ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Οπτικοποίηση Μεγάλων Γράφων Βασισμένη στην Χρήση Εξωτερικής Μνήμης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Της

Μαρίας Κ. Κρομμύδα

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα,

Η σελίδα αυτή είναι σκόπιμα λευκή.

Υπόδειγμα φύλλου τίτλου (μπροστινή σελίδα) του αντιτύπου που υποβάλλεται στις βιβλιοθήκες (διπλωματική εργασία)

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

Οπτικοποίηση Μεγάλων Γράφων Βασισμένη στην Χρήση Εξωτερικής Μνήμης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Μαρίας Κ. Κρομμύδα

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα,

(Υπογραφή)

.....

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

Περίληψη

Η χρήση και η εκμετάλλευση των διασυνδεδεμένων δεδομένων βρίσκεται προς το παρόν σχεδόν τελείως περιορισμένη στην κοινότητα του Σημασιολογικού Ιστού. Η ραγδαία αύξηση των διασυνδεδεμένων δεδομένων που βρίσκονται διαθέσιμα στο διαδίκτυο όμως, καθώς και η αποδεδειγμένη χρησιμότητα τους για τους απλούς χρήστες, αυτούς που δεν έχουν τις τεχνικές γνώσεις ώστε να μπορέσουν να καταλάβουν τις δομές που διέπουν τα δεδομένα, καθιστούν επιτακτική την ανάγκη για μία εφαρμογή που θα ξεπερνάει τις δυσκολίες μέσα από την οπτικοποίηση των δεδομένων. Η οπτικοποίηση των διασυνδεδεμένων δεδομένων πρέπει να γίνει με ένα κατανοητό και εύχρηστο τρόπο, που θα επιτρέπει στον απλό χρήστη να αντιληφθεί την δομή τους, να πλοηγηθεί σε αυτά και να τα κατανοήσει με τελικό στόχο να εντοπίσει την ζητούμενη πληροφορία.
Για να καλύψουμε αυτήν την ανάγκη, ξεκινήσαμε με είσοδο ένα σύνολο διασυνδεδεμένων δεδομένων, δομημένα με το RDF μοντέλο, τα επεξεργαστήκαμε κατάλληλα και δημιουργήσαμε μία web εφαρμογή που παρουσιάζει τα δεδομένα στον χρήστη με την μορφή ενός κατευθυνόμενου γράφου. Πλαισιώσαμε τον γράφο με μια σειρά από εργαλεία με στόχο να κάνουμε πιο εύκολη για τον χρήστη τόσο την εξερεύνηση των δεδομένων όσο και την κατανόηση της πληροφορίας που αυτά περιέχουν

Λέξεις Κλειδιά: <<RDF, οπτικοποίηση, γράφος, διασυνδεδεμένα δεδομένα, Σημασιολογικός Ιστός>>

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

The uptake and consumption of Linked Data is currently restricted almost entirely to the Semantic Web community. The dramatic increase of the amount of the semantic data available on the Web, as well as their proven usefulness for the non-tech savvy web users -who are unable to understand the structure behind the data- have resulted in a need for an application that would overcome these hurdles by visualizing of the data. Visualizing and interacting with Linked Data should be implemented with a coherent and legible manner that would allow the user to understand their structure, browse the data and retrieve new pieces of information.
In answer to this challenge, we processed a Linked Data dataset, in RDF format, and we created a web application that presents the data to the user as a directed graph. We also implemented a series of modules with the purpose of helping the user explore the data and grasp a better understanding of their meaning.

Keywords: <<RDF, visualization, graph, Linked Data, Semantic Web>>

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πίνακας περιεχομένων

1 Εισαγωγή 4

1.1 Οπτικοποίηση RDF δεδομένων 4

1.2 Αντικείμενο διπλωματικής 5

1.2.1 Συνεισφορά 5

1.3 Οργάνωση κειμένου 6

2 Σχετικές εργασίες 7

2.1 Οπτικοποίηση Διασυνδεδεμένων Δεδομένων 8

2.2 Οπτικοποίηση Οντολογιών 9

3 Θεωρητικό υπόβαθρο 10

3.1 RDF Δεδομένα 10

3.2 OWL 11

4 Σχεδίαση Συστήματος 13

4.1 Αρχιτεκτονική 13

4.2 Περιγραφή Λειτουργιών 14

4.2.1 Επεξεργασία Δεδομένων Εισόδου 14

4.2.2 Υπηρεσίες Server 16

4.2.3 Υπηρεσίες Client 16

4.3 Περιγραφή Κλάσεων 18

4.3.1 RDFToDot 18

4.3.2 SplitGraphvizInput 18

4.3.3 MergeGraphvizOutput 18

4.3.4 CreateImportFileForDB 19

4.3.5 GraphVisualisation 19

4.3.6 SearchInGraph 19

4.3.7 NodeNeighborsPanel 19

4.4 Βάση Δεδομένων 20

5 Υλοποίηση 22

5.1 Λεπτομέρειες υλοποίησης 22
5.1.1 Ονόματα Κόμβων *22*
5.1.2 Μεταβλητές Graphviz *23*
5.1.3 Δημιουργία γράφου *23*
5.1.4 Διαγραφή γράφου *24*
5.1.5 Zoom *24*
5.1.6 Panning *24*
5.1.7 Μετακίνηση σε επιλεγμένο κόμβο *25*
5.2 Πλατφόρμες και προγραμματιστικά εργαλεία 25
5.2.1 Apache Jena *26*
5.2.2 Graphviz Software *26*
5.2.3 MxGraph *27*
5.2.4 JQuery *27*
6 Έλεγχος 28
6.1 Έλεγχος χρόνου ανταπόκρισης της εφαρμογής 28
6.2 Παρουσίαση ελέγχου για χρόνο εμφάνισης γράφου 30
6.2.1 Yago facts dataset *30*
6.2.2 Dbpedia dataset *35*
6.3 Παρουσίαση ελέγχου για αναζήτηση με λέξη-κλειδί 39
6.3.1 Yago facts dataset *39*
6.3.2 Dbpedia dataset *40*
6.4 Παρουσίαση ελέγχου για αναζήτηση γειτονικών κόμβων 40
7 Επίλογος 41
7.1 Σύνοψη και συμπεράσματα 41
8 Βιβλιογραφία 42

