσφέρει το faceted search καταλήξαμε στο επιθυμητό αποτέλεσμα χωρίς να παιδευτούμε πολύ

.

5. Πρακτική εφαρμογή – Project

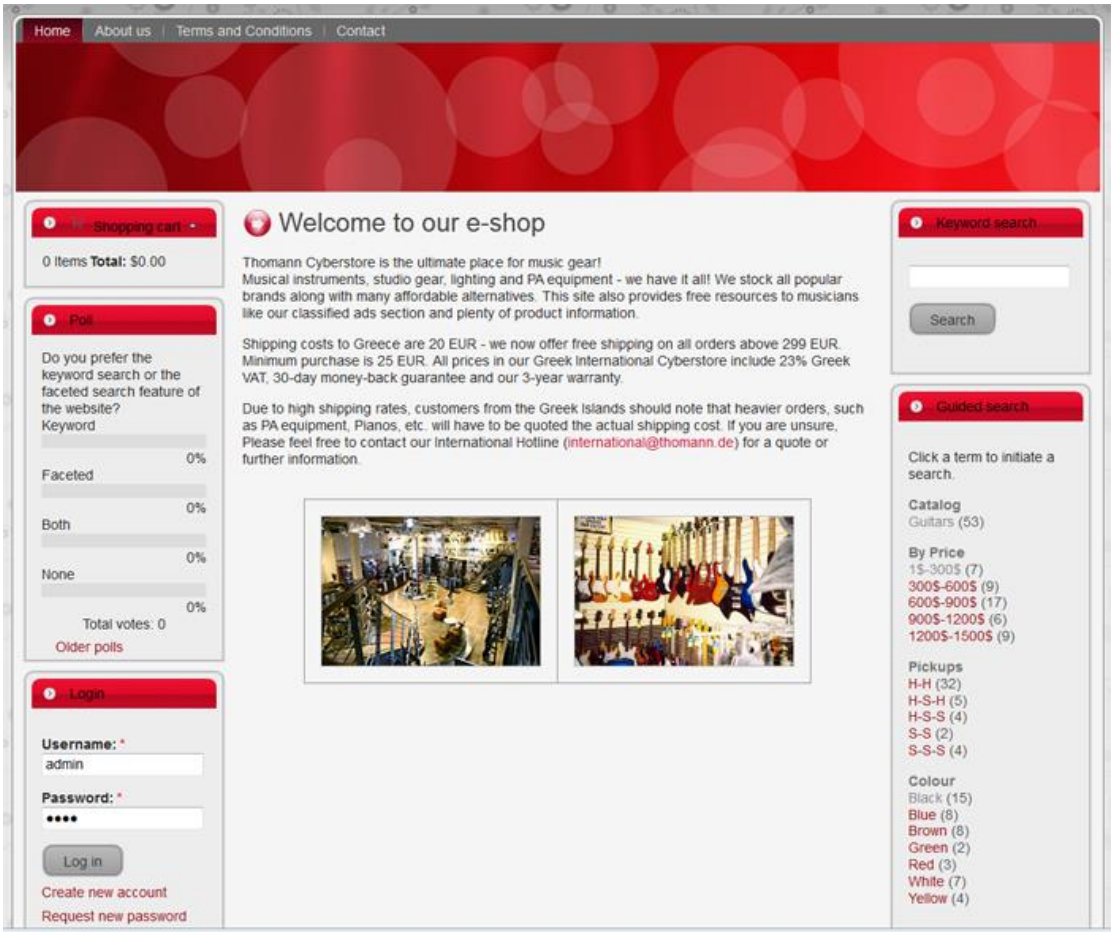
5.1. Εισαγωγή

Σε αυτό το σημείο θα κάνουμε μια περιγραφή της ιστοσελίδας που κατασκευάστηκε με σκοπό να συνοδέψει την πτυχιακή. Το url που μπορείτε να ρίξετε μια ματιά είναι το http://www.apositive.com/test/ptyxiaki . Για τη δημιουργία της, χρησιμοποιήσαμε το CMS Drupal μαζί με το faceted search component του.

5.2. Γιατί Drupal και γιατί e-shop;

Ο λόγος που επιλέξαμε το Drupal αναλύεται πλήρως στο προηγούμενο κεφάλαιο. Οι λόγοι που επιλέξαμε την κατασκευή ενός e-shop είναι ότι εν έτη 2011 τείνει να αντικαταστήσει σιγά σιγά τα παραδοσιακά καταστήματα, χρησιμοποιείται από πολλούς χρήστες και τρίτον και σημαντικότερον το faceted search διανθίζει πλέον την πλειοψηφία των e-shops στο διαδίκτυο.

