Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Όταν η Wikipedia συναντά την Google: Απευθύνοντας ερωτήματα με νόημα στη μηχανή αναζήτησης Google**
Ονοματεπώνυμο Φοιτητή	**Αποστολάτος Ιωάννης**
Πατρώνυμο	**Σπυρίδων**
Αριθμός Μητρώου	**ΜΠΠΛ09017**
Επιβλέπων	**Ιωάννης Παπαδάκης, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **02 Μαρτίου 2012**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Χρήστος Δουληγέρης Καθηγητής	Παναγιώτης Κοτζανικολάου Λέκτορας	Ιωάννης Παπαδάκης Επίκουρος Καθηγητής

Περιεχόμενα

Όταν η Wikipedia συναντά την Google: Απευθύνοντας ερωτήματα με νόημα στη μηχανή αναζήτησης Google

Περίληψη

Οι μηχανές αναζήτησης είναι ο κύριος τρόπος εύρεσης πληροφοριών στο Διαδίκτυο. Τα αποτελέσματα όμως μιας αναζήτησης πολλές φορές διαφέρουν κατά πολύ από αυτό που είχε στο νου του να ψάξει αρχικά ο χρήστης. Η μηχανή αναζήτησης στο ίδιο ερώτημα, επιστρέφει πολλές φορές το ίδιο αποτέλεσμα χωρίς να υπολογίζει κάτω από ποιες συνθήκες αυτό υποβλήθηκε. Επίσης, μια λέξη που συμμετέχει στο ερώτημα (λέξη-κλειδί) μπορεί να έχει πολλές σημασίες. Έτσι, ο χρήστης αναγκάζεται να αλλάξει το αρχικό ερώτημά του προς τη μηχανή αναζήτησης, πληκτρολογώντας νέες λέξεις-κλειδιά, μέχρι να βρει αυτό που ψάχνει.

Στη διατριβή αυτή παρουσιάζεται μια υπηρεσία υποβοήθησης του χρήστη κατά τη διάρκεια υποβολής ερωτήματος προς τη μηχανή αναζήτησης. Η προτεινόμενη υπηρεσία λειτουργεί συμπληρωματικά προς τη μηχανή αναζήτησης, καθώς υλοποιεί μια διαδικασία βάσει της οποίας ο χρήστης αλληλεπιδρά με ένα σημασιολογικό σύστημα προτάσεων (recommender system) που τον βοηθάει να διαμορφώσει ένα ερώτημα που προσεγγίζει την πληροφοριακή του ανάγκη προτού αυτό υποβληθεί προς τη μηχανή αναζήτησης. Πιο συγκεκριμένα, ο χρήστης διαμορφώνει την πληροφοριακή του ανάγκη σε ερώτημα με τη βοήθεια μιας υπηρεσίας που βασίζεται στα διασυνδεδεμένα δεδομένα της DBPedia [14] . Κατόπιν, το ερώτημα που προκύπτει ως αποτέλεσμα αυτής της διαδικασίας υποβάλλεται στη μηχανή αναζήτησης Google.

Abstract

Search engines are the main way of finding information on the Internet. The search results, however, often vary from what the user had in mind of searching in the beginning. The search engine on the same query many times returns the same result without making into account under what circumstances it was made. Also, a word that participates in the query (keyword) can have many meanings. Thus, the user is forced to change his original query to the search engine by typing new keywords until he finds what he is looking for.

This thesis presents a service to aid the user during the submission process to the search engine. The proposed service complements the search engine, by implementing a process by which the user interacts with a system of semantic proposals (recommender system) that helps to form a query that approximates the information needs before it (the query) is submitted to the search engine. More specifically, the user configures his information need to a query with the help of a service based on DBPedia's linked data. Then the question that arises as a result of this process is submitted to Google's search engine.

Εισαγωγή

Το Διαδίκτυο είναι ένα παγκόσμιο πλέγμα διασυνδεδεμένων υπολογιστών που παρέχει στους χρήστες του υπηρεσίες και πληροφορίες. Από την αρχή της δημιουργίας του Διαδικτύου μέχρι τις μέρες μας, ο όγκος αυτός των υπηρεσιών και πληροφοριών αυξάνεται καθημερινά με ιλιγγιώδεις ρυθμούς. Αυτό έχει σαν συνέπεια να γίνεται όλο και πιο δύσκολο να βρει ο χρήστης κάτι στο Διαδίκτυο. Σε αυτό το πρόβλημα καλούνται να δώσουν λύση οι μηχανές αναζήτησης.

Μια μηχανή αναζήτησης είναι μια εφαρμογή που επιτρέπει την αναζήτηση πληροφοριακών πόρων (resources) στο Διαδίκτυο. Στην ουσία, μια μηχανή αναζήτησης αποτελείται από μια τεράστια βάση δεδομένων με πληροφορίες που συλλέγει από το Διαδίκτυο και από ένα γραφικό περιβάλλον μέσω του οποίου αλληλεπιδρά με τον χρήστη. Ο χρήστης πληκτρολογεί αυτό που θέλει να βρει στο γραφικό περιβάλλον, αυτό επικοινωνεί με την βάση δεδομένων και του επιστρέφει τα αποτελέσματα. Ωστόσο, τα περισσότερα αποτελέσματα που επιστρέφει η μηχανή αναζήτησης είναι άχρηστα για τον χρήστη. Όσο πιο καλά περιγράφει την πληροφοριακή του ανάγκη ο χρήστης τόσο πιο χρήσιμα είναι τα αποτελέσματα που επιστρέφει η μηχανή αναζήτησης. Για παράδειγμα, αν ο χρήστης πληκτρολογήσει τη λέξη ‘τζάγκουαρ’ ψάχνοντας πληροφορίες για το αυτοκίνητο, η μηχανή αναζήτησης θα του επιστρέψει εκτός από το αυτοκίνητο πληροφορίες και για το θηλαστικό. Ο χρήστης λοιπόν για να κάνει πιο συγκεκριμένο το ερώτημα του, πρέπει να πληκτρολογήσει μετά τη λέξη ‘τζάγκουαρ’ και τη λέξη ‘αυτοκίνητο’ έτσι ώστε η μηχανή αναζήτησης να του επιστρέψει αποτελέσματα που να πλησιάζουν σε αυτό που πραγματικά ψάχνει. Όσο πιο κατάλληλες λέξεις-κλειδιά προσθέτει ο χρήστης στο αρχικό του ερώτημα τόσο τα αποτελέσματα που επιστρέφει η μηχανή αναζήτησης πλησιάζουν στην πληροφοριακή ανάγκη του χρήστη. Υπάρχει όμως τρόπος να τεθούν αυτές οι λέξεις-κλειδιά υπόψη του χρήστη εκ των προτέρων;

Ένας τρόπος να επιτευχθεί κάτι τέτοιο είναι με την πρόταση συναφών λέξεων ως προς την αρχική λέξη που εισάγει ο χρήστης. Στην παρούσα μεταπτυχιακή διατριβή παρουσιάζουμε μια προσέγγιση πρότασης συναφών λέξεων που προέρχονται από τη Wikipedia. Πιο συγκεκριμένα, αφού ο χρήστης αρχίσει να πληκτρολογεί το αρχικό του ερώτημα, προτείνονται συναφείς λέξεις οι οποίες όταν επιλεχθούν, προστίθενται αυτόματα στη μηχανή αναζήτησης κάνοντας το αρχικό ερώτημα πιο σχετικό με την πληροφοριακή ανάγκη του χρήστη. Η μηχανή αναζήτησης που επιλέξαμε να χρησιμοποιήσουμε προκειμένου να υλοποιήσουμε την αντίστοιχη υπηρεσία που προτείνουμε είναι η Google, μια από τις μεγαλύτερες εταιρείες διαδικτυακών υπηρεσιών, που είναι πρώτη στις προτιμήσεις των χρηστών του Διαδικτύου. Για να βρούμε τις συναφείς λέξεις χρησιμοποιήσαμε τα δεδομένα της Wikipedia τα οποία αντλήσαμε από την DBpedia [3] .

Η διατριβή αυτή περιλαμβάνει τις εξής ενότητες: α) Μηχανές αναζήτησης, τι είναι και πώς λειτουργούν, β) Συναφείς λέξεις και διασυνδεδεμένα δεδομένα, όπου γίνεται επεξήγηση των συναφών λέξεων και πώς χρησιμοποιήθηκαν στην εργασία μας, γ) Wikipedia και DBpedia όπου αναφέρουμε τη σύνδεση μεταξύ τους, και πώς μας βοήθησαν στο να αντλήσουμε τα διασυνδεδεμένα δεδομένα και δ) Υλοποίηση της αντίστοιχης υπηρεσίας, που περιλαμβάνει τα εργαλεία που χρησιμοποιήσαμε και το πώς ουσιαστικά δουλεύει αυτή.

Μηχανές Αναζήτησης

Ας δούμε όμως πιο αναλυτικά τι είναι μια μηχανή αναζήτησης και πώς λειτουργεί. Οι μηχανές αναζήτησης είναι προγράμματα που επιτρέπουν στον χρήστη να ψάξει, βάσει των λέξεων-κλειδιών που πληκτρολογεί, τεράστιες βάσεις δεδομένων [2] . Οι βάσεις αυτές περιέχουν αντίγραφα τμημάτων των ιστοσελίδων που συλλέγονται και κατόπιν ευρετηριάζονται από ειδικά προγράμματα. Θα μπορούσαμε να πούμε ότι μια μηχανή αναζήτησης αποτελείται από τα εξής μέρη:

- Το αυτόματο πρόγραμμα συλλογής πληροφοριών το οποίο το συναντάμε με διάφορες ονομασίες όπως Spider ή Robot ή Crawler. Το πρόγραμμα αυτό είναι υπεύθυνο για τη συλλογή πληροφοριών από τις ιστοσελίδες και τους συνδέσμους που βρίσκονται μέσα σε αυτές. Ανά τακτά χρονικά διαστήματα επισκέπτεται τις ίδιες ιστοσελίδες για να καταγράψει τις αλλαγές μέσα σε αυτές.

- Το ευρετήριο (index) που είναι η βάση δεδομένων της μηχανής αναζήτησης και περιέχει όλες τις πληροφορίες των ιστοσελίδων που κατέγραψε ο Spider. Εδώ πρέπει να σημειωθεί ότι κάποιες μηχανές αναζήτησης ευρετηριάζουν το πλήρες κείμενο μιας ιστοσελίδας (μέχρι ένα όριο μεγέθους, π.χ. 100k για την Google) ενώ κάποιες άλλες μόνο τον τίτλο και ορισμένες λέξεις-κλειδιά που βρίσκονται σε συγκεκριμένες ετικέτες της HTML.
- Το μηχανισμό αναζήτησης που είναι το πρόγραμμα που ψάχνει στο ευρετήριο για να βρει ιστοσελίδες βάση των λέξεων-κλειδιών που έθεσε ο χρήστης.
- Το γραφικό περιβάλλον (GUI – Graphical User Interface) που είναι αυτό με το οποίο αλληλεπιδρά ο τελικός χρήστης. Ουσιαστικά, πρόκειται για μια ιστοσελίδα στο Διαδίκτυο που περιέχει την φόρμα αναζήτησης. Εκεί ο χρήστης πληκτρολογεί τις λέξεις-κλειδιά που αντιστοιχούν στην πληροφοριακή του ανάγκη και σαν επιστροφή παίρνει τους τίτλους των ιστοσελίδων με μία σύντομη περιγραφή καθώς και τις υπερσυνδέσεις που οδηγούν σε αυτές.

Αυτό που περιγράψαμε παραπάνω είναι ο βασικός κορμός μιας μηχανής αναζήτησης. Στις μέρες μας όμως οι μηχανές αναζήτησης έχουν εξελιχθεί και έχουν κάνει ένα βήμα παραπάνω στον τρόπο αναζήτησης.

Τωρινή κατάσταση στις μηχανές αναζήτησης

Η τωρινή κατάσταση των μηχανών αναζήτησης έχει αλλάξει πολύ σε σχέση με το πώς ξεκίνησαν. Οι μεγαλύτερες και πιο δημοφιλείς μηχανές (Google, Yahoo, Bing κ.α.) έχουν τροποποιήσει το γραφικό τους περιβάλλον (GUI), παρέχοντας στον χρήστη μια πληθώρα από επιλογές ώστε να τον βοηθήσουν στην αναζήτησή του.

Η σημαντικότερη νέα προσθήκη των μηχανών αναζήτησης είναι η αυτόματη συμπλήρωση (autocomplete). Ο χρήστης ξεκινάει να πληκτρολογεί στο πεδίο αναζήτησης της μηχανής και αυτή από κάτω του παρουσιάζει μια λίστα με προτεινόμενες λέξεις-κλειδιά. Η προτεινόμενη αυτή λίστα είναι προϊόν διαφόρων παραγόντων, όπως δημοφιλείς αναζητήσεις, γλώσσα αναζήτησης, ή τόπος από τον οποίο προέρχεται η αναζήτηση κ.α. [8] . Έτσι ο χρήστης, έχει την επιλογή, από τα πρώτα κιόλας γράμματα που πληκτρολογεί να επιλέξει από τη λίστα χωρίς να χρειαστεί να πληκτρολογήσει ολόκληρη τη λέξη-φράση που αναζητά. Επιλέγοντας ο χρήστης από τη λίστα αμέσως η μηχανή θα του εμφανίσει τα αντίστοιχα αποτελέσματα.

Εκτός από την αυτόματη συμπλήρωση που έχει προστεθεί στις μηχανές αναζήτησης τα τελευταία χρόνια, οι μηχανές έχουν προσθέσει στο γραφικό τους περιβάλλον και άλλες επιλογές σύνθετης αναζήτησης. Έτσι δίνουν τη δυνατότητα στον χρήστη να επιλέξει αν η μηχανή θα αναζητήσει τις λέξεις του σε αρχεία, εικόνες, βίντεο κ.α. Όπως επίσης σε ποια γλώσσα ή από ποια χώρα θα προέρχονται τα αποτελέσματα της αναζήτησης καθώς και το χρόνο που ανέβηκαν τα αρχεία στο Διαδίκτυο. Αυτές είναι μόνο μερικές από τις επιλογές των σύνθετων αναζητήσεων που παρέχουν οι μηχανές αναζήτησης στον χρήστη και που βοηθούν τον χρήστη να βρει αυτό που πραγματικά ψάχνει. Με εξαίρεση την αυτόματη συμπλήρωση, οι υπόλοιπες προσθήκες στο γραφικό περιβάλλον των μηχανών αναζήτησης παρέχονται αφού επιστραφούν τα πρώτα αποτελέσματα αναζήτησης. Έτσι, στην περίπτωση όπου το αρχικά αποτελέσματα δεν προσεγγίζουν την πραγματική πληροφοριακή ανάγκη του χρήστη, οι προσθήκες αυτές καθίστανται ουσιαστικά άχρηστες.

Έτσι οι χρήστες εξακολουθούν να έχουν πρόβλημα στο να προσεγγίσουν την πληροφορία που αναζητούν εξαιτίας του σημασιολογικού χάσματος που υπάρχει ανάμεσα στην πληροφοριακή ανάγκη που είχαν αρχικά στο μυαλό τους και στο ερώτημα που τελικά διατύπωσαν στη μηχανή αναζήτησης. Τα πράγματα γίνονται ακόμα χειρότερα αν λάβουμε υπόψη μας το γεγονός ότι οι χρήστες των μηχανών αναζήτησης μπορεί να βρίσκονται σε διαφορετικές πληροφοριακές καταστάσεις σε κάθε τους αναζήτηση.

Οι πληροφοριακές καταστάσεις των χρηστών των μηχανών αναζήτησης μεγάλης κλίμακας στον Παγκόσμιο Ιστό

Ο κάθε χρήστης του Διαδικτύου όχι μόνο είναι ξεχωριστός αλλά ακολουθεί διαφορετική συμπεριφορά κάθε φορά που διατυπώνει ερωτήματα προς τις μηχανές αναζήτησης. Αυτό οφείλεται κυρίως στο διαφορετικό γνωστικό υπόβαθρο του κάθε χρήστη, στις ανάγκες του την χρονική στιγμή που αναζητά την πληροφορία, καθώς και στην ίδια την πληροφορία που αναζητά. Ο τρόπος προσέγγισης της πληροφορίας από τους χρήστες είναι σημαντικός για την εργασία μας, για να μπορέσουμε να κατανοήσουμε τη λογική του χρήστη. Υπάρχουν αρκετές μελέτες που προσπαθούν να αποσαφηνίσουν τους διαφορετικούς τρόπους αναζήτησης που χρησιμοποιούν οι χρήστες (π.χ.[10] , [11]). Για τους σκοπούς της εργασίας μας, βασιστήκαμε στην προσέγγιση της Spencer [33] , η οποία παρουσιάζει τέσσερα μοντέλα αναζήτησης της πληροφορίας από τους χρήστες.