Πίνακας 1: Τυχαία παράθυρα, yago facts dataset 30
Πίνακας 2: 20% νέος καμβάς, yago facts dataset 30
Πίνακας 3: 50% νέος καμβάς, yago facts dataset 31
Πίνακας 4: 80% νέος καμβάς, yago facts dataset 32
Πίνακας 5: Τυχαία παράθυρα, dbpedia dataset 35

Πίνακας 6: 20% νέος καμβάς, dbpedia dataset 35
Πίνακας 7: 50% νέος καμβάς, dbpedia dataset 36
Πίνακας 8: 80% νέος καμβάς, dbpedia dataset 36

Εικόνα 1: Τα υποσυστήματα και πως αυτά επικοινωνούν μεταξύ τους.....14
Εικόνα 2: Οι κλάσεις που κατασκευάζουν το αρχείο για την βάση δεδομένων.....15
Εικόνα 3: Η jsp σελίδα καλεί την κατάλληλη Java κλάση ανάλογα με την ενέργεια του χρήστη 16
Εικόνα 4: Ενέργειες του χρήστη που στέλνουν ένα Ajax request.....17

1

Εισαγωγή

1.1 Οπτικοποίηση RDF δεδομένων

Τα τελευταία χρόνια, ο όγκος των διασυνδεδεμένων δεδομένων (Linked Data[1]) που είναι διαθέσιμα στο διαδίκτυο αυξάνεται με γεωμετρικούς ρυθμούς, αφού όλο και περισσότεροι οργανισμοί και βάσεις δεδομένων παρέχουν τα δεδομένα τους σε αυτήν την μορφή με στόχο να προσφέρουν στους χρήστες μία ενιαία πηγή δεδομένων κατάλληλη για εκμετάλλευση. Παρόλο που τα πλεονεκτήματα των διασυνδεδεμένων δεδομένων είναι αδιαμφισβήτητα - κυρίως λόγο της δυνατότητας που προσφέρουν στην διασύνδεση πηγών δεδομένων με τελείως διαφορετικές δομές- ο τρόπος με τον οποίο παρουσιάζονται μέσα από το RDF (Resource Description Framework) μοντέλο οδηγεί στην χρήση τους αποκλειστικά και μόνο από την κοινότητα των επιστημών που είναι εκπαιδευμένοι είτε στα διασυνδεδεμένα δεδομένα είτε στον Σημασιολογικό Ιστό (Semantic Web[2]) ενώ τα καθιστά απρόσιτα για το ευρύ κοινό. Με τον όγκο τους να αυξάνεται διαρκώς, αυξάνεται ταυτόχρονα και η πολυπλοκότητα τους αλλά και η δυσκολία του να αντιληφθεί ο μέσος χρήστης το νόημα της πληροφορίας όπως αυτή παρουσιάζεται. Η διασύνδεση ενός μεγάλου αριθμού ανεξάρτητων πηγών δεδομένων, καθιστά αδύνατη τόσο την χρήση όσο και την εξερεύνηση τους από

[1] http://linkeddata.org/

[2] http://semanticweb.org/wiki/Main_Page

χρήστες που δεν έχουν την κατάλληλη γνώση και εμπειρία. Από τα πρώτα βήματα των διασυνδεδεμένων δεδομένων υπήρξε άμεση αναγνώριση αυτού του προβλήματος και για αυτόν τον λόγο έχουν γίνει μεγάλες προσπάθειες για να βρεθούν εργαλεία που όχι μόνο να οπτικοποιούν τα διασυνδεδεμένα δεδομένα αλλά και να βοηθούν στην αλληλεπίδραση του χρήστη με αυτά. Το ίδιο το RDF μοντέλο που δημιουργεί το πρόβλημα της κατανόησης μας εξασφαλίζει ότι ακολουθούνται μια σειρά από κανόνες και μια προκαθορισμένη δομή, στοιχεία που δίνουν στα διασυνδεδεμένα δεδομένα ένα τεράστιο πλεονέκτημα σε σύγκριση με όλες τις προηγούμενες στρατηγικές οπτικοποίησης πληροφορίας. Έχοντας μία συγκεκριμένη δομή είναι δυνατόν να χτιστεί πάνω σε αυτήν μια στρατηγική οπτικοποίησης απαλλαγμένη από προβλήματα ανομοιογένειας και ασυνέπειας των δεδομένων εισόδου.