5.2.1. Επιλογή front end design



Εικόνα 41 - Front end layout του E-shop μας

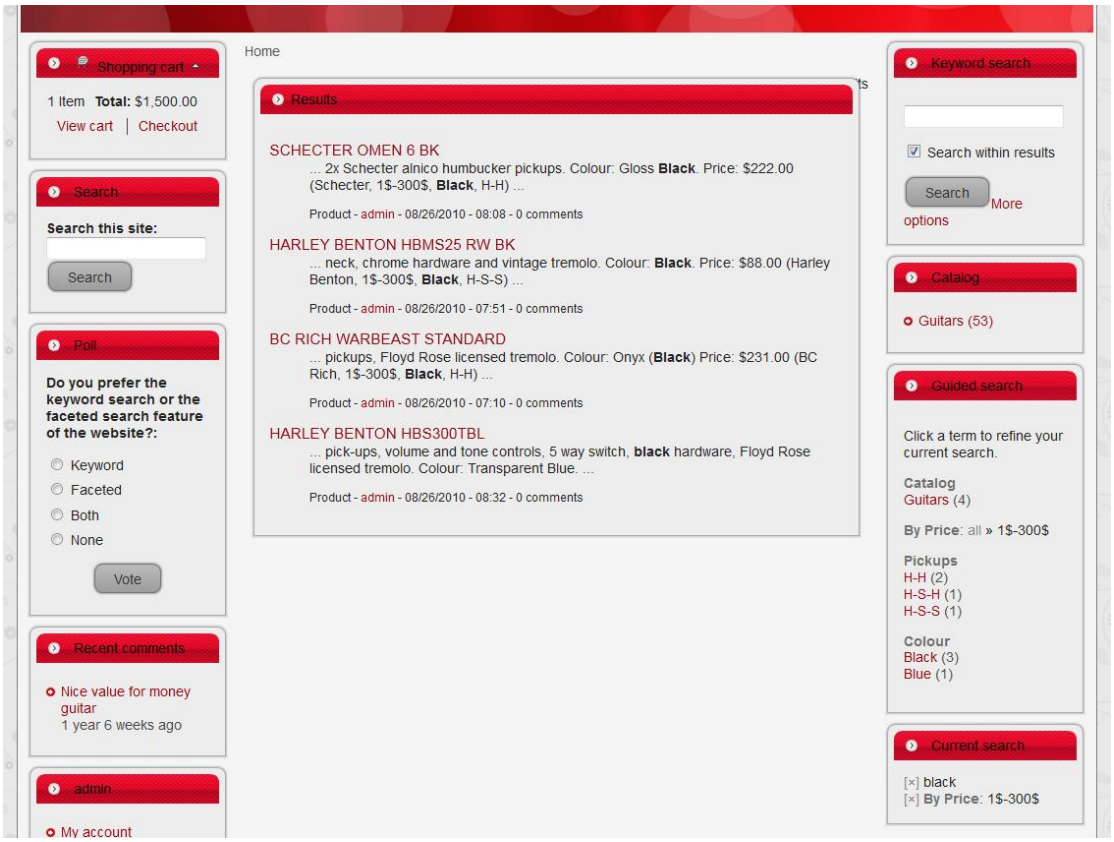
Όπως παρατηρούμε στην εικόνα το layout που διαλέξαμε φέρνει το Search Box πάνω δεξιά και από κάτω οι πίνακες με τα διάφορα facets. Κέντρο επιλέξαμε να εμφανίζουμε τα αποτελέσματα δίνοντας έτσι όσο το δυνατόν περισσότερο χώρο σε

αυτά. Αριστερά έχουμε άλλα blocks που χρειαζόμαστε για την λειτουργία του e-shop όπως το shopping cart.

Στην παρακάτω εικόνα ετοιμάσαμε ένα παράδειγμα όπου έχουμε αναζητήσει, με τη χρήση του search box, την λέξη "black" θέλοντας να φιλτράρουμε ενδεχομένως μαύρες κιθάρες. Ταυτόχρονα επιλέξαμε το facet value που ορίζει το εύρος των τιμών από 1-300. Τα αποτελέσματα εμφανίζονται στο κέντρο. Παράλληλα έχουμε ορίσει το search box να κάνει αναζήτηση στα υπάρχοντα αποτελέσματα. Αυτό βέβαια αλλάζει «ξετικάροντας» το αντίστοιχο check-box που εμφανίζεται από κάτω.