Το μοντέλο αναζήτησης "**γνωστό αντικείμενο**" υφίσταται όταν ο χρήστης ξέρει τι ψάχνει και γνωρίζει ποιες λέξεις πρέπει να χρησιμοποιήσει για να το βρει. Σχεδόν πάντα ο χρήστης είναι ικανοποιημένος με την πρώτη απάντηση που θα του επιστρέψει η μηχανή αναζήτησης.

Το μοντέλο της "**διερευνητικής**" αναζήτησης υφίσταται όταν ο χρήστης ξέρει τι ψάχνει αλλά δεν γνωρίζει ποιες λέξεις να χρησιμοποιήσει για να εκφράσει το ερώτημά του. Στη συγκεκριμένη περίπτωση η μηχανή αναζήτησης δυσκολεύεται στο να επιστρέψει τα σωστά αποτελέσματα εξαιτίας της αντίστοιχης δυσκολίας του χρήστη να περιγράψει σωστά την πληροφοριακή του ανάγκη.

Το μοντέλο αναζήτησης "**δεν γνωρίζω τι ακριβώς ψάχνω**" υφίσταται όταν ο χρήστης υποβάλλει ερωτήματα στη μηχανή αναζήτησης χωρίς να ξέρει τι πραγματικά ψάχνει. Συνήθως ο χρήστης υποβάλλει τέτοια ερωτήματα όταν θέλει να μάθει αν υπάρχει κάτι νεότερο σχετικά με μια ευρύτερη θεματική κατηγορία. Στην περίπτωση αυτή η μηχανή αναζήτησης δυσκολεύεται να κατανοήσει τι πραγματικά ψάχνει ο χρήστης ο οποίος πρέπει να βασιστεί στα αποτελέσματα του αρχικού ερωτήματος έτσι ώστε να το τροποποιήσει κατάλληλα.

Τέλος, το μοντέλο αναζήτησης "**επανεύρεση**" υφίσταται όταν ο χρήστης υποβάλλει ερωτήματα στη μηχανή αναζήτησης για να βρει κάτι που έχει ξανασυναντήσει σε παλαιότερο ερώτημα του. Αυτό μπορεί να αντιμετωπιστεί μέσω της αυθεντικοποίησης στη μηχανή αναζήτησης ώστε η τελευταία να αποθηκεύει τα ερωτήματα του χρήστη καθώς και κάποια αγαπημένα (εξατομίκευση - personalization).

Ας δούμε όμως πιο εκτενώς το πρόβλημα της διατύπωσης ερωτημάτων ακριβείας στις μηχανές αναζήτησης.

Το πρόβλημα της διατύπωσης ερωτημάτων ακριβείας στις μηχανές αναζήτησης

Στη σημερινή εποχή, οι μηχανές αναζήτησης είναι ο κύριος τρόπος εύρεσης πληροφοριών στο Διαδίκτυο. Κάθε χρήστης, μπορεί απλά πληκτρολογώντας μερικές λέξεις-κλειδιά στο πεδίο αναζήτησης της μηχανής να βρει αυτό που ψάχνει. Δεν είναι όμως λίγες οι φορές που οι χρήστες δεν ικανοποιούν τις πληροφοριακές τους ανάγκες από τα αποτελέσματα που τους επιστρέφει η μηχανή αναζήτησης και αυτό οφείλεται σε διάφορους λόγους.

Όπως αναφέρθηκε και στην προηγούμενη παράγραφο, ένας λόγος είναι ότι οι χρήστες δυσκολεύονται στο να εντοπίσουν τις λέξεις-κλειδιά που πρέπει να χρησιμοποιήσουν στο ερώτημά τους έτσι ώστε η μηχανή αναζήτησης να τους επιστρέψει τα επιθυμητά αποτελέσματα. Αυτό οφείλεται κυρίως στο ότι οι χρήστες δεν γνωρίζουν το εξειδικευμένο λεξιλόγιο που χρησιμοποιείται στους πόρους (resources) που ψάχνουν. Αυτό έχει σαν αποτέλεσμα οι χρήστες να χρησιμοποιούν λέξεις συνώνυμες με αυτές που έχει ευρετηριάσει η μηχανή αναζήτησης και η μηχανή αναζήτησης να τους επιστρέφει αποτελέσματα που δεν ικανοποιούν τις πληροφοριακές τους ανάγκες.

Άλλος λόγος είναι ότι η μηχανή αναζήτησης δεν έχει τρόπο να δώσει στον χρήστη τη δυνατότητα να ορίσει τη σημασία της λέξης που βάζει, όταν αυτή έχει πολλές σημασίες

(ambiguity). Έτσι, η μηχανή αναζήτησης ταξινομεί ψηλά τους πιο διάσημους πόρους οι οποίοι μπορεί να μην ενδιαφέρουν το χρήστη καθώς έχουν άλλη σημασία. Είναι το γνωστό πρόβλημα του 'word sense disambiguation' [12] που αντιμετωπίζουν οι υπολογιστές, όπου μια λέξη μπορεί να έχει πολλές έννοιες ανάλογα πώς θα χρησιμοποιηθεί μέσα σε μια πρόταση. Για παράδειγμα η αγγλική λέξη 'bass' (μπάσο), μπορεί να είναι είδος ψαριού ή τόνος χαμηλής συχνότητας. Ανάλογα με τον τρόπο που θα χρησιμοποιηθεί η λέξη σε μία πρόταση ο άνθρωπος εύκολα μπορεί να καταλάβει τη σημασία της. Δε συμβαίνει όμως το ίδιο και με τους υπολογιστές. Πιο συγκεκριμένα στην περίπτωσή μας, η μηχανή αναζήτησης για να κατανοήσει τι ακριβώς ψάχνει ο χρήστης και να επιστρέψει σωστά αποτελέσματα θα χρειαστεί και άλλες λέξεις-κλειδιά. Έτσι λοιπόν, ο χρήστης θα πρέπει να βάλει τη λέξη 'fish' δίπλα στη λέξη 'bass' ώστε η μηχανή να κατανοήσει ότι ο χρήστης ψάχνει πληροφορίες για το ψάρι και να μην του επιστρέψει αποτελέσματα για τη μουσική.

Σε αυτήν τη διατριβή, προτείνουμε μια υπηρεσία υποβοήθησης του χρήστη κατά τη διάρκεια υποβολής ερωτήματος προς τη μηχανή αναζήτησης, η οποία αντιμετωπίζει το πρόβλημα της συνωνυμίας και της αμφισημίας των όρων παρέχοντας στους χρήστες μια επιπλέον βοήθεια έτσι ώστε να διατυπώσουν το ερώτημα τους προς τη μηχανή αναζήτησης με ακρίβεια και αυτή με τη σειρά της να τους επιστρέψει τα αποτελέσματα που επιθυμούν. Πιο συγκεκριμένα, τους παρέχουμε συναφείς λέξεις με την πληροφοριακή τους ανάγκη, έτσι ώστε να μπορέσουν να την εκφράσουν με ακρίβεια προς τη μηχανή αναζήτησης.

Για την εύρεση των κατάλληλων λέξεων-κλειδιών χρησιμοποιήσαμε τα διασυνδεδεμένα δεδομένα της DBpedia. Για να καταλάβουμε όμως πώς λειτουργούν τα διασυνδεδεμένα δεδομένα, πρέπει πρώτα να εξηγήσουμε τι είναι ο Σημασιολογικός Ιστός [27] καθώς και το πώς αναπαριστάται η πληροφορία στο Διαδίκτυο μέσω του προτύπου RDF (Resource Description Framework) [23] .

Σημασιολογικός Ιστός: Το πρότυπο RDF και τα διασυνδεδεμένα δεδομένα

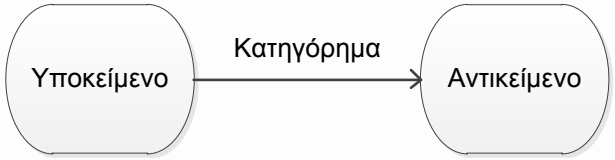
Ο Σημασιολογικός Ιστός είναι μια επέκταση του Παγκόσμιου Ιστού, όπου η πληροφορία αποκτά δομή και σημασιολογία, έτσι ώστε να γίνει πιο αποδοτική η αναζήτηση, επεξεργασία και ενοποίηση των δεδομένων. Πρωτεύων σκοπός είναι η δόμηση της πληροφορίας, έτσι ώστε αυτή να είναι προσπελάσιμη από εφαρμογές υπολογιστών. Ο Σημασιολογικός Ιστός δεν είναι η σύνδεση μεταξύ ιστοσελίδων όπως πιστεύουν πολλοί. Ο Σημασιολογικός Ιστός περιγράφει τις σχέσεις μεταξύ πραγμάτων και τα χαρακτηριστικά τους. Υπάρχουν πολλές δραστηριότητες που αποτελούν μέρος του Σημασιολογικού Ιστού. Εμείς θα ασχοληθούμε με τα διασυνδεδεμένα δεδομένα και το πρότυπο RDF.

Το πρότυπο RDF, είναι ένας τρόπος αναπαράστασης της πληροφορίας, ο οποίος έχει αναπτυχθεί από την W3C (World Wide Web Consortium) [22] . Το RDF αποτελεί τη βάση των διασυνδεδεμένων δεδομένων, καθώς η αναπαράσταση του σε μηχαναγνώσιμη μορφή επέτρεψε τη δημιουργία του σύννεφου διασυνδεδεμένων δεδομένων (lod cloud) [34] . Σε κάθε κόμβο του σύννεφου των διασυνδεδεμένων δεδομένων βρίσκεται μια αποθήκη πληροφοριών σε μορφή RDF Triplestore [35] και ενδεχομένως ένα σημείο πρόσβασης στις πληροφορίες αυτές μέσω της γλώσσας ερωτημάτων SPARQL (SPARQL Protocol and RDF Query Language) [29] που ονομάζεται 'SPARQL Endpoint'. Ο παράγοντας εκείνος που καθιστά τους κόμβους (και κατ' επέκταση τα δεδομένα) του σύννεφου διασυνδεδεμένους, είναι η δυνατότητα κοινής χρήσης URIs για τους πόρους (resources) που συνθέτουν την πληροφορία RDF σε κάθε Triplestore ξεχωριστά.

Το πρότυπο RDF και η αναπαράσταση του σε μορφότυπο N3/Turtle

Το πρότυπο RDF χρησιμοποιεί τριπλέτες (triples) της μορφής υποκείμενο-κατηγόρημα-αντικείμενο για να περιγράψει τη σχέση των δεδομένων. Το αντικείμενο και το υποκείμενο είναι οι πόροι του Ιστού και το κατηγόρημα εκφράζει τη σχέση μεταξύ τους. Κάθε τέτοια τριπλέτα ονομάζεται γράφος RDF και μπορεί να απεικονιστεί με δύο κόμβους συνδεδεμένους μεταξύ

τους με ένα τόξο (Εικόνα 1). Οι κόμβοι του γραφήματος είναι το υποκείμενο και το αντικείμενο ενώ το τόξο είναι το κατηγόρημα και δείχνει πάντα προς το αντικείμενο.



Εικόνα 1: Γράφος RDF

Κάθε κόμβος μπορεί να είναι ένα URI (Uniform Resource Identifier) ή ένα λεκτικό (literal). Όταν το URI χρησιμοποιείται ως κατηγόρημα, προσδιορίζει τη σχέση μεταξύ των κόμβων που συνδέει.

Πώς όμως η αναπαράσταση αυτή της σχέσης των δεδομένων μπορεί να αποτυπωθεί σε μια γλώσσα κειμένου φιλική προς τον άνθρωπο; Την απάντηση σε αυτό το ερώτημα έρχεται να δώσει το μορφότυπο Notation3/Turtle. Πιο συγκεκριμένα, το Notation3 [30] [29] είναι μια μηχαναγνώσιμη αναπαράσταση της πληροφορίας στον Σημασιολογικό Ιστό ενώ το Turtle [31] είναι μια έκδοση αυτής πιο φιλική προς τον άνθρωπο.

Το μορφότυπο Turtle επιτρέπει την αναπαράσταση ενός RDF γράφου σε μορφή κειμένου. Στο μορφότυπο Turtle οι τριπλέτες διαχωρίζονται μεταξύ τους με την τελεία ενώ στο τέλος κάθε μιας τριπλέτας ή σε ξεχωριστή γραμμή μπορούν να προστεθούν σχόλια μετά το σύμβολο #. Οι όροι (υποκείμενο, κατηγόρημα, αντικείμενο) της τριπλέτας διαχωρίζονται μεταξύ τους με το κενό π.χ.:

```
<http://fake.foo.gr/ioannis>  example:fav  <http://fake.foo.gr/akis> . #relation
```

Αν κάποιος από τους όρους της τριπλέτας είναι σε μορφή URI τότε αυτός περικλείεται από ‘<’ και ‘>’. Ενώ αν κάποιος από τους όρους της τριπλέτας είναι λεκτικό τότε αυτός πρέπει να βρίσκεται ανάμεσα σε διπλά εισαγωγικά. Ενώ τα λεκτικά και τα URI μπορούν να περιλαμβάνουν χαρακτήρες διαφυγής π.χ.:

```
<http://fake.foo.gr/ioannis>  <http://fake.foo.gr/relation>  “akis” .
```

Τέλος πολλές φορές για συντομογραφία κόβουμε τα αρχικό κοινό μέρος των URI με μια δήλωση της μορφής @prefix λεκτικό. Για παράδειγμα αν έχουμε τα κατηγορήματα: http://xmlns.com/foaf/0.1/name και http://xmlns.com/foaf/0.1/knows, μπορούμε να δηλώσουμε το εξής: @prefix foaf http://xmlns.com/foaf/0.1/. Έτσι τα κατηγορήματα γίνονται: foaf:name και foaf:knows, όπου foaf είναι ο κωδικός που αντιστοιχεί στον κωδικό http://xmlns.com/foaf/0.1/. Οι συντομογραφίες αυτές δεν περικλείονται από τα σύμβολα ανισότητας όπως τα URI. Έτσι για παράδειγμα, αν θέλαμε να πούμε ότι ο Ιωάννης έχει αγαπημένο φίλο τον Άκη, παίρνουμε σαν παραδοχή ότι η σχέση «έχει αγαπημένο φίλο» αντιστοιχεί στο URI: http://CelebrityMagazine.com/personal_relationships#fav. Οπότε μια πλήρης τριπλέτα N3/Turtle θα είχε τη μορφή:

```
<http://fake.foo.gr/ioannis>
<http://CelebrityMagazine.com/personal_relationships#fav><http://fake.foo.gr/akis> .
```

Μπορούμε βέβαια να κάνουμε τη δήλωση ότι:

```
@prefix example <http://CelebrityMagazine.com/personal_relationships#> .
```

Οπότε η τριπλέτα να γίνει:

```
<http://fake.foo.gr/ioannis> example:fav <http://fake.foo.gr/akis> .
```

SPARQL

Η γλώσσα SPARQL μοιάζει στη σύνταξη της με την SQL και χρησιμοποιείται για τη διατύπωση ερωτημάτων στους γράφους RDF. Τα περισσότερα ερωτήματα σε γλώσσα SPARQL, περιέχουν μια σειρά από μοτίβα τριπλέτας που ονομάζονται 'βασικό μοτίβο γραφήματος'. Τα μοτίβα αυτά είναι σαν τις τριπλέτες RDF με τη διαφορά ότι καθένα από τα υποκείμενο, κατηγόρημα, αντικείμενο μπορεί να είναι μια μεταβλητή.

Για παράδειγμα αν έχουμε την τριπλέτα

```
<http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> "SPARQL Tutorial" .
```

και θέλουμε από τη συγκεκριμένη να βρούμε τον τίτλο του βιβλίου, τότε το ερώτημα σε γλώσσα SPARQL θα ήταν:

```
SELECT ?title
WHERE {<http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> ?title .}
```

Το ερώτημα αυτό αποτελείται από δύο μέρη: το SELECT που προσδιορίζει τη μεταβλητή που θα επιστρέψει το αποτέλεσμα του ερωτήματος και το WHERE που προσδιορίζει το βασικό μοτίβο γράφου που πρέπει να βρεθεί στα δεδομένα. Το βασικό μοτίβο γράφου στο παράδειγμα μας αποτελείται από ένα μοτίβο τριπλέτας με μία μεταβλητή (?title) στην θέση του αντικειμένου. Έτσι η απάντηση στο ερώτημα μας θα είναι 'SPARQL Tutorial'.