1.2 Αντικείμενο διπλωματικής

Στόχος αυτής της διπλωματικής είναι η αναπαράσταση μεγάλων γράφων δεδομένων, που προέρχονται από κάποια πηγή δεδομένων σε RDF μορφή, χωρίς την χρήση περίπλοκών υποδομών και ισχυρών προγραμματιστικών εργαλείων, αλλά μέσα από απλές δομές. Ο όγκος των δεδομένων που κληθήκαμε να διαχειριστούμε ήταν τέτοιος που απαγόρευε την οποιαδήποτε σκέψη για καθολική προβολή τους και δημιούργησε μία βασική ανάγκη για την αποθήκευση της πληροφορίας. Επιλέχθηκε οι πληροφορίες για τον γράφο να αποθηκευτούν σε μία βάση δεδομένων με σκοπό να χρησιμοποιηθούν ορισμένα εργαλεία που προσφέρονται για την διευκόλυνση της εξερεύνησης των δεδομένων από τον χρήστη. Έχοντας πλέον την πληροφορία αποθηκευμένη υπήρχε η δυνατότητα να δημιουργήσουμε ένα ευρύ φάσμα λειτουργιών που περιλαμβάνουν την αναζήτηση με λέξεις-κλειδιά, την αναζήτηση των γειτόνων ενός κόμβου αλλά και το κυριότερο που αφορά την εμφάνιση στην οθόνη μόνο του τμήματος του γράφου τον οποίον επιθυμεί να δει ο χρήστης. Δημιουργήθηκε λοιπόν μία web εφαρμογή που εμφανίζει στους χρήστες τα διασυνδεδεμένα δεδομένα με την μορφή γράφου και καθιστά την περιήγηση σε αυτά γρήγορη και εύκολη.

1.2.1 Συνεισφορά

Η συνεισφορά της εργασίας είναι ένα σύστημα επεξεργασίας, αποθήκευσης και αναπαραγωγής δεδομένων που βρίσκονταν αρχικά σε RDF μορφή. Αναλυτικότερα, μελετήσαμε την μορφή των RDF μοντέλων για τα διασυνδεδεμένα δεδομένα και επιλέξαμε το κατάλληλο εργαλείο για να τα επεξεργαστούμε. Στην συνέχεια διαχειριστήκαμε τα δεδομένα ώστε να τα φέρουμε σε μία μορφή κατάλληλη για οπτικοποίηση. Έπειτα επιλέξαμε το καταλληλότερο εργαλείο που μας επέτρεψε να μετατρέψουμε έναν πολύ μεγάλο όγκο

δεδομένων σε ένα σαφή και κατανοητό γράφο. Η πληροφορία αποθηκεύτηκε σε μία βάση δεδομένων και στηριζόμενοι σε βασικές τεχνικές, κυρίως του indexing, μπορέσαμε να δημιουργήσουμε μια εφαρμογή που να παρουσιάζει κατανοητά και γρήγορα τον γράφο στον χρήστη, δίνοντας του ταυτόχρονα την δυνατότητα να εξερευνήσει τα δεδομένα με σαφήνεια.

1.3 Οργάνωση κειμένου

Η διπλωματική εργασία περιλαμβάνει στο Κεφάλαιο 2 σχετικές εργασίες αναφορικά με εργαλεία και τεχνικές που έχουν προταθεί για την οπτικοποίηση της πληροφορίας. Στο Κεφάλαιο 3 δίνεται το θεωρητικό υπόβαθρο, η αναλυση των δομών και των κανόνων που διέπουν το RDF μοντέλο αλλά και η αυστηρότερη μορφή του, το μοντέλο OWL (Web ontology Language) ώστε να γίνει κατανοητό πως το ίδιο το μοντέλο μας δίνει την δυνατότητα να μετατρέψουμε τα δεδομένα σε γράφο. Το Κεφάλαιο 4 αφιερώνεται στην αρχιτεκτονική του συστήματος, στον τρόπο με τον οποίο είναι δομημένα τα υποσυστήματα του, στο περιεχόμενο των κλάσεων που δημιουργήσαμε και στην δομή της βάσης δεδομένων που χρησιμοποιούμε. Στο Κεφάλαιο 5 παρουσιάζονται ορισμένα σημεία της εφαρμογής με ιδιαίτερο ενδιαφέρον καθώς και τα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής. Στο Κεφάλαιο 6 παρουσιάζονται αναλυτικά τα σενάρια ελέγχου του συστήματος, ο τρόπος με τον οποίο μετρήθηκε η απόδοση του και τέλος γίνεται μία αξιολόγηση των αποτελεσμάτων που προέκυψαν. Το Κεφάλαιο 7 περιλαμβάνει μια σύνοψη της συνεισφοράς της εργασίας ενώ το Κεφάλαιο 8 περιλαμβάνει την βιβλιογραφία.

2

Σχετικές εργασίες

Η εργασία μας επιχειρεί την οπτικοποίηση με την μορφή ενός κατευθυνόμενου γράφου διασυνδεδεμένων δεδομένων που βρίσκονται αποθηκευμένα με βάση το RDF μοντέλο. Θεματικές περιοχές οι οποίες περιέχουν σχετικές εργασίες είναι η οπτικοποίηση διασυνδεδεμένων δεδομένων που βρίσκονται είτε σε RDF μορφή είτε σε OWL μορφή.