Παρατηρείστε ότι τα facets έχουν ταξινομηθεί αλφαβητικά αντί με βάση το πλήθος που και αυτό παρουσιάζεται στην παρένθεση.

Η λογική που έχουμε ακολουθήσει δεν ξεφεύγει πολύ από ότι έχουμε περιγράψει και ότι ισχύει γενικότερα. Ο λόγος φυσικά είναι ότι δε θέλουμε να μπερδέψουμε και να συγχύσουμε τον χρήστη περισσότερο από ότι καταφέρνει το information overload, το οποίο είναι λίγο αναπόφευκτο αν θέλουμε να έχουμε μεγάλη ποικιλία στο e-shop.



Εικόνα 42 - Παρουσίαση αποτελεσμάτων E-shop

5.2.2. Back end

Και εδώ οι επιλογές μας δεν ξέφυγαν από τα συνηθισμένα. Κάναμε προσεκτική επιλογή των facets και ομαδοποιήσαμε ότι ήταν εφικτό. Με τα metatags ήμασταν φειδωλοί και για κάθε προϊόν δεν επιλέξαμε παραπάνω από τρία. Στο url

http://www.apositive.com/test/ptyxiaki/admin/settings/faceted_search/1 μπορείτε να δείτε όλες τις ρυθμίσεις που έχουν γίνει.

5.2.3. Σύνοψη και συμπεράσματα

Κατά την υλοποίηση της εφαρμογής είχαμε την ευκαιρία να αντιμετωπίσουμε όλα τα προβλήματα και τις προκλήσεις που αναφέραμε στη πτυχιακή. Ήρθαμε αντιμέτωποι με το πρόβλημα του information overload καθώς και το κλασικό δίλλημα μεταξύ ακρίβειας και ανάκλησης. Το μεγαλύτερο και ακριβέστερο feedback βέβαια το παίρνουμε από τα σχόλια πολλών διαφορετικών χρηστών. Η σχεδίαση έλαβε υπόψη γενικούς κανόνες και δοκιμασμένες συνταγές αλλά παρόλα αυτά υπήρξαν και τα φαινόμενα δυσαρέσκειας από μερικούς χρήστες. Το πόρισμα που βγαίνει είναι ότι δε θα τους ικανοποιήσουμε ποτέ όλους τους χρήστες αλλά πάντα θα πηγαίνουμε με γνώμονα την ευχρηστία των περισσότερων. Ειδικά όταν πρόκειται για ένα ηλεκτρονικό κατάστημα μας ενδιαφέρει ο χρήστης να ξαναέρθει στο «μαγαζί». Ίσως είναι λίγο οξύμωρο αλλά επειδή πρόκειται για ψώνια πρέπει αυτά να είναι διασκεδαστικά όπως ακριβώς και η βόλτα στην αγορά. Αυτή ήταν και η μεγαλύτερη πρόκληση που είχαμε να αντιμετωπίσουμε να φτιάξουμε ένα interface το οποίο θα ελκύει τον χρήστη να το χρησιμοποιήσει ξανά και ξανά.

6. Επίλογος

Η αποδοτική αναζήτηση είναι το κλειδί για μια επιτυχημένη διαδικτυακή εφαρμογή, ειδικά σε μια κοινωνία πληροφορίας που διογκώνεται συνεχώς. Το faceted search ήρθε για να δώσει λύσεις σε αυτό το πρόβλημα του information overload και τα καταφέρνει περίφημα. Απόδειξη η ολοένα και μεγαλύτερη απήχηση που έχει στο διαδίκτυο. Ξαναθυμίζουμε ότι μέχρι και η Google κινείται προς αυτή την κατεύθυνση, το οποίο από μόνο του, μας δείχνει πόσο αποδοτικός τρόπος αναζήτησης είναι το faceted search. Όσοι ασχολούνται με το χώρο της αναζήτησης στο διαδίκτυο γνωρίζουνε βέβαια πάρα πολύ καλά ότι δε χρειαζόταν η Google να δείξει ενδιαφέρον, ώστε να πειστούν για την χρησιμότητα και την αποδοτικότητα ενός τέτοιου εργαλείου. Όταν από αρχαιοτάτων χρόνων υπάρχουν αναφορές για τέτοιου είδους τεχνικές καταλαβαίνουμε ότι μάλλον κάποιος λόγος θα υπάρχει. Και σκεφτείτε ότι τότε δεν υπήρχε όγκος πληροφορίας ή τουλάχιστον και να υπήρχε δεν ήταν συγκεντρωμένος σε κανένα «διαδίκτυο».