Τέτοια ερωτήματα SPARQL μπορούν να χρησιμοποιηθούν στα διασυνδεδεμένα δεδομένα της DBPedia. Για παράδειγμα μπορούμε να δώσουμε το εξής ερώτημα:

```
SELECT ?abstract
WHERE { {
<http://dbpedia.org/resource/Civil_engineering><http://dbpedia.org/ontology/abstract>?abstract
} }
```

για να πάρουμε όλες τις περιλήψεις που αφορούν τους πολιτικούς μηχανικούς σε όλες τις γλώσσες. Αν όμως δώσουμε το ερώτημα:

```
SELECT ?abstract
WHERE { {
<http://dbpedia.org/resource/Civil_engineering><http://dbpedia.org/ontology/abstract>?abstract.
FILTER langMatches( lang(?abstract), 'en') } }
```

κάνουμε το ερώτημα πιο συγκεκριμένο αφού ζητάμε μόνο τις περιλήψεις που αφορούν τους πολιτικούς μηχανικούς και είναι γραμμένες στην αγγλική γλώσσα. Ας δούμε όμως τι είναι η DBPedia και πώς συνδέεται με την Wikipedia.

DBPedia

Η Wikipedia είναι μια ελεύθερη, διαδικτυακή, πολυγλωσσική εγκυκλοπαίδεια. Ξεκίνησε τον Ιανουάριο του 2001 και σήμερα περιέχει πάνω από 20 εκατομμύρια άρθρα, τα οποία είναι γραμμένα και ανανεώνονται καθημερινά, από εθελοντές από όλον τον κόσμο. Αυτήν τη μεγάλη εγκυκλοπαίδεια αποφασίσαμε να χρησιμοποιήσουμε έτσι ώστε να βρούμε τις συναφείς λέξεις που χρειαζόμαστε. Πιο συγκεκριμένα, χρησιμοποιήσαμε τα διασυνδεδεμένα δεδομένα της DBpedia.

Η DBPedia είναι μια κοινοτική προσπάθεια εξαγωγής της μη δομημένης πληροφορίας που παρέχει η Wikipedia, δίνοντας προγραμματιστική πρόσβαση σε αυτήν την πληροφορία με τη χρήση τεχνολογιών του σημασιολογικού Ιστού όπως είναι τα διασυνδεδεμένα δεδομένα και η SPARQL.

Η DBpedia λοιπόν έχει πάρει τα τις πληροφορίες της Wikipedia και τις παρέχει ως διασυνδεδεμένα δεδομένα στους χρήστες με τη μορφή συνόλου δεδομένων (datasets) τα οποία περιέχουν την πληροφορία σε τριπλέτες RDF [15] . Τα datasets της DBpedia που χρησιμοποιήσαμε εμείς για την ανάπτυξη της εργασίας μας είναι τα εξής: 'Wikipedia Pagelinks', 'Articles Categories', 'Disambiguation Links', 'Wordnet Classes' και 'Ontology Infobox Properties'. Ας δούμε όμως συνοπτικά τι περιέχει το κάθε σύνολο δεδομένων καθώς και σχετικά παραδείγματα.

Προτεινόμενη προσέγγιση

Αποφασίσαμε λοιπόν να χρησιμοποιήσουμε τα παραπάνω διασυνδεδεμένα δεδομένα της DBpedia, για να βοηθήσουμε τον χρήστη να βρει τις κατάλληλες λέξεις-κλειδιά που χρειάζεται για να διατυπώσει το ερώτημά του. Έτσι κατασκευάσαμε μια εφαρμογή η οποία από τα πρώτα γράμματα που πληκτρολογεί ο χρήστης, του παρέχει λέξεις για να τον βοηθήσει να ξεκινήσει την αναζήτησή του. Στη συνέχεια, όταν ο χρήστης επιλέξει την αρχική του λέξη, η εφαρμογή του επιστρέφει συναφείς λέξεις τις οποίες μπορεί να επιλέξει έτσι ώστε να προστεθούν στο αρχικό ερώτημα. Αυτό αποδείχτηκε ιδιαίτερα βολικό γιατί ο χρήστης όχι μόνο βρίσκει τις κατάλληλες λέξεις-κλειδιά αλλά δεν χρειάζεται καν να τις πληκτρολογήσει.

Ας δούμε λοιπόν ποια από τα διασυνδεδεμένα δεδομένα της DBpedia χρησιμοποιήσαμε στην υπηρεσία που κατασκευάσαμε.

Τα διασυνδεδεμένα δεδομένα της DBPedia που χρησιμοποιεί η εφαρμογή

Το αρχείο '**Wikipedia Pagelinks**' περιέχει τις εσωτερικές συνδέσεις ανάμεσα στα άρθρα της Wikipedia. Το κατηγόρημα στο συγκεκριμένο αρχείο είναι το ίδιο σε όλες τις εγγραφές. Για παράδειγμα μια ενδεικτική τριπλέτα του αρχείου είναι η:

```
<http://dbpedia.org/resource/AfghanistanGeography>
<http://dbpedia.org/ontology/wikiPageWikiLink>
<http://dbpedia.org/resource/Geography_of_Afghanistan> .
```

Σύμφωνα με την παραπάνω τριπλέτα ο πληροφοριακός πόρος 'AfghanistanGeography' με URI http://dbpedia.org/resource/AfghanistanGeography συνδέεται μέσω της σχέσης wikiPageWikiLink (Σύνδεση μιας σελίδας Wikipedia με μια άλλη σελίδα Wikipedia) με τον πληροφοριακό πόρο 'Geography_of_Afghanistan' με URI

http://dbpedia.org/resource/Geography_of_Afghanistan.

Το αρχείο '**Articles Categories**' μας δίνει πληροφορίες για το σε ποιες κατηγορίες ανήκει κάθε άρθρο της Wikipedia. Είναι και αυτό σε μορφή τριπλέτας URI με την ίδια δομή με το προηγούμενο. Για παράδειγμα μια ενδεικτική τριπλέτα του αρχείου είναι η:

```
<http://dbpedia.org/resource/Alabama>
<http://purl.org/dc/terms/subject>
<http://dbpedia.org/resource/Category:Former_British_colonies> .
```

Σύμφωνα με την παραπάνω τριπλέτα ο πληροφοριακός πόρος ‘Alabama’ με URI http://dbpedia.org/resource/Alabama συνδέεται μέσω της σχέσης subject (το θέμα του πληροφοριακού πόρου) με τον πληροφοριακό πόρο ‘Category:Former_British_colonies’ με URI http://dbpedia.org/resource/Category:Former_British_colonies.

Το επόμενο αρχείο ‘**Disambiguation Links**’ μας δίνει την πληροφορία για κάθε άρθρο σε ποια πρότυπα αποσαφήνισης ανήκει. Για παράδειγμα ένα άρθρο για τον ‘Ερμή’ μπορεί να αναφέρεται σε κάποιο στοιχείο, στον πλανήτη, στον Έλληνα Θεό και σε άλλα. Μία ενδεικτική τριπλέτα του αρχείου είναι η:

```
<http://dbpedia.org/resource/Alien>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Alien_%28law%29> .
```

Σύμφωνα με την παραπάνω τριπλέτα ο πληροφοριακός πόρος ‘Alien’ με URI http://dbpedia.org/resource/Alien συνδέεται μέσω της σχέσης wikiPageDisambiguates (Αποσαφήνιση του πληροφοριακού πόρου της Wikipedia) με τον πληροφοριακό πόρο ‘Alien_(law)’ με URI http://dbpedia.org/resource/Alien_%28law%29.

Το επόμενο αρχείο ‘**Wordnet Classes**’ μας πληροφορεί σε ποια τάξη του Wordnet [25] ανήκει κάθε άρθρο. Δεν θα επεκταθούμε σε αυτό απλά αναφέρουμε ότι το Wordnet είναι μια βάση αγγλικού λεξικού. Η τριπλέτα του αρχείου αυτού είναι βασισμένη στα πρότυπα του Wordnet. Μία ενδεικτική τριπλέτα του αρχείου είναι η:

```
<http://dbpedia.org/resource/%21Hero>
<http://dbpedia.org/property/wordnet_type>
<http://www.w3.org/2006/03/wn/wn20/instances/synset-musical-noun-1> .
```

Σύμφωνα με την παραπάνω τριπλέτα ο πληροφοριακός πόρος ‘!Hero’ με URI http://dbpedia.org/resource/%21Hero συνδέεται μέσω της σχέσης wordnet_type (Τύπος wordnet) με τον πληροφοριακό πόρο ‘synset-musical-noun-1’ με URI http://www.w3.org/2006/03/wn/wn20/instances/synset-musical-noun-1.

Τέλος έχουμε το αρχείο ‘**Ontology Infobox Properties**’ που περιέχει τις πληροφορίες που βρίσκουμε στα άρθρα της Wikipedia μέσα στα κουτιά πληροφοριών. Μία ενδεικτική τριπλέτα του αρχείου είναι η:

```
<http://dbpedia.org/resource/Autism>
<http://dbpedia.org/ontology/diseasesdb> "1142"@en .
```

Σύμφωνα με την παραπάνω τριπλέτα ο πληροφοριακός πόρος ‘Autism’ με URI http://dbpedia.org/resource/Autism συνδέεται μέσω της σχέσης diseasesdb (Βάση δεδομένων ‘Ασθένειες’) με το λεκτικό '1142’ σε αγγλική γλώσσα.

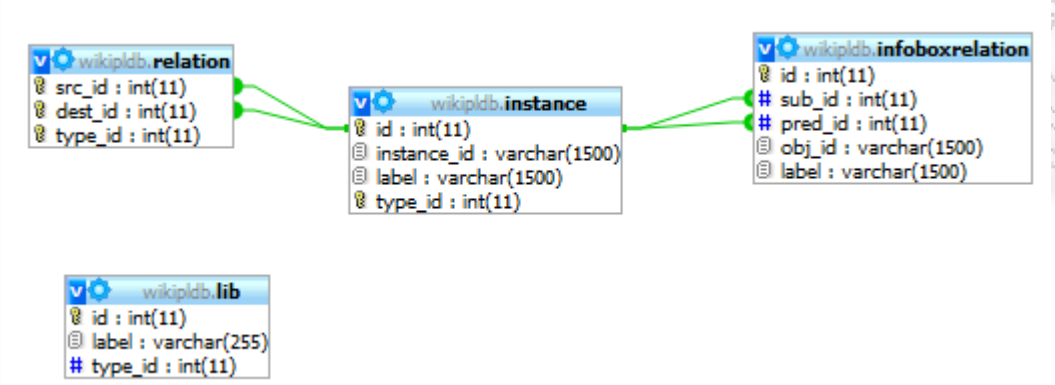
Ας δούμε λοιπόν πώς χρησιμοποιήσαμε αυτά τα διασυνδεδεμένα δεδομένα της DBPedia.

Βάση

Αφού μεταφορτώσαμε από την DBpedia τα αρχεία που θα χρησιμοποιήσουμε δημιουργήσαμε μια βάση για να τα αποθηκεύσουμε (Εικόνα 2). Για τη δημιουργία της βάσης χρησιμοποιήσαμε τα παρακάτω queries:

- CREATE DATABASE `wikipldb` /*!40100 DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci */
- CREATE TABLE `instance` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  -- ένα autoincrement πεδίο ως primary key
  `instance_id` varchar(1500) COLLATE utf8_unicode_ci NOT NULL,
  -- το uri των <subject>, <object> του αρχείου που κατεβάσαμε (χωρίς τα σύμβολα της ανισότητας)
  `label` varchar(1500) COLLATE utf8_unicode_ci NOT NULL,
  -- από το uri, φτιάχνουμε ένα human readable label (από το τελευταίο μέρος του uri, χωρίς underscores, κλπ)
  `type_id` int(11) NOT NULL,
  -- βάζουμε παντού την τιμή 1 εκτός από την σειρά που αναφέρεται στο predicate
  PRIMARY KEY (`id`),
  UNIQUE KEY `id` (`id`,`type_id`),
  KEY `instance_id` (`instance_id`(333))
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
- CREATE TABLE `relation` (
  `src_id` int(11) NOT NULL,
  -- το id του table instance: ουσιαστικά, εδώ βάζουμε τα id των <subject>
  `dest_id` int(11) NOT NULL,
  -- το id του table instance: ουσιαστικά, εδώ βάζουμε τα id των <object>
  `type_id` int(11) NOT NULL,
  -- βάζουμε 8
  PRIMARY KEY (`src_id`,`dest_id`,`type_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
- CREATE TABLE `lib` (
  `id` int(11) NOT NULL,
  `label` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `type_id` int(11) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
- CREATE TABLE `infoboxrelation` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  -- ένα autoincrement πεδίο ως primary key
  `sub_id` int(11) NOT NULL,
  -- το id του table instance: ουσιαστικά, εδώ βάζουμε τα id των <subject>
  `pred_id` int(11) NOT NULL,
  -- το id του table instance: ουσιαστικά, εδώ βάζουμε τα id των <predicates>
  `obj_id` varchar(1500) COLLATE utf8_unicode_ci NOT NULL,
  -- το λεκτικό που βρίσκεται ανάμεσα στα "" ή το URI
  `label` varchar(1500) COLLATE utf8_unicode_ci NOT NULL,

```
-- από το uri, φτιάχνουμε ένα human readable label
 PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
```



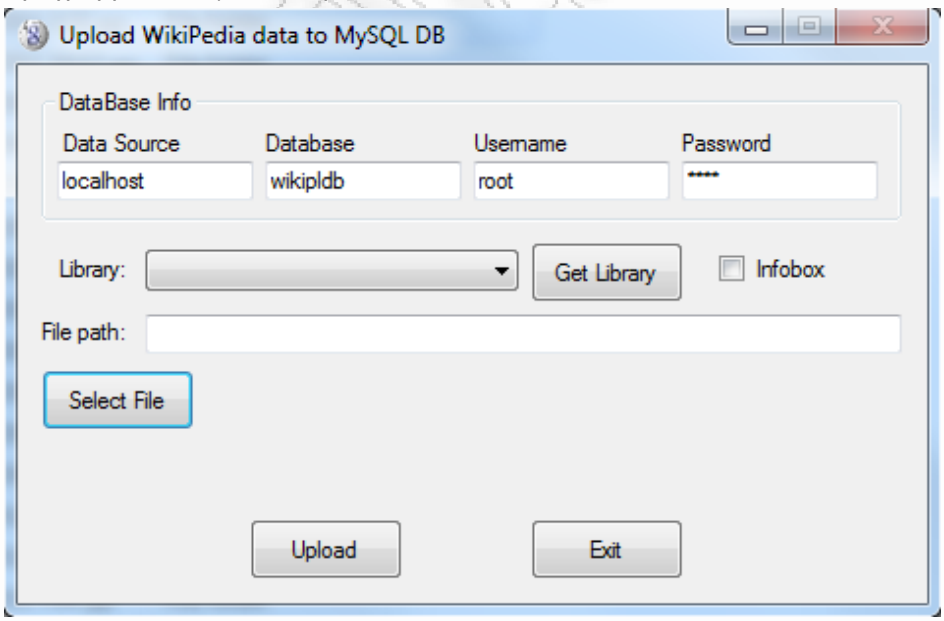
Εικόνα 2: Σχήμα βάσης

Οι πίνακες που μας ενδιαφέρουν είναι οι: instance, relation και infoboxrelation. Ενώ ο πίνακας lib έχει φτιαχτεί για μελλοντική επέκταση της εργασίας. Πιο αναλυτικά, ο πίνακας instance περιέχει όλα τα υποκείμενα, κατηγορήματα και αντικείμενα των αρχείων. Ο πίνακας relation περιέχει την πληροφορία της σύνδεσης του υποκειμένου με το αντικείμενο σε κάθε τριπλέτα και ο πίνακας infoboxrelation την πληροφορία της σύνδεσης του υποκειμένου με το αντικείμενο σε κάθε τριπλέτα του αρχείου Ontology Infobox Properties.

Για να διαβάσουμε τα αρχεία που αντλήσαμε από την DBpedia και να τα φορτώσουμε στην βάση χρησιμοποιήσαμε Visual Basic και MySQL.

Φόρτωμα Βάσης

Αφού κατεβάσαμε τα απαραίτητα αρχεία, προέκυψε η ανάγκη δημιουργίας προγράμματος για το φόρτωμα των δεδομένων τους στην βάση. Έτσι κατασκευάσαμε σε Visual Basic το πρόγραμμα αυτό (Εικόνα 3).