Υπάρχουν πολλές προσεγγίσεις σε ότι αφορά τις τεχνικές οπτικοποίησης των δεδομένων, όπως χαρακτηριστικά βλέπουμε στο [4]. Σε αυτήν την εργασία γίνεται μία έρευνα ώστε να βρεθούν τα κύρια χαρακτηριστικά που πρέπει να έχει ένα σύστημα οπτικοποίησης διασυνδεδεμένων δεδομένων ώστε να μπορεί να χαρακτηριστεί επιτυχημένο και λειτουργικό. Οι προσπάθειες οπτικοποίησης χωρίζονται αρχικά σε δύο τύπους, στον πρώτο τύπο περιλαμβάνονται τα εργαλεία που παρουσιάζουν τα δεδομένα με μια απλοποιημένη μορφή κειμένου –στοχεύοντας σε μία λεπτομερή και στοχευμένη ανάλυση ενός πολύ μικρού τμήματος της πληροφορίας-, ενώ στον δεύτερο ανήκουν εργαλεία που στηρίζονται στην εικόνα για την παρουσίαση των δεδομένων –επιτρέποντας στον χρήστη να έχει μία ξεκάθαρη εικόνα της ευρύτερης δομής των δεδομένων. Ακόμα βασικός διαχωρισμός είναι και αν τα μοντέλα απευθύνονται σε έμπειρους ή απλούς χρήστες, αυτός ο διαχωρισμός καθορίζει το εάν η αναζήτηση πάνω στα δεδομένα θα γίνεται με απλό ή σύνθετο τρόπο και το εάν τα δεδομένα θα εμφανίζονται στην κανονική τους ή σε πιο απλοποιημένη μορφή. Σχεδόν κατά αποκλειστικότητα παρατηρείται ότι τα εργαλεία που εμφανίζουν τα δεδομένα σε μορφή κειμένου απευθύνονται μόνο σε καταρτισμένους χρήστες αφού απαιτούνται σύνθετες και πολύπλοκες αναζητήσεις μέχρι την εύρεση του τελικού αποτελέσματος.

Το [5] έρχεται να επισημάνει ότι η οπτικοποίηση οντολογιών είναι ουσιαστικά μία υποκατηγορία του προβλήματος της οπτικοποίησης δομημένων γράφων. Αυτό έχει ως αποτέλεσμα να δίνεται μεγάλη βαρύτητα στην εικόνα του γράφου –ένα προγραμματιστικό πρόβλημα που έχει απασχολήσει πολύ με αποτέλεσμα να υπάρχει μια μεγάλη συλλογή προγραμματιστικών εργαλείων και εφαρμογών ήδη διαθέσιμα καθώς και ένα σημαντικό θεωρητικό υπόβαθρο-και να παραμελείται η ανάγκη για εξερεύνηση και κατανόηση των δεδομένων, αφού τα περισσότερα εργαλεία παραμελούν την πραγματική ανάγκη που οδηγεί στην οπτικοποίηση των διασυνδεδεμένων δεδομένων που είναι η εύκολη εύρεση της ζητούμενης πληροφορίας. Τα περισσότερα εργαλεία δεν παρέχουν αποτελεσματικά εργαλεία αναζήτησης και δεν εκμεταλλεύονται την σημασιολογία της πληροφορίας, αδυνατώντας έτσι να βοηθήσουν τον –απλό κυρίως- χρήστη να περιηγηθεί εύκολα σε αυτά.

2.1 Οπτικοποίηση Διασυνδεδεμένων Δεδομένων

Έχουν γίνει σημαντικές προσπάθειες ώστε να βρεθεί ένας αποδοτικός και εύχρηστος τρόπος οπτικοποίησης των διασυνδεδεμένων δεδομένων και στην συνέχεια παρουσιάζονται κάποια από τα πιο αξιόλογα εργαλεία που έχουν προκύψει.

Στην εργασία [1] παρουσιάζεται ένα επίσημο μοντέλο οπτικοποίησης διασυνδεδεμένων δεδομένων το LDMV (Linked Data Visualization Model), το οποίο επιτρέπει την σύνδεση διαφορετικών συνόλων δεδομένων, το κάθε ένα από τα οποία μπορεί να έχει διαφορετική οπτικοποίηση, με έναν δυναμικό τρόπο. Για να πετύχει τέτοια ελαστικότητα και έναν υψηλό βαθμό αυτοματισμού το LDVM βασίζεται σε ένα διάγραμμα ροής της οπτικοποίησης που ενσωματώνει βήματα αναλυτικής εξαγωγής και οπτικής αφαίρεσης. Κάθε ένα από τα βήματα του διαγράμματος ροής περιλαμβάνει έναν αριθμό από μεταμορφωτικούς τελεστές, που μπορούν να οριστούν με έναν δηλωτικό τρόπο. Αυτό έχει ως αποτέλεσμα το LDVM να ισορροπεί ανάμεσα στην ευελιξία των τρόπων οπτικοποίησης και στην αποδοτικότητα της εκτέλεσης και σύνθεσης του τελικού αποτελέσματος. Για τον σκοπό αυτό επιλέγεται η εμφάνιση του γράφου στον χρήστη με δυναμικό τρόπο, αφού σε κάθε βήμα του διαγράμματος ροής ο γράφος συμπληρώνεται και αλλάζει μορφή.

Τέλος, πρέπει να αναφέρουμε ότι παρόλο που το εργαλείο χρησιμοποιεί γράφους για την παρουσίαση των τελικών αποτελεσμάτων απευθύνεται σε χρήστες με γνώσεις γύρω από τα διασυνδεδεμένα δεδομένα και τον Σημασιολογικό Ιστό, αφού κατά το στάδιο της εξαγωγής των τελικών δεδομένων από τα αρχικά χρησιμοποιούνται τελεστές και συναρτήσεις από την SPARQL.

Στην εργασία [2] η προσοχή εστιάζεται κυρίως στον απλό χρήστη του διαδικτύου το βασικό πρόβλημα του οποίου είναι ότι δεν έχει τις κατάλληλες γνώσεις ώστε να αξιοποιήσει τα

διασυνδεδεμένα δεδομένα αφού η περιήγηση σε αυτά με στόχο την αναζήτηση πληροφορίας απαιτεί γνώσεις γύρω από τον Σημασιολογικό Ιστό. Η λύση που προτείνεται για αυτό το πρόβλημα είναι η απόκρυψη της στοίβας των πληροφοριών από τον χρήστη (hide the stack). Το εργαλείο επιτρέπει στον τελικό χρήστη να δει τα δεδομένα, με την μορφή γράφου, να εξερεύνηση τις πληροφορίες που περιέχονται σε αυτά και να ανακάλυψη την γνώση που προσφέρει η χρήση των διασυνδεδεμένων δεδομένων. Επιλέχθηκε να παρουσιάζεται ο γράφος σε επίπεδα κάτι που όμως τελικά αποδείχτηκε ότι περισσότερο μπέρδευε τους απλούς χρήστες –γιατί ο γράφος άλλαζε μορφή-, παρά τους βοηθούσε –όπως είχε εκτιμηθεί αρχικά ότι θα συνέβαινε λόγω των λιγότερων κόμβων που προβάλλονταν σε κάθε επίπεδο.