Οι πολυάριθμες προσπάθειες που έγιναν αργότερα προς αυτήν την κατεύθυνση μας μαρτυράνε την σημαντικότητα του εγχειρήματος. Σήμερα, έχουμε επιλογές από εμπορικές εφαρμογές αλλά και από open source πλατφόρμες που έχουν εξίσου αποτελεσματικές υλοποιήσεις.

Μια ματιά στο διαδίκτυο και ειδικότερα στα ηλεκτρονικά καταστήματα θα σας πείσει ακόμα περισσότερο για το γεγονός ότι το faceted search κερδίζει ολοένα και περισσότερο έδαφος. Αυτό σε συνδυασμό με το γεγονός ότι κατευθυνόμαστε προς την Semantic Web εποχή προμηνύει ότι η αναζήτηση στο διαδίκτυο δεν θα είναι όπως την ξέραμε. Και όλα αυτά γιατί και ο όγκος και ο τύπος της πληροφορίας αλλάζει και διαφοροποιείται.

Το faceted search λοιπόν, πέραν των άλλων πλεονεκτημάτων, έρχεται να δώσει τέλος στις αδιέξοδες αναζητήσεις όπως και σε αναζητήσεις που τα αποτελέσματα ήταν αμφιλεγόμενα όσο σωστά keywords και να δίναμε. Με όπλο την καθοδηγούμενη αναζήτηση και ένα πετυχημένο συνδυασμό ακρίβειας και ανάκλησης πληροφορίας πετυχαίνει να λύσει το μείζον θέμα του information overload.

Τελειώνοντας, εύκολα τίθεται το ερώτημα: Άραγε θα αντιμετωπίσει ποτέ το faceted search πρόβλημα με την ολοένα και αυξανόμενη ποσότητα πληροφορίας, τηρουμένων πάντα των αναλογιών; Δεν υπάρχει εύκολο ναι η όχι εδώ. Η υπερβολή πάντα θα είναι πρόβλημα από όποια σκοπιά και αν το δούμε. Η δυσκολία θα παρουσιαστεί όταν θα έχουμε καινούριους τύπους πληροφορίας το οποίο σημαίνει περισσότερα facets. Περισσότερα facets, όπως είδαμε και σε προηγούμενο κεφάλαιο, οδηγούν σε σύγχυση.

Η δομή του faceted search είναι τέτοια όμως που επιτρέπει στον προγραμματιστή να αντιμετωπίσει τέτοια εμπόδια. Για να είμαστε ρεαλιστές όμως ξαναθυμίζουμε ότι όσο περισσότερη πληροφορία έχουμε να διαχειριστούμε τόσο πρέπει να ανανεώνουμε και τις μεθόδους προσπέλασης της.

Για την ώρα, το faceted search κάνει αυτό που υπόσχεται. Επιτυγχάνει αποδοτικές αναζητήσεις, εξαλείφει τα μηδενικά αποτελέσματα και προσφέρει στο χρήστη καθοδήγηση ως προς το τι μπορεί να ψάξει ή και τι θέλει να ψάξει.

Βιβλιογραφία

1. Faceted Search - Daniel Tunkelang - 2009
2. Dynamic Taxonomies and Faceted - Search Theory, Practice, and Experience - Giovanni Maria Sacco, Yannis Tzitzikas – 2009
3. Designing Search: UX Strategies for ECommerce Success – 2011
4. Web application design patterns - Pawan Vora – 2009
5. Beginning Drupal - Jacob Redding – 2010
6. Beginning Drupal 7 – Todd Tomlinson

Αναφορές στο διαδίκτυο

1. http://en.wikipedia.org/wiki/Faceted_search
2. http://drupal.org/project/faceted_search
3. http://www.lucidimagination.com/devzone/technical-articles/faceted-search-solr
4. http://facetedsearch.davidlesieur.com/
5. http://en.wikipedia.org/wiki/Faceted_classification
6.http://www.ioncannon.net/programming/1055/faceted-search-with-sphinx/
7.http://flamenco.berkeley.edu/papers/faceted-workshop06.pdf
8.http://en.wikipedia.org/wiki/Taxonomy
9.http://en.wikipedia.org/wiki/Ontology_%28information_science%29
10. http://en.wikipedia.org/wiki/Entity_extraction
11. http://en.wikipedia.org/wiki/Information_extraction