Εικόνα 3: Πρόγραμμα φόρτωσης δεδομένων στην βάση

Το παραπάνω πρόγραμμα είναι πολύ απλό στη χρήση του. Αφού συμπληρώσουμε τα στοιχεία της βάσης στην οποία θα φορτώσουμε τα δεδομένα μας, πατάμε το κουμπί ‘Get Library και η λίστα που βρίσκεται δεξιά του γεμίζει με στοιχεία από τον πίνακα Lib. Ουσιαστικά γεμίζει με τα ονόματα των data sets (Page links, Category κ.α.) Αν ο πίνακας δεν έχει στοιχεία, τότε δημιουργείται και φορτώνονται κάποια default data sets που εχουμε ορίσει εμείς. Το query με τις τιμές που φορτώνει τον πίνακα Lib με τα default data sets είναι το εξής:

```
INSERT INTO `lib` (`id`, `label`, `type_id`) VALUES
(1, 'Article', 1),
(2, 'Disambiguation', 1),
(3, 'Category', 1),
(4, 'Wordnet Category', 1),
(5, 'disambig_by', 2),
(6, 'in_category', 2),
(7, 'in_wordnet_category', 2),
(8, 'reference', 1),
(9, 'refers_to', 2);
```

Αυτές οι τιμές του πεδίου label εμφανίζονται στη λίστα και από εκεί επιλέγουμε ποιο data set θέλουμε να φορτώσουμε. Αν πάλι θέλουμε να φορτώσουμε infobox, τσεκάρουμε το infobox και αγνοούμε εντελώς τη λίστα (η λίστα απενεργοποιείται). Τέλος πατάμε το κουμπί ‘Select File’, επιλέγουμε το αρχείο που θέλουμε να φορτώσουμε και πατάμε το κουμπί ‘Upload’. Στην όλη διαδικασία του upload, ο χρήστης βλέπει ένα progess bar με την πορεία φορτώματος στη βάση, καθώς και πόσες σειρές του αρχείου έχουν διαβαστεί και εισαχθεί στην βάση. Όταν τελειώσει το διάβασμα του αρχείου, ο χρήστης ενημερώνεται με σχετικό μήνυμα στο οποίο και αναγράφεται ο τελικός αριθμός των σειρών (τριπλέτες N3/Turtle) που διαβάστηκαν.

Γραφικό περιβάλλον (GUI)

Στην ενότητα αυτή περιγράφουμε το γραφικό περιβάλλον, το οποίο βοηθάει τις μηχανές αναζήτησης να κάνουν ένα βήμα παραπάνω στην αποτελεσματικότητά τους. Το περιβάλλον αυτό είναι Διαδικτυακό και γράφτηκε σε γλώσσα προγραμματισμού php και javascript ενώ τα δεδομένα όπως προαναφέρθηκε ήταν αποθηκευμένα σε mysql.

Για να ξεκινήσει ο χρήστης την αναζήτησή του έπρεπε να χρησιμοποιήσουμε ένα text box όπου θα πληκτρολογούσε ο χρήστης το ερώτημά του. Έτσι χρησιμοποιήσαμε ένα έτοιμο autosuggest text box [17] (Εικόνα 4).

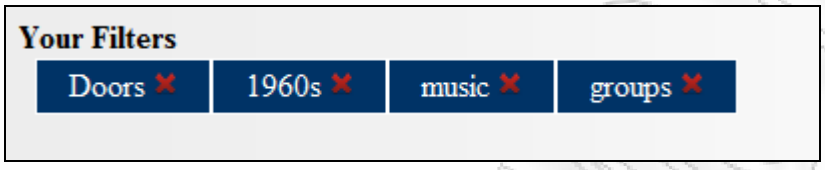
Εικόνα 4: Autosuggest text box

Εδώ ο χρήστης πληκτρολογεί το αρχικό ερώτημά του. Καθώς πληκτρολογεί, εμφανίζεται μια λίστα με τις πιθανές λέξεις. Η λίστα αυτή γεμίζει δυναμικά από το πεδίο ‘label’ του πίνακα

'instance'. Όταν ο χρήστης, επιλέξει κάτι από τη διαθέσιμη λίστα, τότε εμφανίζονται οι διαθέσιμες πληροφορίες (pagelinks, categories κ.λ.π.).

Οι πληροφορίες αυτές είναι στην ουσία λέξεις-κλειδιά τις οποίες μπορεί να επιλέξει ο χρήστης. Αυτόματα οι λέξεις από την επιλογή του χρήστη προστίθενται στη μηχανή αναζήτησης της Google και εμφανίζονται τα αποτελέσματα της αναζήτησης. Πριν όμως γίνει αυτό οι λέξεις φιλτράρονται βάσει μιας λίστας (stopwords [21]) ώστε όσες βρίσκονται στη λίστα να μην προστίθενται στην αναζήτηση. Αυτή η λίστα περιέχει λέξεις τις οποίες οι μηχανές αναζήτησης αγνοούν για να επιταχύνουν την αναζήτησή τους οπότε και εμείς επιλέξαμε να μην τις βάλουμε στην αναζήτηση.

Στη συνέχεια ο χρήστης μπορεί να επιλέξει κάποια από τις διαθέσιμες πληροφορίες, που εμφανίστηκαν μετά την επιλογή, και να προστεθούν και αυτές στη μηχανή αναζήτησης κάνοντας πιο συγκεκριμένο το ερώτημα. Κάθε φορά που ο χρήστης επιλέγει μια πρόσθετη λέξη αυτή προστίθεται και σε μία λίστα από λέξεις ώστε να μπορεί να τις βλέπει όλες και να αφαιρεί όποια από αυτές δεν του κάνει. Αφαιρώντας μια λέξη από τη λίστα, η λέξη αυτή αυτόματα αφαιρείται και από τη μηχανή αναζήτησης, φέρνοντας διαφορετικά αποτελέσματα. Με αυτόν τον τρόπο ο χρήστης μπορεί να αλλάξει εύκολα και γρήγορα το αρχικό του ερώτημα προσθέτοντας ή αφαιρώντας λέξεις-κλειδιά (Εικόνα 5).



Εικόνα 5: Λίστα με λέξεις που έχουν προστεθεί στην αναζήτηση

Εκτός όμως από το στατικό περιεχόμενο που περιέχει τις λέξεις-κλειδιά, μετά την αρχική αναζήτηση, εμφανίζονται στον χρήστη και τα page links (Εικόνα 6). Τα page links είναι οι εσωτερικοί σύνδεσμοι των άρθρων της Wikipedia προς άλλες σελίδες της Wikipedia. Τα page links είναι σε ξεχωριστό σημείο της εφαρμογής γιατί δεν χρησιμοποιούνται σαν λέξεις κλειδιά αλλά βοηθάνε τον χρήστη στο να μεταφερθεί γρήγορα σε μια νέα αναζήτηση που να σχετίζεται κατά κάποιο τρόπο με την αρχική. Για να βοηθήσουμε ακόμα περισσότερο τον χρήστη, έχουμε προσθέσει μια αναζήτηση μέσα στα page links που εμφανίζει δυναμικά τα αποτελέσματα.

Εικόνα 6: Page links που σχετίζονται με την αρχική αναζήτηση

Συνοψίζοντας τα παραπάνω επιμέρους κομμάτια του γραφικού περιβάλλοντος, μαζί με τη μηχανή αναζήτησης που για τις ανάγκες τις εργασίας μας ήταν η Google, προέκυψε ένα άκρως λειτουργικό και πλήρες γραφικό περιβάλλον (Εικόνα 7) που επιτρέπει στον χρήστη να κάνει την αναζήτησή του πιο συγκεκριμένη προσθέτοντας προτεινόμενες λέξεις-κλειδιά χωρίς να τις πληκτρολογεί.

Εικόνα 7: Γραφικό περιβάλλον εφαρμογής

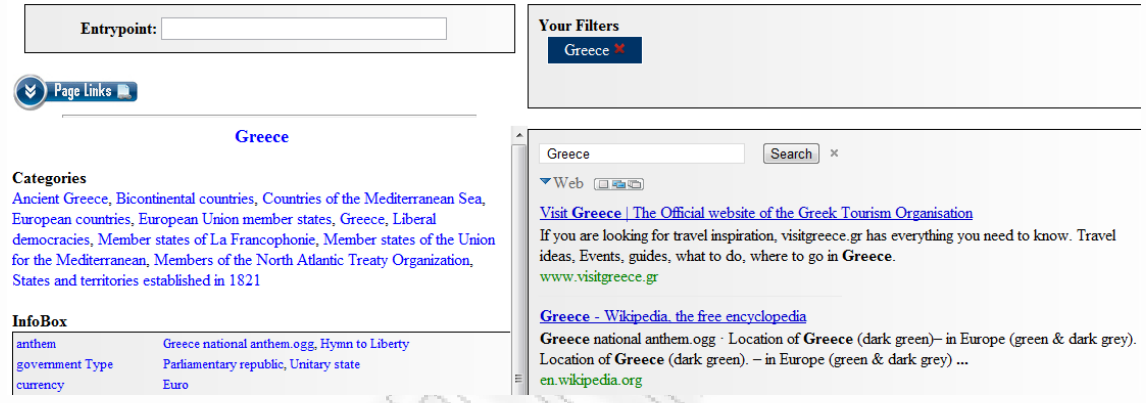
Μελέτες περίπτωσης

Στην ενότητα αυτή θα δούμε πώς καταφέραμε να ξεπεράσουμε τα προβλήματα που αναφέραμε και να βοηθήσουμε τον χρήστη να διατυπώσει ερωτήματα ακριβείας στην μηχανή αναζήτησης.

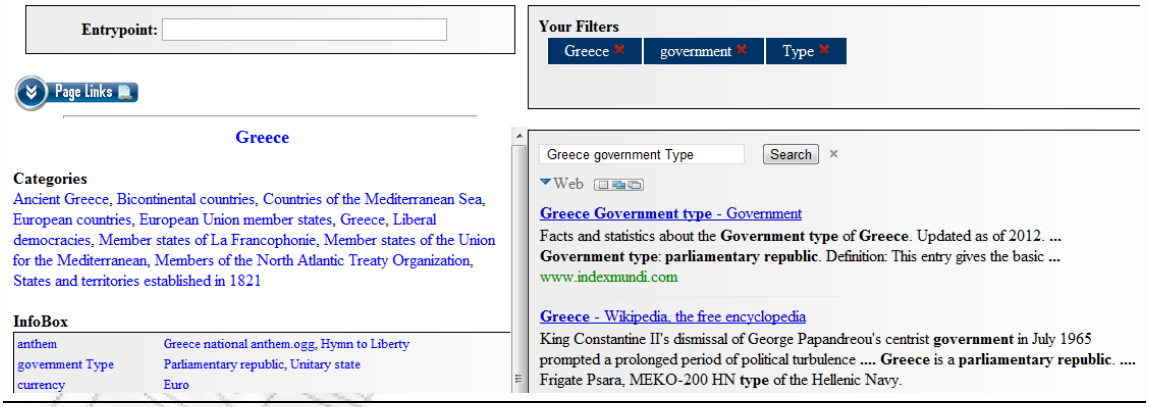
Καταρχάς λύσαμε το πρόβλημα εύρεσης των κατάλληλων λέξεων-κλειδιών με το auto-suggest εργαλείο που χρησιμοποιήσαμε στην εφαρμογή μας. Με τον τρόπο αυτό αντιμετωπίσαμε τους δύο πρώτους τρόπους αναζήτησης πληροφορίας που περιγράφει η Spencer. Σύμφωνα με το μοντέλο αναζήτησης **γνωστό αντικείμενο** ο χρήστης ξέρει τι ψάχνει και ποιες λέξεις πρέπει να χρησιμοποιήσει οπότε ξεκινάει να πληκτρολογεί και εμφανίζεται μια λίστα από λέξεις. Ο χρήστης αναγνωρίζει τη λέξη που ψάχνει και την επιλέγει. Για παράδειγμα, αν ο χρήστης θέλει να ψάξει πληροφορίες για το συγκρότημα 'Doors' ξεκινάει να πληκτρολογεί τη λέξη και εμφανίζεται η λίστα (Εικόνα 8). Ο χρήστης ξέρει τι θέλει οπότε επιλέγει από τη λίστα. Στη συνέχεια μπορεί εύκολα να πλοηγηθεί μέσα από τα διαθέσιμα page links που θα εμφανιστούν.

Εικόνα 8: Αναζήτηση χρήστη για το συγκρότημα 'The Doors'

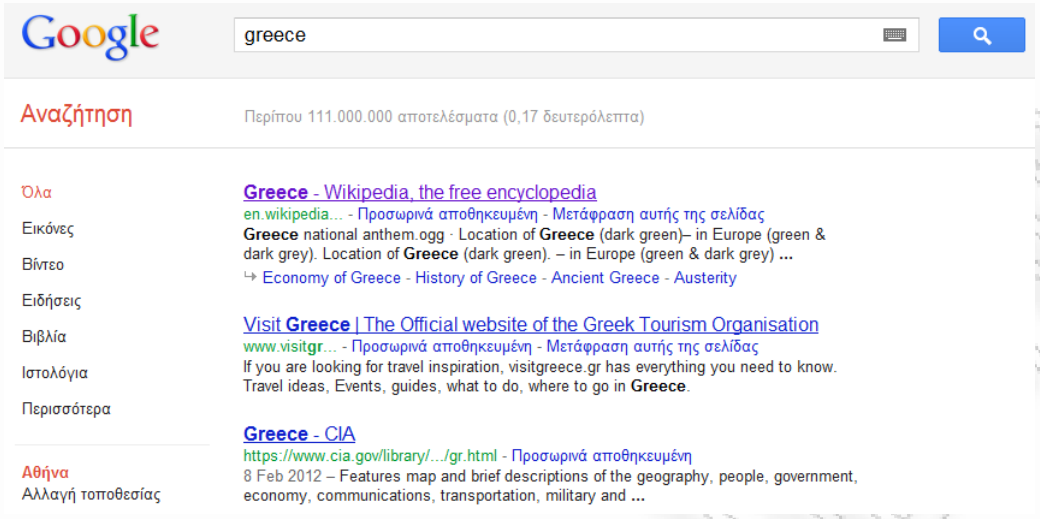
Με τον ίδιο τρόπο αντιμετωπίσαμε το μοντέλο **διερευνητικής** αναζήτησης όπου ο χρήστης ξέρει τι ψάχνει αλλά δεν γνωρίζει ποιες λέξεις πρέπει να χρησιμοποιήσει για να το βρει. Στον χρήστη με το που ξεκινά να πληκτρολογεί, εμφανίζεται μια λίστα με λέξεις που μπορεί να χρησιμοποιήσει. Με αυτόν τον τρόπο όχι μόνο βοηθάμε τον χρήστη να βρει τη λέξη που θέλει, αλλά επιταχύνουμε τη διαδικασία αναζήτησης αφού απλά μπορεί να επιλέξει τη λέξη από τη λίστα. Επιπλέον, με τον τρόπο αυτό αποτρέπουμε τα ορθογραφικά λάθη που τυχόν μπορεί να έκανε ο χρήστης. Για παράδειγμα, αν ο χρήστης θέλει να βρει πληροφορίες για το πολίτευμα της Ελλάδας αλλά δεν ξέρει ποιες λέξεις να χρησιμοποιήσει για να ικανοποιήσει αυτή την πληροφοριακή του ανάγκη, πληκτρολογεί απλά τη λέξη ‘Greece’ (Εικόνα 9). Από τις επιλογές που θα του εμφανιστούν επιλέγει το ‘government type’ και εμφανίζεται το επιθυμητό αποτέλεσμα (Εικόνα 10). Ας δούμε όμως τι θα γινόταν αν ο χρήστης έψαχνε απευθείας στη μηχανή αναζήτησης και όχι μέσω της εφαρμογής που αναπτύξαμε. Τα αποτελέσματα που θα του επέστρεφε η μηχανή αναζήτησης θα ήταν χαοτικά αν ο χρήστης χρησιμοποιούσε τη λέξη ‘Greece’ (Εικόνα 11). Για να βρει στη συνέχεια το πολίτευμα της Ελλάδας, θα έπρεπε να ανοίξει τα αποτελέσματα της αναζήτησης και να ψάξει σε καθένα πληροφορίες για το πολίτευμα της χώρας χωρίς όμως να είναι και βέβαιο ότι θα το βρει. Πράγμα που είναι ιδιαίτερα χρονοβόρο.