Τέλος μπορούμε να πούμε ότι το Hide the Stack είναι ένα εργαλείο που συμβάλει στην εξοικείωση του απλού χρήστη με τα διασυνδεδεμένα δεδομένα, την εικόνα και την δομή τους, και θα τους βοηθήσει να βρουν την πληροφορία που αναζητούν, μέσα από την αναζήτηση με λέξεις κλειδιά αλλά έχει πολύ λίγα να προσφέρει σε έναν πιο πεπειραμένο χρήστη με κάποια εμπειρία στον Σημασιολογικό Ιστό.

2.2 Οπτικοποίηση Οντολογιών

Τα εργαλεία που έχουν προταθεί για την οπτικοποίηση των οντολογιών είναι πολύ λιγότερα σε σχέση με αυτά που βασίζονται στο RDF μοντέλο κάτι που οφείλεται κυρίως στην αυστηρότερη δομή που έχει το OWL μοντέλο περιορίζοντας κατά πολύ την ευελιξία κατά την οπτικοποίηση.

Στο [3] επισημαίνεται ότι η πλειονότητα των χρηστών δεν είναι ικανοποιημένοι με την υποστήριξη που παρέχεται με τα μέχρι τώρα εργαλεία για την οπτικοποίηση οντολογιών και κυρίως με την δυνατότητα που δίνεται για αναζήτηση και εξερεύνηση τους. Τα προβλήματα αυτά γίνονται ακόμα πιο έντονα για τους απλούς χρήστες αφού αυτοί είναι που κατά βάση βασίζονται σε αποδοτικά εργαλεία ώστε να μπορέσουν να εστιάσουν στις λεπτομέρειες που τους ενδιαφέρουν και να κατανοήσουν το περιεχόμενο και την δομή των οντολογιών. Ως λύση σε αυτό το πρόβλημα προτείνεται το KC-Viz, ένα εργαλείο το οποίο εκμεταλλεύεται μια εμπειρικά αποδεδειγμένη μέθοδο για την περίληψη των οντολογιών με στόχο να παρουσιάσει μικρότερα σχήματα οντολογιών αλλά και να δώσει μία «μέση λύση» στην εξερεύνηση των οντολογιών ξεκινώντας με αυτές που είναι πιο πλούσιες σε πληροφορία.

Το γεγονός ότι η μέθοδος που χρησιμοποιείται είναι εμπειρική και το ότι μεγάλος όγκος των πληροφοριών αποκρύπτεται από τον τελικό χρήστη, καθιστά το εργαλείο πολύ εύχρηστο για τους χρήστες που δεν έχουν ιδιαίτερες γνώσεις και απαιτήσεις ενώ από την άλλη περιορίζει τον πεπειραμένο χρήστη που θα ήθελε να δει την διαδρομή με την οποία έφτασε σε αυτόν αυτή η πληροφορία, ώστε να μπορέσει να αξιολογήσει την πηγή.

3

Θεωρητικό υπόβαθρο

Το μοντέλο RDF αποτελεί το βασικό μοντέλο για την αποθήκευση και απεικόνιση των δεδομένων του Σημασιολογικού Ιστού, και για αυτό επιλέχθηκε ως το μοντέλο που θα ακολουθούν και τα δεδομένα εισόδου της εφαρμογής μας.

3.1 RDF Δεδομένα

Με τον όρο RDF[3] (Resource Description Framework) αναφερόμαστε σε μια κατηγορία προδιαγραφών του W3C (World Wide Web Consortium)[4] που αρχικά σχεδιάστηκε ως μοντέλο μετα-δεδομένων. Στην συνέχεια όμως άλλαξε χαρακτήρα και πλέον χρησιμοποιείται σαν μία γενική μέθοδος για την εννοιολογική περιγραφή ή την μοντελοποίηση πληροφοριών που περιέχονται σε διαδικτυακές πηγές, χρησιμοποιώντας μια ποικιλία από συντακτικούς κανόνες και μορφές σειριοποίησης δεδομένων.

Το μοντέλων RDF για τα δεδομένα είναι βασισμένο στην ιδέα της παραγωγής προτάσεων (statements) από πηγές (resources) -κυρίως διαδικτυακές- με την μορφή υποκείμενο-κατηγορούμενο-αντικείμενο (subject-predicate-object). Οι εκφράσεις αυτές είναι γνωστές με τον όρο τριπλέτες (triplets). Το υποκείμενο υποδηλώνει την πηγή από την οποία έχουν προέλθει τα δεδομένα ενώ το κατηγορούμενο τονίζει πτυχές και γνωρίσματα της πηγής και

[3] http://www.w3.org/RDF/

[4] http://www.w3.org

εκφράζει μία σχέση μεταξύ υποκειμένου και αντικειμένου. Το υποκείμενο και κατηγορούμενο στο RDF μοντέλο είναι πάντα ένα URI (Uniform Resource Identifier) ενώ το αντικείμενο εκτός από URI μπορεί να είναι και μία συμβολοσειρά, παρέχοντας κάποιο σχόλιο ή περιγραφή για το υποκείμενο.