Εικόνα 9: Αναζήτηση χρήστη για τη ‘Ελλάδα’



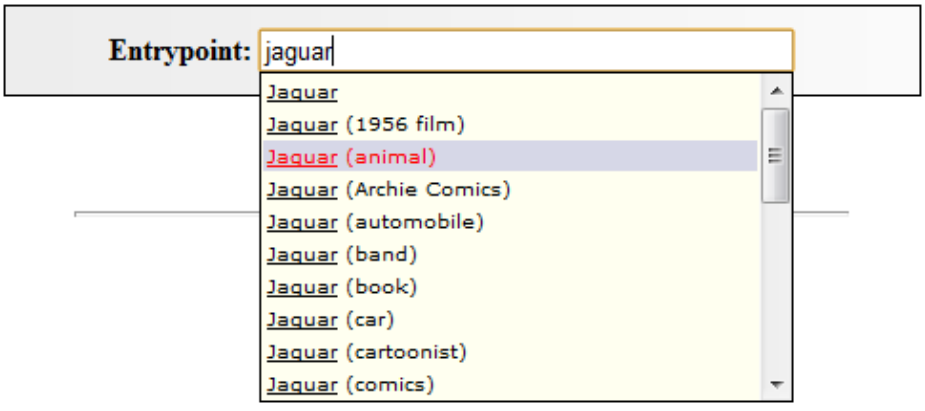
Εικόνα 10: Αναζήτηση χρήστη για το πολίτευμα της Ελλάδας



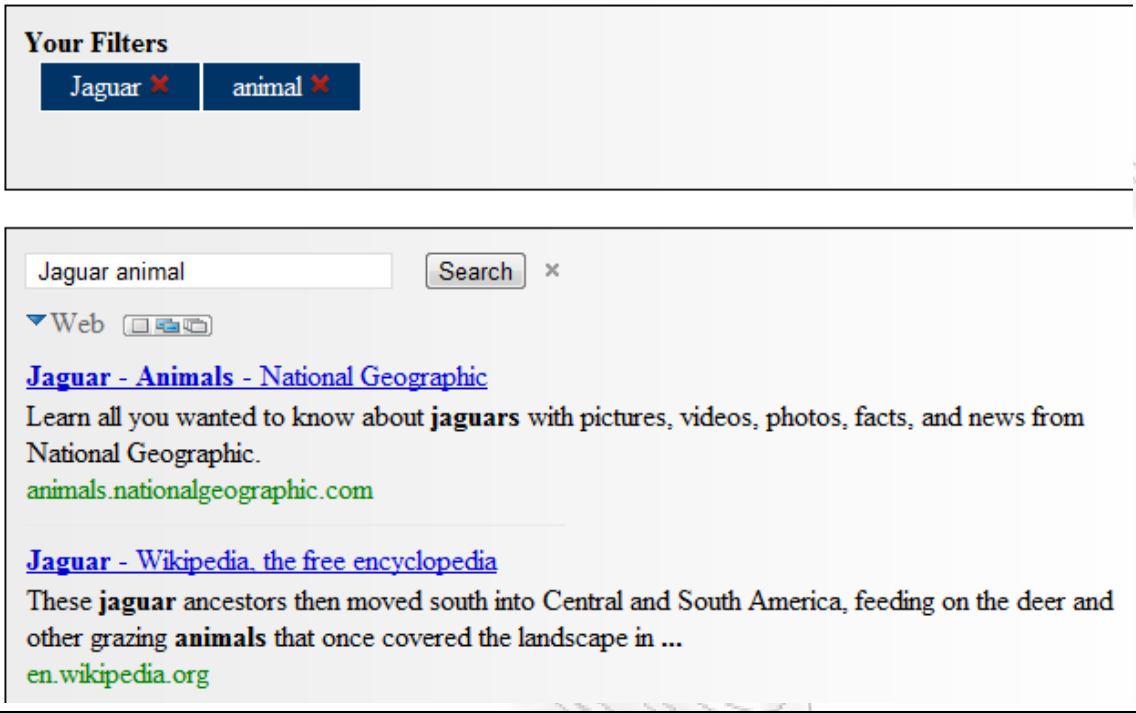
Εικόνα 11: Αποτελέσματα αναζήτησης για την λέξη 'Greece'

Μετά την επιλογή της λέξης εμφανίζονται οι κατάλληλες πληροφορίες που θα βοηθήσουν τον χρήστη να κάνει το ερώτημά του πιο συγκεκριμένο. Επιλέγοντας οποιαδήποτε λέξη από τις πρόσθετες αυτές πληροφορίες, αυτή προστίθεται αυτόματα στην αναζήτησή του. Ο χρήστης μπορεί με τον τρόπο αυτό να προσθέσει αλλά και να αφαιρέσει λέξεις-κλειδιά. Με τον τρόπο αυτό αντιμετωπίσαμε το μοντέλο αναζήτησης **δεν γνωρίζω τι ακριβώς ψάχνω** της Spencer, όπου οι χρήστες πληκτρολογούν μια λέξη αλλά στην ουσία ψάχνουν κάτι άλλο. Έτσι βοηθάμε τον χρήστη στον να επιλύσει το πρόβλημα του εντοπισμού των κατάλληλων λέξεων-κλειδιών που απαιτούνται για βρει αυτό που ψάχνει.

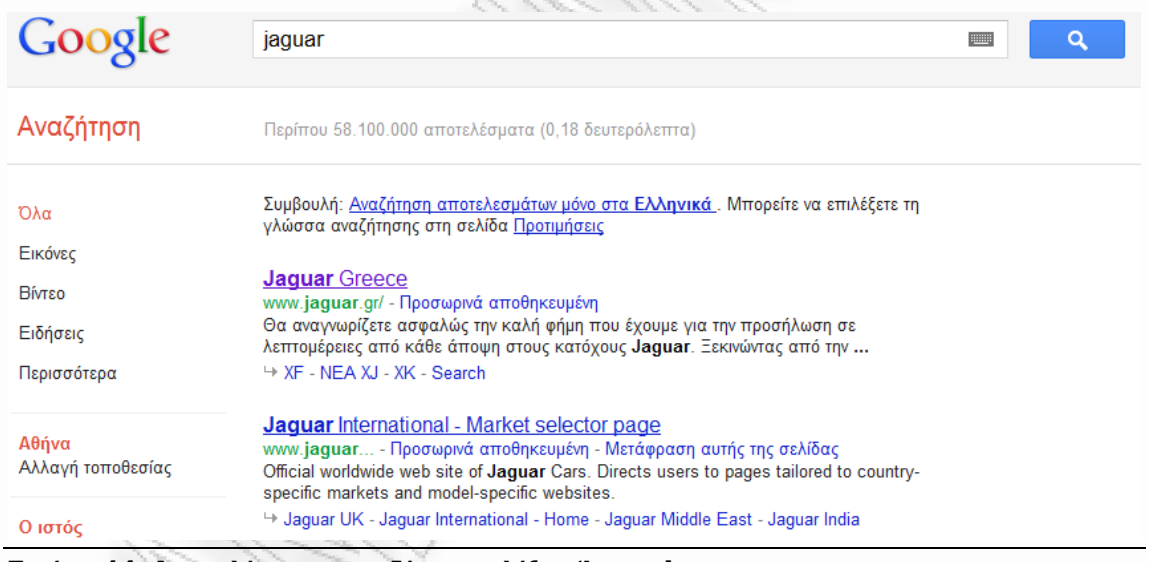
Ένα άλλο σημαντικό πρόβλημα που επιδιώκει να λύσει η εφαρμογή είναι το πρόβλημα της πολλαπλής σημασίας μιας λέξης. Αν ο χρήστης επιλέξει μια λέξη που έχει πολλές σημασίες, η εφαρμογή θα εμφανίζει την κατηγορία Disambiguation που θα περιέχει τις σημασίες αυτές. Ο χρήστης μπορεί να επιλέξει τη σημασία της λέξης που θέλει και αυτή θα προστεθεί αυτόματα στη μηχανή αναζήτησης ώστε να βοηθήσει τη μηχανή να κατανοήσει τι πραγματικά ψάχνει ο χρήστης. Για παράδειγμα, αν ο χρήστης θέλει να βρει πληροφορίες για το ζώο jaguar (τζάγκουαρ), ξεκινάει να πληκτρολογεί τη λέξη jaguar και του εμφανίζεται η λίστα με τις διαθέσιμες επιλογές (Εικόνα 12). Από τις επιλογές που θα του εμφανιστούν επιλέγει το 'Jaguar (animal)' και αμέσως οι λέξεις-κλειδιά 'jaguar' και 'animal' προστίθενται στη μηχανή αναζήτησης και ο χρήστης παίρνει τα αποτελέσματα που ψάχνει (Εικόνα 13). Τα αποτελέσματα είναι τελείως διαφορετικά αν ο χρήστης χρησιμοποιήσει απευθείας τη μηχανή αναζήτησης και όχι μέσω της εφαρμογής μας για να καλύψει την ίδια πληροφοριακή του ανάγκη (Εικόνα 14). Η μηχανή του επιστρέφει μεικτά αποτελέσματα που αφορούν και το ζώο jaguar αλλά και το αυτοκίνητο βάζοντας πιο ψηλά το αυτοκίνητο σύμφωνα με τα δικά της κριτήρια αξιολόγησης.



Εικόνα 12: Αναζήτηση χρήστη για το αυτοκίνητο 'jaguar'



Εικόνα 13: Αποτελέσματα αναζήτησης λέξης 'jaguar (car)'



Εικόνα 14: Αποτελέσματα αναζήτησης λέξης 'jaguar'

Τέλος, η εφαρμογή παρέχει στον χρήστη τα Wikipedia page links (εσωτερικοί σύνδεσμοι της Wikipedia). Αυτό βοηθάει τον χρήστη να εντοπίσει αυτό που ψάχνει όταν δεν ξέρει ή δεν θυμάται τις λέξεις που πρέπει να χρησιμοποιήσει. Με τον τρόπο αυτό αντιμετωπίσαμε το μοντέλο αναζήτησης **επανεύρεσης** της Spencer, όπου οι χρήστες ψάχνουν κάτι που έχουν ξανασυναντήσει σε παλαιότερο ερώτημα. Μπορούν έτσι να πληκτρολογήσουν μια λέξη-κλειδί και μετά να πλοηγηθούν στα page links για να το βρουν. Για παράδειγμα, αν ο χρήστης ψάχνει πληροφορίες για το τραγούδι 'Imagine' του 'John Lennon' αλλά δεν θυμάται τον τίτλο, μπορεί να αναζητήσει τη λέξη 'John Lennon' και από τα page links που θα του επιστρέψει η εφαρμογή να βρει τον τίτλο του τραγουδιού που ψάχνει, στο παράδειγμα μας το 'Imagine'. Επιλέγοντας τον τίτλο του τραγουδιού από τα page links η εφαρμογή θα ξεκινήσει μια νέα αναζήτηση με βάση τον τίτλο και θα φέρει νέες πληροφορίες και page links.

Συμπεράσματα

Οι σημερινές μηχανές αναζήτησης είναι εντελώς τυποποιημένες και δεν καλύπτουν τις ανάγκες των χρηστών. Η εκτεταμένη χρήση συναφών λέξεων μπορεί να οδηγήσει ένα βήμα μπροστά την αναζήτηση πληροφοριών στο Διαδίκτυο μέσω των μηχανών αναζήτησης. Στη διατριβή αυτή παρουσιάσαμε ένα νέο μοντέλο αναζήτησης με την χρήση συναφών λέξεων. Για να αντλήσουμε τις λέξεις αυτές βασιστήκαμε στα άρθρα της Wikipedia και πιο συγκεκριμένα στα διασυνδεδεμένα δεδομένα της DBPedia.

Παρουσιάσαμε λοιπόν ένα πλήρες λειτουργικό περιβάλλον που βοηθά τον χρήστη να χρησιμοποιήσει πιο αποτελεσματικά τη μηχανή αναζήτησης. Στην ουσία του παρέχουμε λέξεις-κλειδιά που τον βοηθάνε να κάνει το ερώτημά του πιο συγκεκριμένο. Με αυτόν τον τρόπο ο χρήστης δεν αναλώνει τον χρόνο του ψάχνοντας να βρει τις κατάλληλες λέξεις αφού απλά τις επιλέγει και αυτές προστίθενται στο αρχικό του ερώτημα.

Η προσέγγιση που παρουσιάσαμε σε αυτή την εργασία μπορεί να αποδειχθεί ιδιαίτερα χρήσιμη στους χρήστες που χρησιμοποιούν τις μηχανές αναζήτησης. Θα μπορούσε να είναι το επόμενο βήμα για την εξέλιξη των μηχανών αυτών. Καθώς η χρήση των μηχανών αναζήτησης γίνεται αναπόσπαστο κομμάτι της καθημερινότητας του χρήστη, η ανάγκη για καλύτερες μηχανές αναζήτησης είναι επιτακτική.

Βιβλιογραφία

[1] Auer S., Bizer C., Kobilarov G., Lehmann J., Cyganiak R., Ives Z., DBpedia: a nucleus for a web of open data, in: ISWC, volume 4825 of LNCS, pp. 722-735, 2007.

[2] Lewandowski D., Wahlig H., Meyer-Bautor G., The freshness of web search engine databases, in: Journal of Information Science vol. 32, pp. 2 131-148, April 2006.

[3] Bizer C., Heath T., Berners-Lee T., linked Data - the story so far, in: International Journal on Semantic Web and Information Systems, Volume 5, Issue 3, pp. 1-22, 2009.

[4] Papadakis I., Stamou S., Stefanidakis M., Andreou I., A Query Construction Service for large-scale Web Search Engines, in: Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT '09. IEEE/WIC/ACM International Joint Conferences, pp. 96-99, 15-18 Sept. 2009.

[5] Lawrence S., Context in Web search. IEEE Data Engineering Bulletin, 23(3), pp. 25-32, 2000.

[6] Γιατράς Δ., πτυχιακή εργασία: Συνδεδεμένα δεδομένα: Επισκόπηση της τεχνολογίας και των υπαρχόντων συστημάτων, διαθέσιμο από: http://invenio.lib.auth.gr/record/127148/files/%CE%A0%CF%84%CF%85%CF%87%CE%B9%CE%B1%CE%BA%CE%AE.pdf, πρόσβαση: 04/12/2011.

[7] Μπουλογεώργου-Νημά Ε., Διπλωματική εργασία: Σημασιολογικός Ιστός και Συνδεδεμένα Δεδομένα: Εφαρμογή στην ηλεκτρονική διακυβέρνηση, διαθέσιμο από: http://dspace.lib.uom.gr/bitstream/2159/13816/1/Boulogeorgou_Msc2010.pdf, πρόσβαση: 04/12/2011.

[8] Broder A., A taxonomy of web search. In SIGIR Forum 36(2), pp. 3-10, 2002.

[9] Qui F. Cho J. Automatic Identification of User Interest for Personalized Search. In Proceedings of the 15th Intl. World Wide Web Conference, pp. 727-236, 2006.

[10] Broder, A. 2002. A taxonomy of web search. In SIGIR Forum, 36(2).

[11] Spink, A., Park, M., Jansen, B. and Pedersen, O. 2006. Multitasking during web search session. In Information Processing and Management, 42(5): 1366-1378.

[12] Palta E., Εργασία με θέμα: Word Sense Disambiguation, διαθέσιμο από: http://www.it.iitb.ac.in/~esha/resources/firststage.pdf, πρόσβαση: 07/02/2012.

Ιστογραφία

[13] Wikipedia, διαθέσιμο από: http://en.wikipedia.org, πρόσβαση: 05/01/2012.

[14] DBPedia, διαθέσιμο από: http://dbpedia.org/About, πρόσβαση: 02/01/2012.

[15] DBPedia Downloads36, διαθέσιμο από: http://wiki.dbpedia.org/Downloads36, πρόσβαση: 01/12/2011.

[16] MySQl, διαθέσιμο από: http://www.mysql.com/, πρόσβαση :05/01/2012.

[17] Auto-suggest Control, διαθέσιμο από: http://www.codeproject.com/KB/scripting/AutoSuggestControl.aspx, πρόσβαση: 15/09/2011.

[18] AJAX Tutorial, διαθέσιμο από: http://www.w3schools.com/ajax/default.asp, πρόσβαση: 15/09/2012.

[19] PHP Tutorial, διαθέσιμο από: http://www.w3schools.com/php/, πρόσβαση: 15/09/2011.

[20] Google Web Search API, διαθέσιμο από: http://code.google.com/apis/websearch/, πρόσβαση: 18/11/2011.

[21] Stop Words, διαθέσιμο από: http://www.webconfs.com/stop-words.php, πρόσβαση: 10/12/2011.

[22] W3C, διαθέσιμο από: http://www.w3.org/, πρόσβαση: 02/12/2011.

[23] RDF Concepts, διαθέσιμο από: http://www.w3.org/TR/rdf-concepts/, πρόσβαση: 02/12/2011.

[24] Turtle - Terse RDF Triple Language, διαθέσιμο από: http://www.w3.org/TeamSubmission/turtle/, πρόσβαση: 02/12/2011.

[25] WordNet A lexical database for English, διαθέσιμο από: http://wordnet.princeton.edu/, πρόσβαση 08/10/2011.

[26] Linked Data - Connect Distributed Data across the Web, διαθέσιμο από: http://linkeddata.org/, πρόσβαση 05/12/2011.

[27] W3C Semantic Web activity, διαθέσιμο από: http://www.w3.org/2001/sw/, πρόσβαση 05/12/2011.

[28] N-Triples, διαθέσιμο από: http://www.w3.org/2001/sw/RDFCore/ntriples/, πρόσβαση 05/12/2011.

[29] SPARQL Query Language for RDF, διαθέσιμο από: http://www.w3.org/TR/rdf-sparql-query/, πρόσβαση: 16/12/2011.

[30] Notation3 (N3): A readable RDF syntax, διαθέσιμο από: http://www.w3.org/TeamSubmission/n3/, πρόσβαση: 16/12/2011.

[31] Turtle - Terse RDF Triple Language, διαθέσιμο από: http://www.w3.org/TeamSubmission/turtle/, πρόσβαση: 18/12/2011.

[32] How Google Instant's Autocomplete Suggestions Work, διαθέσιμο από: http://searchengineland.com/how-google-instant-autocomplete-suggestions-work-62592, πρόσβαση: 15/01/2012.

[33] Four Modes of Seeking Information and How to Design for them, διαθέσιμο από: http://www.boxesandarrows.com/view/four_modes_of_seeking_information_and_how_to_design_for_them, πρόσβαση: 17/01/2012.

[34] The Linking Open Data cloud diagram, διαθέσιμο από: http://lod-cloud.net, πρόσβαση: 06/02/2012.

[35] Triplestore, διαθέσιμο από: http://en.wikipedia.org/wiki/Triplestore, πρόσβαση: 06/02/2012.

Παράρτημα: Ο πηγαίος κώδικας με σχόλια

Στην ενότητα αυτή παρουσιάζουμε τον πηγαίο κώδικα της εφαρμογής με επεξηγήσεις και σχόλια.

Πρόγραμμα που φορτώνει τα δεδομένα της DBPedia στην βάση

Το πρόγραμμα αυτό χρησιμοποιείται για το φόρτωμα της βάσης με τα δεδομένα της DBPedia. Έτσι αφού επιλέξει ο χρήστης το αρχείο που θέλει να φορτώσει στην βάση, επιλέγει τι είδος αρχείου δεδομένων θέλει να ανεβάσει (Pagelinks, Categories κ.λ.π.) ή αν το αρχείο είναι infobox και πατάει το κουμπί 'Upload'.

Αν το αρχείο που θέλει να ανεβάσει ο χρήστης δεν είναι infobox, πρέπει ο χρήστης να πατήσει το κουμπί Get Library ώστε η εφαρμογή να γεμίσει την λίστα της εφαρμογής Library από τον πίνακα Lib. Αν ο πίνακας Lib είναι άδειος, τότε η εφαρμογή βγάζει σχετικό μήνυμα και μας προτείνει να γεμίσει τον πίνακα και συνεπώς την λίστα με τις default τιμές (Σελίδα 15). Πατώντας το κουμπί 'Upload' η εφαρμογή γεμίζει τους πίνακες instance και relation. Διαβάζει το αρχείο που του δίνει ο χρήστης γραμμή-γραμμή και με λίγα λόγια κάνει τα εξής:

I. Αποθηκεύει σε μεταβλητές το URI του υποκειμένου, κατηγορήματος και αντικειμένου που βρίσκεται ανάμεσα στα '<' και '>'.
II. Αποθηκεύει σε μεταβλητές το τελευταίο μέρος του URI του υποκειμένου, κατηγορήματος και αντικειμένου.
III. Κάνει αποκρυπτογράφηση URL χαρακτήρων των παραπάνω μεταβλητών σε ASCII.
IV. Εισάγει το κατηγόρημα στην βάση instance μια φορά αφού είναι ίδιο σε όλες τις εγγραφές του αρχείου.
V. Ελέγχει το μήκος του URI έτσι ώστε σε περίπτωση που είναι μεγαλύτερο από το αρχικά δηλωμένο να αλλάξει και το μέγεθος των αποδεκτών χαρακτήρων στην βάση.
VI. Ψάχνει στην βάση instance για να δει αν υπάρχει ήδη εγγραφή για το συγκεκριμένο υποκείμενο και αντικείμενο που θέλει να προσθέσει.
VII. Αν υπάρχουν ήδη εγγραφές του υποκειμένου και αντικειμένου στον πίνακα instance, κρατάει τα id τους και ενημερώνει τον πίνακα relation για την σχέση τους αν αυτή δεν υπάρχει.
VIII. Αν δεν υπάρχουν εγγραφές του υποκειμένου και αντικειμένου στον πίνακα instance, τα προσθέτει, κρατάει τα id τους και ενημερώνει τον πίνακα relation για την σχέση τους αν αυτή δεν υπάρχει.

Αν το αρχείο που θέλει να ανεβάσει ο χρήστης είναι infobox, πρέπει ο χρήστης να τσεκάρει το 'Infobox'. Πατώντας το κουμπί 'Upload' η εφαρμογή γεμίζει τους πίνακες instance και infoboxrelation. Διαβάζει το αρχείο που του δίνει ο χρήστης γραμμή-γραμμή και με λίγα λόγια κάνει τα εξής:

I. Αποθηκεύει σε μεταβλητές το URI του υποκειμένου και κατηγορήματος που βρίσκεται ανάμεσα στα '<' και '>'.
II. Κάνει έλεγχο για το αν το αντικείμενο είναι URI ή λεκτικό και το αποθηκεύει και αυτό σε μεταβλητή.
III. Αποθηκεύει σε μεταβλητές το τελευταίο μέρος του URI του υποκειμένου, κατηγορήματος και αντικειμένου (αν είναι URI ή απλά το ίδιο το λεκτικό).
IV. Κάνει αποκρυπτογράφηση URL χαρακτήρων των παραπάνω μεταβλητών σε ASCII.
V. Ελέγχει το μήκος του URI έτσι ώστε σε περίπτωση που είναι μεγαλύτερο από το αρχικά δηλωμένο να αλλάξει και το μέγεθος των αποδεκτών χαρακτήρων στην βάση.
VI. Ψάχνει στην βάση instance για να δει αν υπάρχει ήδη εγγραφή για το συγκεκριμένο υποκείμενο και κατηγόρημα που θέλει να προσθέσει.
VII. Αν υπάρχουν ήδη εγγραφές του υποκειμένου και κατηγορήματος στον πίνακα instance, κρατάει τα id τους και ενημερώνει τον πίνακα infoboxrelation για την σχέση τους αν αυτή δεν υπάρχει προσθέτοντας και το αντικείμενο που έχει αποθηκεύσει.

VIII. Αν δεν υπάρχουν εγγραφές του υποκειμένου και κατηγορήματος στον πίνακα instance, τα προσθέτει, κρατάει τα id τους και ενημερώνει τον πίνακα infoboxrelation για την σχέση τους αν αυτή δεν υπάρχει προσθέτοντας και το αντικείμενο που έχει αποθηκεύσει.