Ο μηχανισμός αυτός, που επιτρέπει την περιγραφή πηγών, είναι μια μεγάλη συνιστώσα στην δράση του W3C Semantic Web αφού αποτελεί μια επανάσταση για τον παγκόσμιο ιστοχώρο επιτρέποντας σε αυτοματοποιημένο λογισμικό να αποθηκεύσει, να ανταλλάξει και να χρησιμοποιήσει πληροφορίες σε μορφή αναγνώσιμη για μηχανές. Από τον τρόπο που το RDF μοντέλο είναι δομημένο, μπορούμε να πούμε ότι μια συλλογή από RDF τριπλέτες, ουσιαστικά αναπαριστά έναν κατευθυνόμενο γράφο. Ως εκ τούτου ένα μοντέλο δεδομένων που βασίζεται στους RDF κανόνες μπορεί πιο εύκολα να αναπαρασταθεί μέσα από μοντέλα από ότι τα σχεσιακά ή άλλα οντολογικά σχήματα.

3.2 OWL

Ο Σημασιολογικός Ιστός αποτελεί το όραμα του μέλλοντος του διαδικτύου αφού έχει ως στόχο να δώσει στη πληροφορία ρητό νόημα, καθιστώντας πιο εύκολο για τις μηχανές να επεξεργάζονται αυτόματα και να ενοποιούν πληροφορίες που είναι διαθέσιμες στο διαδίκτυο. Ο Σημασιολογικός Ιστός στηρίζεται στην ικανότητα της XML[5] (Extensible Markup Language) να ορίζει ένα δομημένο σύστημα ετικετών τροποιημένο με τέτοιο τρόπο ώστε να καλύπτει τις συγκεκριμένες ανάγκες των εκάστοτε εφαρμογών αλλά και στην ελαστική προσέγγιση του RDF μοντέλου στην αναπαράσταση δεδομένων. Το επόμενο στοιχείο που είναι απαραίτητο για τον Σημασιολογικό Ιστό είναι μία γλώσσα οντολογιών που θα μπορεί να περιγράφει με τυπικό τρόπο την σημασιολογία των κλάσεων και των ιδιοτήτων που χρησιμοποιούνται στα κείμενα του διαδικτύου. Για να μπορούν οι μηχανές να εκτελούν χρήσιμες συλλογιστικές εργασίες πάνω σε αυτά τα κείμενα, η γλώσσα πρέπει να ξεπερνάει την βασική σημασιολογία του RDF μοντέλου.

Το κενό αυτό έρχεται να καλύψει η OWL [6](Web ontology Language), η οποία δημιουργήθηκε για να χρησιμοποιείται αποκλειστικά και μόνο όταν την πληροφορία που περιέχεται σε αρχεία του διαδικτύου πρόκειται να την επεξεργαστούν μηχανές και όχι να παρουσιαστεί στον άνθρωπο. Η OWL μπορεί να χρησιμοποιηθεί για να παρουσιάσει με τυπικό τρόπο την σημασία των όρων σε λεξιλόγια και τις σχέσεις μεταξύ αυτών των όρων. Η OWL παρέχει περισσότερες υποδομές για την σημασιολογική έκφραση των οντολογιών από

[5] http://www.w3schools.com/xml/

[6] http://www.w3.org/TR/owl-features/

ότι η XML και το RDF μοντέλο και για αυτό έχει περισσότερες ικανότητες στο να παρουσιάζει κείμενα με περιεχόμενα κατανοητά απο μηχανές.

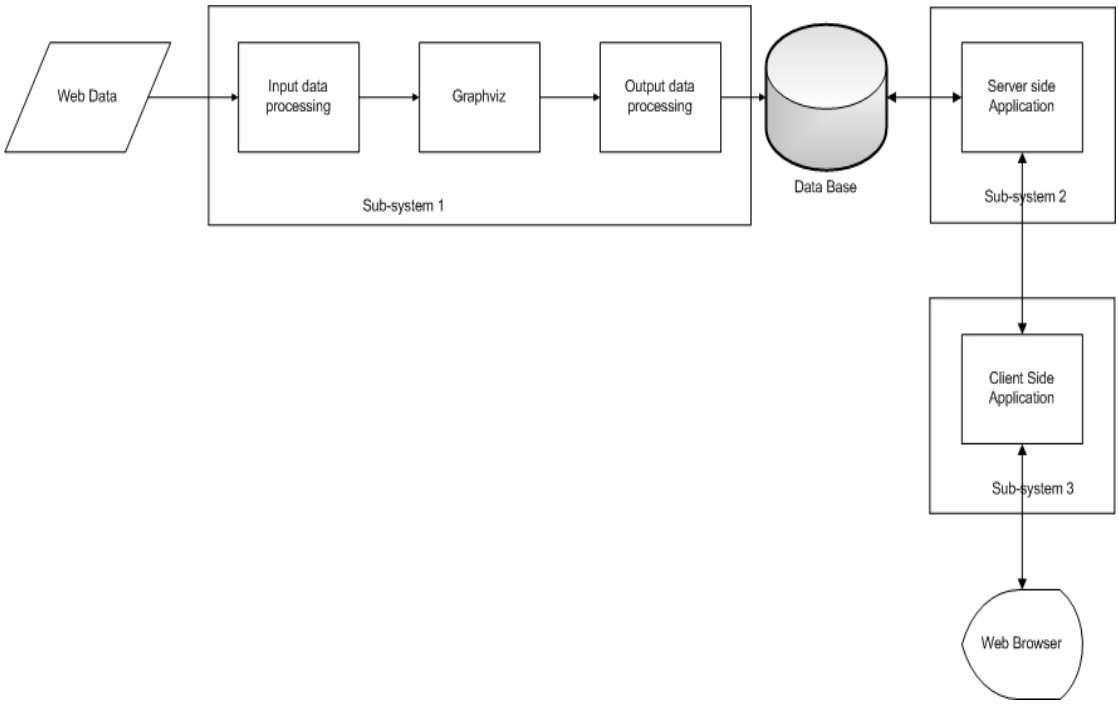
Η OWL έχει χτιστεί πάνω στην RDF, γράφεται με XML και έχει πολύ ισχυρούς συντακτικούς κανόνες. Έχει σχεδιαστεί αποκλειστικά για να την καταλαβαίνουν μηχανές και για να κάνει εύκολη την επεξεργασία των πληροφοριών που συλλέγονται από το διαδίκτυο ενώ είναι πολύ δύσκολο να διαβαστεί από τον άνθρωπο.