Ο πηγαίος κώδικας της εφαρμογής με σχόλια είναι ο εξής:

```
Imports Microsoft.VisualBasic.FileIO
Imports MySql.Data.MySqlClient
Imports System.Text
Imports System.Text.RegularExpressions

Public Class UploadDataFrm
    'Disable Close form button code
    Protected Overrides ReadOnly Property CreateParams() As CreateParams
        Get
            Dim cp As CreateParams = MyBase.CreateParams
            Const CS_DBLCLKS As Int32 = &H8
            Const CS_NOCLOSE As Int32 = &H200
            cp.ClassStyle = CS_DBLCLKS Or CS_NOCLOSE
            Return cp
        End Get
    End Property

    Private Property FileListString As Object 'Variable to store parshed file path
    Dim connection As New MySqlConnection

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BrowseBtn.Click
        OpenFileDialog1.InitialDirectory = "C:\" 'Default open dialog directory
        OpenFileDialog1.Filter = "nt files (*.nt)|*.nt|All files (*.*)|*.*" 'Default open dialog files
        OpenFileDialog1.FileName = "page_links_en" 'Default open dialog file name
        If OpenFileDialog1.ShowDialog() = System.Windows.Forms.DialogResult.OK Then
            FilePathTxt.Text = OpenFileDialog1.FileName 'Store path to textbox
        End If
    End Sub

    Private Sub UploadBtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles UploadBtn.Click
        If (FilePathTxt.Text = "") Then 'Check for empty text box and combobox
            MsgBox("Please select a file.")
            Exit Sub
        End If
        LibCB.Enabled = False
        LibBtn.Enabled = False
        BrowseBtn.Enabled = False
        FilePathTxt.Enabled = False
```

```
        UploadBtn.Enabled = False
        DSTxt.Enabled = False
        DBTxt.Enabled = False
        UsernameTxt.Enabled = False
        PasswordTxt.Enabled = False
        UploadLbl.Visible = True
        UploadPB.Visible = True
        Dim recount As Long 'Record counter
        recount = 0
        Dim myconnstring As String
        myconnstring = "Database=" & DBTxt.Text & ";Data Source=" & DSTxt.Text & ";User Id=" & UsernameTxt.Text & ";Password=" & PasswordTxt.Text & ";Charset=utf8;Default Command Timeout=180" 'SQL connection string
        If InfoboxCB.Checked = False Then
            If (String.IsNullOrEmpty(LibCB.Text)) Then 'Check for empty text box and combobox
                MsgBox("Please select Library and a file.")
                Exit Sub
            End If
            Dim subjectstr, predicatestr, objectstr, friendlysubjectstr, friendlyobjectstr, friendlypredicatestr, findcateg As String
            Dim runonce As Integer 'Variable for predicate so it will be stored to database on time
            runonce = 0
            Try
                connection.ConnectionString = myconnstring
                connection.Open()
                Dim subjectid, objectid, relation, strlen, catfound, typeid As Integer
                strlen = 1500
                Dim reader As MySqlDataReader 'Get Combobox selection lib id for relation table
                Dim sqllibid As New MySqlCommand("select id from lib where label = '" & LibCB.Text & "';", connection)
                reader = sqllibid.ExecuteReader()
                While reader.Read()
                    typeid = reader("id")
                End While
                reader.Close()

                Dim FileLineString As String
                FileListString = New TextFieldParser(FilePathTxt.Text) 'Store file path to string
                Do Until FileListString.EndOfData 'Search file until end
                    subjectid = 0
                    objectid = 0
                    relation = 0
                    catfound = 0
                    FileLineString = FileListString.ReadLine() 'Read file line by line
                    subjectstr = Split(Split(FileLineString, "<")(1), ">")(0) 'Store string between first < and >
```

```
            objectstr = Split(Split(FileLineString, "<")(3), ">")(0) 'Store string between third < and >
            friendlysubjectstr = subjectstr.Substring((subjectstr.LastIndexOf("/") + 1), (subjectstr.Length - subjectstr.LastIndexOf("/") - 1)) 'Store string between last / and the end
            friendlyobjectstr = objectstr.Substring((objectstr.LastIndexOf("/") + 1), (objectstr.Length - objectstr.LastIndexOf("/") - 1)) 'Store string between last / and the end
            If Not LibCB.Text.ToUpper.Contains("CATEGORY") Then 'Check if selected library is category if not a category do not put entries in db starting with word cat or category
              If friendlyobjectstr.Length > 4 Then ' Check if friendlyobjectstr string length is more than 4 charcters before substring
                findcateg = friendlyobjectstr.Substring(0, 4) 'Store in variable findcateg the first 4 charachters of friendlyobjectstr
                If findcateg.ToUpper.Contains("CAT:") Then
                  catfound = 1
                End If
              End If
              If friendlyobjectstr.Length > 10 Then ' Check if friendlyobjectstr string length is more than 10 charcters before substring
                findcateg = friendlyobjectstr.Substring(0, 9) 'Store in variable findcateg the first 9 charachters of friendlyobjectstr
                If findcateg.ToUpper.Contains("CATEGORY:") Then
                  catfound = 1
                End If
                findcateg = friendlyobjectstr.Substring(0, 10) 'Store in variable findcateg the first 10 charachters of friendlyobjectstr
                If findcateg.ToUpper.Contains("CATEGORY :") Then
                  catfound = 1
                End If
              End If
            End If

            If catfound = 0 Then 'If the object is not a category
              friendlysubjectstr = Uri.UnescapeDataString(friendlysubjectstr) 'url encoding to ASCII
              friendlysubjectstr = friendlysubjectstr.Replace("_", " ") 'replace underscore with blank
              friendlysubjectstr = friendlysubjectstr.Replace("\", "\\") 'delete \
              friendlysubjectstr = friendlysubjectstr.Replace("'", "\'") 'delete '
              friendlysubjectstr = friendlysubjectstr.Replace("′", "\′") 'delete ′
              friendlyobjectstr = Uri.UnescapeDataString(friendlyobjectstr) 'url encoding to ASCII
              friendlyobjectstr = friendlyobjectstr.Replace("_", " ") 'replace underscore with blank
              friendlyobjectstr = friendlyobjectstr.Replace("\", "\\") 'delete \
              friendlyobjectstr = friendlyobjectstr.Replace("'", "\'") 'delete '
              friendlyobjectstr = friendlyobjectstr.Replace("′", "\′") 'delete ′
              If runonce = 0 Then 'Runonce to insert predicate in table instance
```

```
                predicatestr = Split(Split(FileLineString, "<")(2), ">")(0) 'Store string between
second < and >
                friendlypredicatestr = predicatestr.Substring((predicatestr.LastIndexOf("/") +
1), (predicatestr.Length - predicatestr.LastIndexOf("/") - 1)) 'Store string between last / and the
end
                friendlypredicatestr = Uri.UnescapeDataString(friendlypredicatestr) 'url
encoding to ASCII
                friendlypredicatestr = friendlypredicatestr.Replace("_", " ") 'replace underscore
with blank
                friendlypredicatestr = friendlypredicatestr.Replace("\", "\\") 'delete \
                friendlypredicatestr = friendlypredicatestr.Replace("'", "\'") 'delete '
                friendlypredicatestr = friendlypredicatestr.Replace("""", "\""") 'delete "
                Try
                  Dim sqlcomm As New MySqlCommand("insert into instance (instance_id,
label, type_id) VALUES ('" & predicatestr & "', '" & friendlypredicatestr & "', 2);", connection)
                  sqlcomm.ExecuteNonQuery()
                  sqlcomm = Nothing
                Catch ex As Exception
                  MsgBox(ex.Message)
                End Try
                runonce = 1
              End If
              If strlen < subjectstr.Length Then ' Alter navchar max depending on subjectstr
length
                strlen = subjectstr.Length
                Dim sqluplen As New MySqlCommand("ALTER TABLE `wikipldb`.`instance`
CHANGE COLUMN `instance_id` `instance_id` VARCHAR(" & strlen & ") CHARACTER SET
'utf8' COLLATE 'utf8_unicode_ci' NOT NULL , CHANGE COLUMN `label` `label` VARCHAR("
& strlen & ") CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL;", connection)
                sqluplen.ExecuteNonQuery()
                sqluplen = Nothing
              End If
              If strlen < objectstr.Length Then ' Alter navchar max depending on objectstr
length
                strlen = objectstr.Length
                Dim sqluplen As New MySqlCommand("ALTER TABLE `wikipldb`.`instance`
CHANGE COLUMN `instance_id` `instance_id` VARCHAR(" & strlen & ") CHARACTER SET
'utf8' COLLATE 'utf8_unicode_ci' NOT NULL , CHANGE COLUMN `label` `label` VARCHAR("
& strlen & ") CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL;", connection)
                sqluplen.ExecuteNonQuery()
                sqluplen = Nothing
              End If

              While (subjectid = 0 Or objectid = 0)
                ' Search if subject already in table "instance" and store subject id

                Dim sqlsubj As New MySqlCommand("select id from instance where
instance_id = '" & subjectstr & "';", connection)
```

```
                reader = sqlsubj.ExecuteReader()
                While reader.Read()
                   subjectid = reader("id")
                End While
                reader.Close()

                ' Search if object already in table "instance" and store object id
                Dim sqlobj As New MySqlCommand("select id from instance where instance_id = '" & objectstr & "';", connection)
                reader = sqlobj.ExecuteReader()
                While reader.Read()
                   objectid = reader("id")
                End While
                reader.Close()

                ' If subject not found in table "instance", store it
                If subjectid = 0 Then
                   Try
                      Dim sqlcomm As New MySqlCommand("insert into instance (instance_id, label, type_id) VALUES ('" & subjectstr & "', '" & friendlysubjectstr & "', 1);", connection)
                      sqlcomm.ExecuteNonQuery()
                      sqlcomm = Nothing
                   Catch ex As Exception ' Unknown characters found
                      friendlysubjectstr = Regex.Replace(friendlysubjectstr, "[^a-zA-Z0-9- .:,()]", "") ' Strip unknown characters and store it in table "instance"
                      Dim sqlcomm As New MySqlCommand("insert into instance (instance_id, label, type_id) VALUES ('" & subjectstr & "', '" & friendlysubjectstr & "', 1);", connection)
                      sqlcomm.ExecuteNonQuery()
                      sqlcomm = Nothing
                   End Try

                End If

                ' If object not found in table "instance", store it
                If objectid = 0 Then
                   Try
                      Dim sqlcomm As New MySqlCommand("insert into instance (instance_id, label, type_id) VALUES ('" & objectstr & "', '" & friendlyobjectstr & "', 1);", connection)
                      sqlcomm.ExecuteNonQuery()
                      sqlcomm = Nothing
                   Catch ex As Exception ' Unknown characters found
                      friendlyobjectstr = Regex.Replace(friendlyobjectstr, "[^a-zA-Z0-9- .:,()]", "") ' Strip unknown characters and store it in table "instance"
                      Dim sqlcomm As New MySqlCommand("insert into instance (instance_id, label, type_id) VALUES ('" & objectstr & "', '" & friendlyobjectstr & "', 1);", connection)
                      sqlcomm.ExecuteNonQuery()
                      sqlcomm = Nothing
```

```
                    End Try
                End If
                sqlsubj = Nothing
                sqlobj = Nothing
            End While

            'Check if there is already a record in table relation
            Dim sqlcommrelsearch As New MySqlCommand("select * from relation where src_id = " & subjectid & " and dest_id = " & objectid & " and type_id = " & typeid & ";", connection)
            reader = sqlcommrelsearch.ExecuteReader()
            If reader.HasRows = True Then 'If record found
                relation = 1
            End If
            reader.Close()
            reader = Nothing

            'Populate Table "relation" if no record was found
            If relation = 0 Then
                Dim sqlcommrel As New MySqlCommand("insert into relation (src_id, dest_id, type_id) VALUES (" & subjectid & ", " & objectid & ", " & typeid & ");", connection)
                sqlcommrel.ExecuteNonQuery()
                sqlcommrel = Nothing
            End If
        End If
        recount = recount + 1 'Increase record counter
        UploadLbl.Text = "Uploading. Please wait... (" & recount & ") file lines readed"
        If (recount Mod 10000 = 0) Then
            UploadPB.Value = UploadPB.Value + 1 'Increase progressbar
        End If
        Application.DoEvents()
        If UploadPB.Value = UploadPB.Maximum Then
            UploadPB.Value = 0 'Reset Progressbar
        End If
    Loop
    connection.Close()
    MsgBox("File has been uploaded to database successfully.")
    FilePathTxt.Text = ""
    LibCB.Items.Clear() 'Clear Combobox
Catch ex As Exception
    connection.Close()
    MsgBox(ex.Message)
End Try
Else
    '++++
```

```
        Dim subjectstr, predicatestr, objectstr, friendlysubjectstr, friendlyobjectstr, friendlypredicatestr As String
        connection.ConnectionString = myconnstring
        connection.Open()
        Dim subjectid, predicateid, relation, strlen As Integer
        strlen = 1500
        Dim reader As MySqlDataReader 'Get Combobox selection lib id for relation table
        Dim FileLineString As String
        FileListString = New TextFieldParser(FilePathTxt.Text) 'Store file path to string
        Do Until FileListString.EndOfData 'Search file until end
          subjectid = 0
          predicateid = 0
          relation = 0
          FileLineString = FileListString.ReadLine() 'Read file line by line
          subjectstr = Split(Split(FileLineString, "<")(1), ">")(0) 'Store string between first < and >
          predicatestr = Split(Split(FileLineString, "<")(2), ">")(0) 'Store string between second < and >
          Try
            objectstr = Split(Split(FileLineString, "> """)(1), """")(0) 'Store string between third " and "
            friendlyobjectstr = objectstr
          Catch ex As Exception
            objectstr = Split(Split(FileLineString, "<")(3), ">")(0) 'Store string between third < and >
            friendlyobjectstr = objectstr.Substring((objectstr.LastIndexOf("/") + 1), (objectstr.Length - objectstr.LastIndexOf("/") - 1)) 'Store string between last / and the end
            friendlyobjectstr = Uri.UnescapeDataString(friendlyobjectstr) 'url encoding to ASCII
          End Try
          friendlysubjectstr = subjectstr.Substring((subjectstr.LastIndexOf("/") + 1), (subjectstr.Length - subjectstr.LastIndexOf("/") - 1)) 'Store string between last / and the end
          friendlypredicatestr = predicatestr.Substring((predicatestr.LastIndexOf("/") + 1), (predicatestr.Length - predicatestr.LastIndexOf("/") - 1)) 'Store string between last / and the end

          friendlysubjectstr = Uri.UnescapeDataString(friendlysubjectstr) 'url encoding to ASCII
          friendlysubjectstr = friendlysubjectstr.Replace("_", " ") 'replace underscore with blank
          friendlysubjectstr = friendlysubjectstr.Replace("\", "\\") 'delete \
          friendlysubjectstr = friendlysubjectstr.Replace("'", "\'") 'delete '
          friendlysubjectstr = friendlysubjectstr.Replace("′", "\′") 'delete ′
          friendlypredicatestr = Uri.UnescapeDataString(friendlypredicatestr) 'url encoding to ASCII
          friendlypredicatestr = friendlypredicatestr.Replace("_", " ") 'replace underscore with blank
          friendlypredicatestr = friendlypredicatestr.Replace("\", "\\") 'delete \
          friendlypredicatestr = friendlypredicatestr.Replace("'", "\'") 'delete '
          friendlypredicatestr = friendlypredicatestr.Replace("′", "\′") 'delete ′
          objectstr = objectstr.Replace("_", " ") 'replace underscore with blank
```

```
            objectstr = objectstr.Replace("\", "\\") 'delete \
            objectstr = objectstr.Replace("'", "\'") 'delete '
            objectstr = objectstr.Replace("′", "\′") 'delete ′
            friendlyobjectstr = friendlyobjectstr.Replace("_", " ") 'replace underscore with blank
            friendlyobjectstr = friendlyobjectstr.Replace("\", "\\") 'delete \
            friendlyobjectstr = friendlyobjectstr.Replace("'", "\'") 'delete '
            friendlyobjectstr = friendlyobjectstr.Replace("′", "\′") 'delete ′

            If strlen < subjectstr.Length Then ' Alter navchar max depending on subjectstr length
               strlen = subjectstr.Length
               Dim sqluplen As New MySqlCommand("ALTER TABLE `wikipldb`.`instance` CHANGE COLUMN `instance_id` `instance_id` VARCHAR(" & strlen & ") CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL  , CHANGE COLUMN `label` `label` VARCHAR(" & strlen & ") CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL;", connection)
               sqluplen.ExecuteNonQuery()
               sqluplen = Nothing
            End If
            If strlen < objectstr.Length Then ' Alter navchar max depending on objectstr length
               strlen = objectstr.Length
               Dim sqluplen As New MySqlCommand("ALTER TABLE `wikipldb`.`instance` CHANGE COLUMN `instance_id` `instance_id` VARCHAR(" & strlen & ") CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL  , CHANGE COLUMN `label` `label` VARCHAR(" & strlen & ") CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL;", connection)
               sqluplen.ExecuteNonQuery()
               sqluplen = Nothing
            End If

            While (subjectid = 0 Or predicateid = 0)
               ' Search if subject already in table "instance" and store subject id

               Dim sqlsubj As New MySqlCommand("select id from instance where instance_id = '" & subjectstr & "';", connection)
               reader = sqlsubj.ExecuteReader()
               While reader.Read()
                  subjectid = reader("id")
               End While
               reader.Close()

               ' Search if object already in table "instance" and store object id
               Dim sqpred As New MySqlCommand("select id from instance where instance_id = '" & predicatestr & "';", connection)
               reader = sqpred.ExecuteReader()
               While reader.Read()
                  predicateid = reader("id")
               End While
               reader.Close()
```

```
            ' If subject not found in table "instance", store it
            If subjectid = 0 Then
               Try
                  Dim sqlcomm As New MySqlCommand("insert into instance (instance_id, label, type_id) VALUES ('" & subjectstr & "', '" & friendlysubjectstr & "', 1);", connection)
                  sqlcomm.ExecuteNonQuery()
                  sqlcomm = Nothing
               Catch ex As Exception ' Unknown characters found
                  friendlysubjectstr = Regex.Replace(friendlysubjectstr, "[^a-zA-Z0-9- .:,()]", "") ' Strip unknown characters and store it in table "instance"
                  Dim sqlcomm As New MySqlCommand("insert into instance (instance_id, label, type_id) VALUES ('" & subjectstr & "', '" & friendlysubjectstr & "', 1);", connection)
                  sqlcomm.ExecuteNonQuery()
                  sqlcomm = Nothing
               End Try

            End If

            ' If object not found in table "instance", store it
            If predicateid = 0 Then
               Try
                  Dim sqlcomm As New MySqlCommand("insert into instance (instance_id, label, type_id) VALUES ('" & predicatestr & "', '" & friendlypredicatestr & "', 3);", connection)
                  sqlcomm.ExecuteNonQuery()
                  sqlcomm = Nothing
               Catch ex As Exception ' Unknown characters found
                  friendlyobjectstr = Regex.Replace(friendlyobjectstr, "[^a-zA-Z0-9- .:,()]", "") ' Strip unknown characters and store it in table "instance"
                  Dim sqlcomm As New MySqlCommand("insert into instance (instance_id, label, type_id) VALUES ('" & predicatestr & "', '" & friendlypredicatestr & "', 3);", connection)
                  sqlcomm.ExecuteNonQuery()
                  sqlcomm = Nothing
               End Try
            End If
            sqlsubj = Nothing
            sqpred = Nothing
         End While

         'Check if there is already a record in table relation
         Dim sqlcommrelsearch As New MySqlCommand("select * from infoboxrelation where sub_id = " & subjectid & " and pred_id = " & predicateid & " and obj_id = '" & objectstr & "';", connection)
         reader = sqlcommrelsearch.ExecuteReader()
         If reader.HasRows = True Then 'If record found
            relation = 1
         End If
         reader.Close()
```

```
            reader = Nothing

            'Populate Table "relation" if no record was found
            If relation = 0 Then
               Dim sqlcommrel As New MySqlCommand("insert into infoboxrelation (sub_id, pred_id, obj_id, label) VALUES (" & subjectid & ", " & predicateid & ", '" & objectstr & "', '" & friendlyobjectstr & "');", connection)
               sqlcommrel.ExecuteNonQuery()
               sqlcommrel = Nothing
            End If

            recount = recount + 1 'Increase record counter
            UploadLbl.Text = "Uploading. Please wait... (" & recount & ") file lines readed"
            If (recount Mod 10000 = 0) Then
               UploadPB.Value = UploadPB.Value + 1 'Increase progressbar
            End If
            Application.DoEvents()
            If UploadPB.Value = UploadPB.Maximum Then
               UploadPB.Value = 0 'Reset Progressbar
            End If
         Loop
         connection.Close()
         MsgBox("File has been uploaded to database successfully.")
         FilePathTxt.Text = ""
         LibCB.Items.Clear() 'Clear Combobox
      End If
      LibCB.Enabled = True
      LibBtn.Enabled = True
      BrowseBtn.Enabled = True
      FilePathTxt.Enabled = True
      UploadBtn.Enabled = True
      DSTxt.Enabled = True
      DBTxt.Enabled = True
      UsernameTxt.Enabled = True
      PasswordTxt.Enabled = True
      UploadLbl.Visible = False
      UploadPB.Visible = False
      InfoboxCB.Checked = False
      UploadPB.Value = 0
   End Sub

   Private Sub ExitBtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ExitBtn.Click
      If connection.State = ConnectionState.Open Then
         connection.Close()
      End If
```

```
    Me.Close()
  End Sub

  Private Sub LibBtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LibBtn.Click
    LibCB.Items.Clear() 'Clear Combobox
    Dim myconnstring As String
    myconnstring = "Database=" & DBTxt.Text & ";Data Source=" & DSTxt.Text & ";User Id=" & UsernameTxt.Text & ";Password=" & PasswordTxt.Text & ";Charset=utf8;Default Command Timeout=180" 'SQL connection string
    Try
      connection.ConnectionString = myconnstring
      connection.Open()
      Dim reader As MySqlDataReader
      Dim sqlsellib As New MySqlCommand("select label from lib where type_id = 2;", connection)
      reader = sqlsellib.ExecuteReader()
      If (reader.HasRows) Then 'If table lib is filled populate combobox
        While reader.Read()
          LibCB.Items.Add(reader("label")) 'populate combobox
        End While
        LibCB.SelectedIndex = 0
        reader.Close()
      Else 'If table lib is empty ask if user whant to add the default values
        reader.Close()
        If MessageBox.Show("No records found. Do you want to add the default values?", "Lib Table", MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.Yes Then
          'Populate Table "lib"
          Dim sqlfilllib As New MySqlCommand("insert into lib (id, label, type_id) VALUES " & _
                                             "(1,'Article',1)," & _
                                             "(2, 'Disambiguation', 1)," & _
                                             "(3, 'Category', 1)," & _
                                             "(4, 'Wordnet Category', 1)," & _
                                             "(5, 'disambig_by', 2)," & _
                                             "(6, 'in_category', 2)," & _
                                             "(7, 'in_wordnet_category', 2)," & _
                                             "(8, 'reference', 1)," & _
                                             "(9, 'refers_to', 2);", connection)
          sqlfilllib.ExecuteNonQuery()
          sqlfilllib = Nothing

          Dim sqlsellib2 As New MySqlCommand("select label from lib where type_id = 2;", connection)
          reader = sqlsellib2.ExecuteReader()
          While reader.Read()
            LibCB.Items.Add(reader("label")) 'populate combobox
```

```
            End While
            LibCB.SelectedIndex = 0
            reader.Close()
          End If
        End If
        connection.Close()
      Catch ex As Exception
        connection.Close()
        MsgBox(ex.Message)
      End Try
    End Sub

    Private Sub InfoboxCB_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles InfoboxCB.CheckedChanged
      If InfoboxCB.Checked = True Then
        LibCB.Enabled = False
        LibBtn.Enabled = False
      Else
        LibCB.Enabled = True
        LibBtn.Enabled = True
      End If

    End Sub
End Class
```

Πρόγραμμα αλληλεπίδρασης με τον χρήστη

Η κεντρική σελίδα του προγράμματος (index.html) περιέχει το έτοιμο εργαλείο autosuggest με ετικέτα ‘Entrypoint’ από το οποίο ξεκινάει ο χρήστης. Το εργαλείο αυτό επικοινωνεί με την βάση μέσω του αρχείου suggest.php. Το suggest.php επιστρέφει ένα αρχείο xml και γεμίζει την λίστα του autosuggest δυναμικά καθώς πληκτρολογεί ο χρήστης. Όταν ο χρήστης επιλέξει κάτι από την λίστα, τότε γίνεται κλήση στο αρχείο showinfo.js. Το showinfo.js είναι ένα javascript αρχείο που με την σειρά κάνει κλήση ajax στο αρχείο getinfo.php και γεμίζει δυναμικά την κεντρική σελίδα (index.html). Το αρχείο getinfo.php είναι το αρχείο που μιλάει με την βάση και επιστρέφει τα δεδομένα. Εκτός από τις πληροφορίες που επιστρέφει το αρχείο, η σελίδα γεμίζει δυναμικά και με τις λέξεις-κλειδιά που έχει χρησιμοποιήσει ο χρήστης για την αναζήτηση του ενώ στην συνέχεια οι λέξεις προστίθενται στην μηχανή αναζήτησης της Google. Όλα τα style sheets της εφαρμογής βρίσκονται σε ξεχωριστό αρχείο (suggest.css). Στην ενότητα αυτή δεν παρουσιάζουμε όλο τον κώδικα αλλά σημαντικά κομμάτια του.

Ας δούμε ξεχωριστά τον κώδικά σε κάθε αρχείο:

- Ξεκινώντας έχουμε το αρχείο index.html που είναι η κεντρική σελίδα και περιέχει και το εργαλείο autosuggest. Χρησιμοποιεί τον έτοιμο κώδικα που βρίσκεται στην σελίδα autosuggest.js και κάνει κλήση στην σελίδα suggest.php για να γεμίσει την λίστα του ενώ όταν επιλέξει κάτι ο χρήστης, καλεί την showinfo.js με παραμέτρους τον κωδικό και το λεκτικό της επιλογής του χρήστη.

```
<html>
    <head>
        <title>Context-based search powered by WikipediA</title>
        <link rel="stylesheet" type="text/css" href="suggest.css"></link>
```

```
		<script type="text/javascript" src="showinfo.js"></script>
		<script type="text/javascript" src="autosuggest.js"></script>
		<script type="text/javascript" src="http://www.google.com/jsapi"></script>
		<LINK REL="SHORTCUT ICON" HREF="./images/gwiki.ico" />
	</head>
	<meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <body>
		<div id="fullpage">
			<div id="header">
				<img src="./images/googleWP.jpg" width="200" height="96" />
			</div>
			<div id="wikipl">
				<div id="stat">
					<div id="search">
						<b>Entrypoint:</b>
						<input type="text" id="tb" value=""/>
						 <img src="./images/loading.gif"
id="loading"/> 
					</div>
					<div id="showPLText"></br>
						<a id="displayText" href="#"
onmouseover="javascript:toggle('');return false;" onmouseout="javascript:toggle('');return false;"
onclick="return false;"><img src="./images/pl.jpg" id="pl"/></a>
					</div>
					<div id="pagelinkshow"
onmouseover="javascript:toggle('pl');return false;" onmouseout="javascript:toggle('pl');return
false;">
						<div id="pagelinkinside">
						<div id="pagelinksSearch"></div>
						<div id="pagelinks"></div>
						</div>
					</div>
				</div>
				</br><hr class="wikiplhr"/>
				<div id="dyna">
					<img src="./images/loading2.gif" id="loading2"/>
					<div id="name"></div></br>
					<div id="info"></div>
					<div id="infobox"></div></br>
				</div>
			</div>
			<div id="google">
				<div id="searchlist">
					<b>Your Filters</b>
					<ul id="navlist">No filters selected
					</ul>
```

```
                        </div></br>
                        <div id="googlepl">Loading...
                        </div>
                </div>
                <div id="footer">
                 <hr />
                        Google and the Google logo are trademarks of Google, Inc., Wikipedia and the Wikipedia logo are trademarks of Wikipedia Foundation, Inc.
                </div>
        </div>
        <script>
                new autosuggest("tb", "", "suggest.php?tb=", function(index, control) {showInfo(control.values[index], control.keywords[index])});
                google.load("search", "1", {"language" : "en"});
        </script>
    </body>
</html>
```

- Ακολουθεί ο κώδικας του αρχείου suggest.php που επιστρέφει ένα xml αρχείο για να γεμίσει το autosuggest καθώς πληκτρολογεί ο χρήστης. Επικοινωνεί με τον πίνακα instance και επιστρέφει το id και το label. Το xml που επιστρέφει έχει την μορφή:
  xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"yes\" <listdata>id1,label1|id2,label2|id3,label3|id4,label4</listdata>

```
<?php
    $tb = $_GET['tb'];
    $con = mysql_connect("localhost","gapost",""); //conect to mysql
    mysql_query('set character set UTF8',$con); //set UTF-8
    mysql_query("SET NAMES 'UTF8'",$con);
    if (!$con)
    {
        die("Could not connect: " . mysql_error()); //show error if connection fails
    }
    mysql_select_db("wikipldb",$con); //select wikipldb DB
    $result = mysql_query("select id, label from instance where label like '".$tb."%' and type_id = 1 LIMIT 0, 30",$con); //execute query
    header('Content-Type: application/xml;');
    $xml = "<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"yes\" ?>\n";
    $xml .= "<listdata>";
    while($row=mysql_fetch_array($result))
    {
        $xml .= $row['label'].",%,".$row['id']."|&|"; //Store data to xml variable followed by |
    }
    $xml = substr($xml, 0, -3); //strip last | from details
    $xml .= "</listdata>";
    echo $xml; //return xml variable
    mysql_close($con);
```

?>

- Κώδικας του αρχείου getinfo.php που επιστρέφει τις πληροφορίες της επιλογής του χρήστη δηλαδή τα pagelinks, categories, disambiguations, wordnet categories και infobox. Οι πληροφορίες αντλούνται από τους πίνακες instance, relation και infoboxrelation. Το αρχείο καλείται από μια συνάρτηση javascript που βρίσκεται στο showinfo.js και κάνει κλήση ajax προς το αρχείο.

```
<?php
$val=$_GET['val']; //get id
$categ = ""; //keeps category link
$categstrip = ""; //keeps string after Category:
$disamb = ""; // keeps disambiguation link
$wordnetstr = ""; //keeps string between - in wordnet
$wordnet = ""; //keeps wordnet link
$customarray = ""; //keeps pagelinks links divided by specific charachters
$infobox = ""; //keeps infobox table
$infolabel = ""; //keeps infobox label for multiple values with same label
$foundinfobox = ""; //keeps infobox found or not value
        $con = mysql_connect("localhost","gapost",""); //conect to mysql
        mysql_query('set character set UTF8',$con); //set UTF-8
        mysql_query("SET NAMES 'UTF8'",$con);
        if (!$con)
        {
                die("Could not connect: " . mysql_error()); //show error if connection fails
        }
        mysql_select_db("wikipldb",$con); //select wikipldb DB
        $result = mysql_query("SELECT instance.id, instance.label, relation.type_id FROM instance JOIN relation on instance.id=relation.dest_id WHERE relation.src_id = ".$val." order by instance.label",$con); //execute query
        while($row=mysql_fetch_array($result))
        {
                if ($row['type_id']==9) //find PageLinks
                {
                        $customarray .= "<a href='#' id='".$row['id']."' name=\"".$row['label']."\" onclick='showInfo(this.id, this.name);return false;'>".$row['label']."</a>##$";
                }
                elseif($row['type_id']==6) //find Categories
                {
                        $categstrip = substr(strrchr($row['label'], ":"), 1); //strip word Category: before categories
                        $categ .= "<a href='#' name=\"".$categstrip."\" onclick='Googlesearch(this.name);return false;'>".$categstrip."</a>, ";
                }
                elseif($row['type_id']==5) //find Disambiguations
                {
                        $disamb .= "<a href='#' name=\"".$row['label']."\" onclick='Googlesearch(this.name);return false;'>".$row['label']."</a>, ";
```

```
                }
                elseif($row['type_id']==7) //find Wordnet categories
                {
                        $wordnetstr = explode("-", $row['label']); //returns array of strings splitted by -
                        $wordnet .= "<a href='#' name=\"".$wordnetstr[1]."\" onclick='Googlesearch(this.name);return false;'>".$wordnetstr[1]."</a>, ";
                }
        }
        $result = mysql_query("SELECT instance.label, infoboxrelation.label as value FROM instance JOIN infoboxrelation on instance.id=infoboxrelation.pred_id WHERE infoboxrelation.sub_id = ".$val,$con); //execute query
        $infobox .= "<table id='Infobox'>";
        while($row=mysql_fetch_array($result))
        {
                $foundinfobox = 1;
                if (($row['label']!='name') && (substr($row['value'],0,4) != 'http') && ($row['value']!= NULL)) //Do not add url values
                {
                        if ($infolabel == $row['label']) //if previous label = current label
                        {
                                $infobox = substr($infobox, 0, -10); //remove </td></tr>
                                $infobox .= ", <a href='#' name=\"".$row['value']."\" onclick='Googlesearch(this.name);return false;'>".$row['value']."</a></td></tr>"; //add comma and current value and close row again
                        }
                        else//$row['label']
                        {
                                $arr = preg_replace('/(?!^)[[:upper:]]/',' \0',$row['label']); //Split CamelCase string
                                $infobox .= "<tr valign='top'><td class='Infob1sttd'><a href='#' name=\"".$arr."\" onclick='Googlesearch(this.name);return false;'>".$arr."</a></td><td><a href='#' name=\"".$row['value']."\" onclick='Googlesearch(this.name);return false;'>".$row['value']."</a></td></tr>"; //add row to the table
                        }
                        $infolabel = $row['label']; //store previous label
                }
        }
        mysql_close($con);
        $infobox .= "</table>"; //close infobox table
        $customarray = substr($customarray, 0, -3); //remove last 3 chars ##$ the separator for entring string to array
        $categ = substr($categ, 0, -2); //strip last space and comma from categories
        $disamb = substr($disamb, 0, -2); //strip last space and comma from disambiguation
        $wordnet = substr($wordnet, 0, -2); //strip last space and comma from disambiguation
        if (($disamb == "") && ($categ == "") && ($wordnet == ""))
        {
```

```
		echo "<center><b>No data found</b></center></br>";
	}
	if ($disamb != "")
	{
		echo "<b>Disambiguations</b></br>";
		echo $disamb."</br></br>"; //send categories
	}
	if ($categ != "")
	{
		echo "<b>Categories</b></br>";
		echo $categ."</br></br>"; //send categories
	}
	if ($wordnet != "")
	{
		echo "<b>Wordnet Categories</b></br>";
		echo $wordnet."</br></br>"; //send categories
	}
	echo "$$13$$"; //response seperator
	echo $customarray;
	echo "$$13$$"; //response seperator
	if ($foundinfobox != "")
	{
		echo "<b>InfoBox</b></br>";
		echo $infobox;
	}
?>
```

- Κώδικας σε γλώσσα javascript του αρχείου showinfo.js που κάνει κλήση ajax δυναμικά στο προηγούμενο αρχείο (getinfo.php) και γεμίζει την σελίδα. Είναι ο κώδικας που καλεί το getinfo.php παιρνόντας τον κωδικό (id) της επιλογής του χρήστης και παίρνει απάντηση τα δεδομένα εμφανίζοντας τα σε συγκεκριμένα σημεία της σελίδας.

```
function showInfo(val, txt)
{
	searchterms = ''; //initialize searchterms
	document.getElementById("pagelinks").scrollTop = 0; //Go on top of div
	document.getElementById("pagelinkshow").style.display = "none";
	document.getElementById("showPLText").style.display = "none";
	searchtermsarray = [];
	firstrun = ''; //change first run
	Googlesearch(txt); //Call Googlesearch
	var xmlhttp;
	if (window.XMLHttpRequest)
	{// code for IE7+, Firefox, Chrome, Opera, Safari
		xmlhttp=new XMLHttpRequest(); //assign an XMLHttpRequest object to variable
xmlhhtp
	}
```

```
	else
	{// code for IE6, IE5 old version that use uses an ActiveX Object
		xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
	}
	xmlhttp.onreadystatechange=function()
	{
		if (xmlhttp.readyState==4 && xmlhttp.status==200) //When readyState is 4 and status is 200, the response is ready
		{
			document.getElementById("loading2").style.display = "none"; //Hide loading image
			var strArray = xmlhttp.responseText.split("$$13$$"); //seperate responses
			//document.getElementById("name").innerHTML = txt; //Fill name of selection
			document.getElementById("name").innerHTML = "<a href='#' name=\""+txt+"\" onclick='Googlesearch(this.name);return false;'>"+txt+"</a>"
			document.getElementById("info").innerHTML=strArray[0]; //Fill details
			var jsvar = strArray[1].split("##$");//Fill aray with pagelinks
			document.getElementById("pagelinksSearch").style.display='inline'; //Show Pagelinks Text
			document.getElementById("pagelinks").innerHTML=jsvar.join(', '); //show array on pagelinks id
			if (document.getElementById("pagelinks").innerHTML != '')
			{
				document.getElementById("showPLText").style.display='block'; //If no pagelinks found hide pagelink text
			}
			document.getElementById("infobox").innerHTML=strArray[2];//Fill infobox
			if (!document.getElementById("inputsear")) //if textbox not exists create it
			{
				//Code to check if browser supports HTML5 - Start
				var test_canvas = document.createElement("canvas") //try and create sample canvas element
				var canvascheck=(test_canvas.getContext)? true : false //check if object supports getContext() method, a method of the canvas element
				//Code to check if browser supports HTML5 - End
				var element = document.createElement("input"); //create input
				if (canvascheck==true)
				{
					element.type = "search"; //input type = search if browser supports HTML5
				}
				else
				{
```

```
                                        element.type = "text"; //input type = text if browser does not supports HTML5
                                }
                                element.id = "inputsear"; //input id = inputsear
                                element.name = "inputsear"; //input name = inputsear
                                element.placeholder = "Search";
                                element.style.width = "280px"; //input width = 280px
                                element.style.outline = "none"; //input outline = none
                                element.style.border = "none"; //input border = none
                                var foo = document.getElementById("pagelinksSearch");
                                foo.appendChild(element); //Show new input at the end of pagelinksSearch
                        }
                        document.getElementById("inputsear").onkeyup = function () {findstr( document.getElementById("inputsear").value, jsvar)}; //On key release call function
                        document.getElementById("inputsear").onsearch = function () {findstr( document.getElementById("inputsear").value, jsvar)}; //On key release call function
                        document.getElementById("inputsear").value = ''; //Empty input inputsear
                        document.getElementById("tb").value = ''; //Empty autosuggest textbox
                }
        }
        document.getElementById("name").innerHTML = ''; //Clear name of selection
        document.getElementById("info").innerHTML = ''; //Clear details

        document.getElementById("pagelinksSearch").style.display = "none"; //Hide Pagelinks Text
        document.getElementById("pagelinks").innerHTML = ''; //Clear array of pagelinks
        document.getElementById("infobox").innerHTML = '';//Clear infobox
        document.getElementById("loading2").style.display = "block"; //Show loading image
        xmlhttp.open("GET","getinfo.php?val="+val,true); //Send a Request To Server passing 1 value
        xmlhttp.send();
}
```

- Κώδικας σε γλώσσα javascript που φτιάχνει λίστα με τις λέξεις κλειδιά και τις προσθέτει στην λίστα που βρίσκεται στην σελίδα. Ο κώδικας φτιάχνει μια λίστα από τις λέξεις-κλειδιά του χρήστη, και εξετάζει αν βρίσκονται στην λίστα με τις stopwords ώστε να τις κόψει. Έτσι φτιάχνει μια νέα λίστα με τις λέξεις που δεν ανήκουν στις stopwords και τις παρουσιάζει στην οπτική λίστα της κεντρικής σελίδας.

```
function Searchstopwords(txt) {
        var newstr = txt.replace(/[^a-zA-Z 0-9]+/g,' '); //Replace non alphanumeric
        newstr = newstr.replace(/\s{2,}/g, ' '); //strip more than one continuous spaces
        if (newstr.charAt(0)==' ') //if first charachter of string is space
        {
                newstr = newstr.substr(1); //remove first charachter
        }
        if (newstr.charAt(newstr.length-1)==' ') //if last charachter of string is space
```

```
	{
		newstr = newstr.substring(0, newstr.length-1); //remove last charachter
	}
	var brokenstring=newstr.split(" "); //Split text into array
	//var newstring = ''; //string that stores the final string after stopwords search
	var found = 0; //variable that changes to 1 if word is found on stopwords
	for ( i=0; i < brokenstring.length; i++ ) //search new array
	{
		found = 0;
		for ( j=0; j < lines.length; j++ ) //search array of stopwords
		{
			if (brokenstring[i].toLowerCase() == lines[j].toLowerCase()) //if found in stopwords
			{
				found = 1;
			}
		}
		if (found==0) //if not found in stopwords
		{
			//newstring = newstring + brokenstring[i] + " "; //create new string with words not in stopwords seperated by space
			if (searchtermsarray.length<1) //if array is empty
			{
				searchtermsarray.push(brokenstring[i]); //add first element
			}
			for ( k=0; k < searchtermsarray.length; k++ ) //search array
			{
				if (searchtermsarray[k].toLowerCase() == brokenstring[i].toLowerCase()) //if word is found in array
				{
					found = 1;
				}
			}
				if (found==0)
				{
					searchtermsarray.push(brokenstring[i]); //if word not found in array add element
				}
		}
	}

	var list = document.getElementById("navlist"); //populate list
	while( list.hasChildNodes() )
	{
		list.removeChild( list.lastChild ); //clear list
	}
```

```
	for (i = 0; i < searchtermsarray.length; i++) //populate list from array elements
	{
		var li = document.createElement("li");
		li.innerHTML = "<a href='#' name='"+searchtermsarray[i]+"' onclick='removeword(this.name);return false;'>"+searchtermsarray[i]+" <img src='./images/delete.png' id='delete'/></a>";
		list.appendChild(li);
	}

	searchterms = searchtermsarray.join(' ');
}
```

- Κώδικας σε γλώσσα javascript που αφαιρεί μια λέξη-κλειδί από την λίστα και ανανεώνει την λίστα που βρίσκεται στην σελίδα. Είναι η συνάρτηση που καλείται όταν ο χρήστης αφαιρεί κάτι από την λίστα με τις λέξεις-κλειδιά που βρίσκεται στην σελίδα. Στην ουσία ανανεώνει την λίστα και την εμφανίζει στην σελίδα.

```
function removeword(txt)
{
	for ( k=0; k < searchtermsarray.length; k++ ) //search array
	{
		if (searchtermsarray[k] == txt)
		{
			searchtermsarray.splice(k,1); //remove selected element from array
		}
	}

	var list = document.getElementById("navlist"); //populate list
	while( list.hasChildNodes() )
	{
		list.removeChild( list.lastChild ); //clear list
	}

	for (i = 0; i < searchtermsarray.length; i++) //populate list from array elements
	{
		var li = document.createElement("li");
		li.innerHTML = "<a href='#' name='"+searchtermsarray[i]+"' onclick='removeword(this.name);return false;'>"+searchtermsarray[i]+" <img src='./images/delete.png' id='delete'/></a>";
		list.appendChild(li);
	}

	if (!list.hasChildNodes()) //if list is empty than show No filters selected
	{
		var li = document.createElement("li");
		li.innerHTML = "No filters selected";
		list.appendChild(li);
```

```
        }

        searchterms = searchtermsarray.join(' ');
        Googlesearch("removeftomlist"); //Call Googlesearch function passing "removefromlist"
string
}
```