4

Σχεδίαση Συστήματος

4.1 Αρχιτεκτονική

Το σύστημα που έχουμε σχεδιάσει μπορεί να χωριστεί σε τρία υποσυστήματα. Το πρώτο υποσύστημα αφορά την επεξεργασία των δεδομένων εισόδου. Τα δεδομένα εισόδου πρέπει να ακολουθούν τους κανόνες του RDF μοντέλου και να βρίσκονται γραμμένα σε κάποια μορφή που να υποστηρίζεται από την βιβλιοθήκη Jena. Μετά την πρώτη επεξεργασία τους τα δεδομένα μετατρέπονται σε γράφο και στην συνέχεια αποθηκεύονται σε μία βάση δεδομένων. Το δεύτερο υποσύστημα αφορά τις υπηρεσίες server, δέχεται τα ερωτήματα από τον χρήστη – είτε άμεσα μέσα από την αναζήτηση είτε έμμεσα μέσα από την μετακίνηση σε κάποιο άλλο σημείο του γράφου-, υποβάλει τα κατάλληλα SQL Queries στην βάση δεδομένων και επιστρέφει το αποτέλεσμα στον χρήστη κατάλληλα διαμορφωμένο ώστε πριν την εμφάνιση του στον browser να μην μεσολαβεί τίποτα περισσότερο από ένα evaluate. Τέλος, το τρίτο υποσύστημα αφορά το τμήμα της εφαρμογής που απευθύνεται στον τελικό χρήστη, ο τρόπος με τον οποίο ο γράφος εμφανίζεται στον browser καθώς και κάποια εργαλεία σχεδιασμένα με τέτοιο τρόπο ώστε να κάνουν ευκολότερη την πλοήγηση στα δεδομένα.

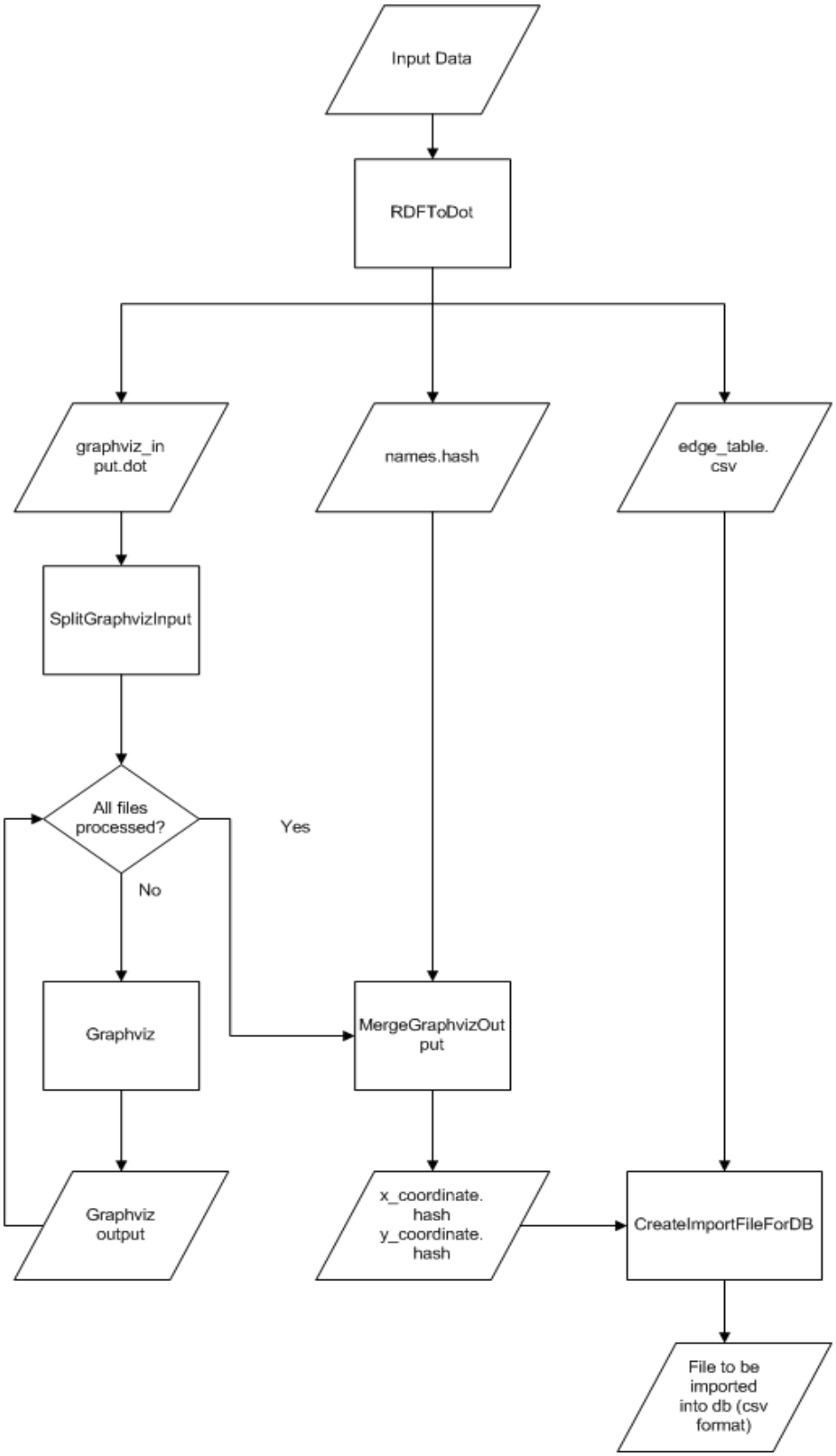


Εικόνα 1: Τα υποσυστήματα και πως αυτά επικοινωνούν μεταξύ τους

4.2 Περιγραφή Λειτουργιών

4.2.1 Επεξεργασία Δεδομένων Εισόδου

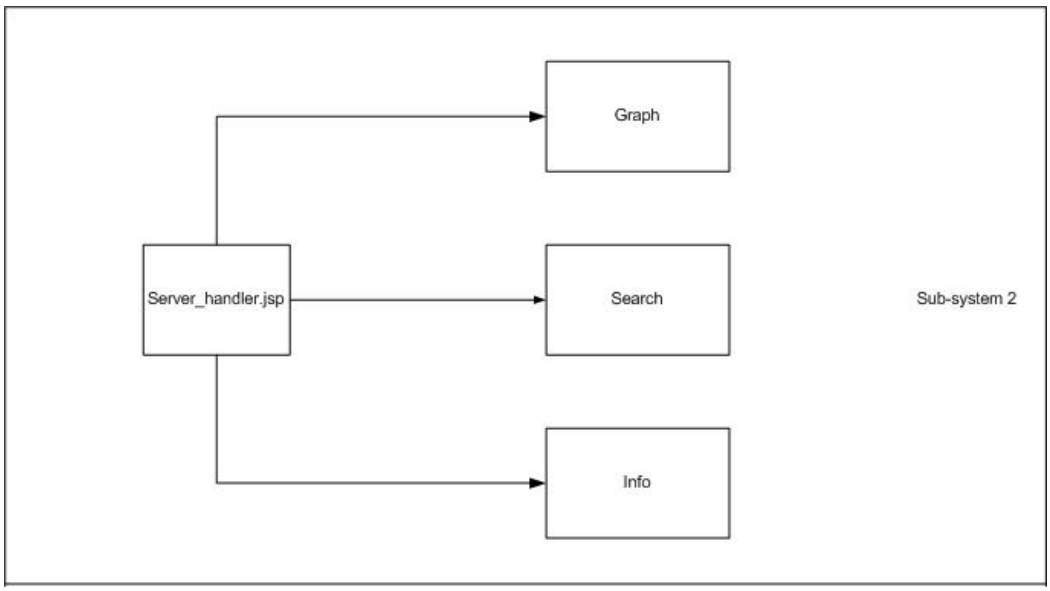
Τα αρχικά δεδομένα εισόδου, που ακολουθούν το RDF μοντέλο, μπορεί να αντιπροσωπεύουν έναν γράφο αλλά δεν είναι στην κατάλληλη μορφή για να τα διαχειριστεί κανένα εργαλείο οπτικοποίησης δεδομένων. Για αυτό αρχικά διαβάζουμε τα δεδομένα εισόδου, κατασκευάζουμε το μοντέλο τους, και το επεξεργαζόμαστε κατάλληλα ώστε οι τριπλέτες να είναι πλέον σε μορφή dot(RDFToDot). Στην συνέχεια, τα δεδομένα με την νέα τους μορφή χωρίζονται σε μικρότερα κομμάτια –δεδομένου ότι τα δεδομένα εισόδου μπορεί να είναι εκατομμύρια τριπλέτες και η μνήμη του συστήματος θα είναι πάντα περιορισμένη αυτό το βήμα είναι απαραίτητο- ώστε να μπορούν να μετατραπούν σε γράφο από το Graphviz (SplitGraphvizInput), τέλος το αποτέλεσμα από τα επιμέρους κομμάτια συνενώνεται (MergeGraphvizOutput) και οι πληροφορίες χρησιμοποιούνται ώστε να φτιαχτεί το αρχείο εισόδου για την βάση δεδομένων (CreateImportFileForDB).



Εικόνα 2: Το διάγραμμα ροής δεδομένων για το java πακέτο InputDataProcessing που επεξεργάζεται τα δεδομένα εισόδου

4.2.2 Υπηρεσίες Server

Για κάθε ενέργεια του χρήστη που απαιτούνται επιπλέον δεδομένα, ενημερώνεται ο server μέσω AJAX request και γίνονται τα κατάλληλα ερωτήματα στην βάση δεδομένων ώστε να βρεθούν τα ζητούμενα δεδομένα, να διαμορφωθούν κατάλληλα και να επιστρέψουν στον client ώστε να παρουσιαστούν στον χρήστη. Οι λειτουργείς που μπορεί να κληθεί να χειριστεί ο server αφορούν την εμφάνιση ενός τμήματος του γράφου (GraphVisualisation) για το παράθυρο στο οποίο μετακινήθηκε ο χρήστης, την αναζήτηση στα δεδομένα με κάποια λέξη-κλειδί (SearchInGraph) και την εμφάνιση της πλήρης ετικέτας ενός κόμβου καθώς και τους γείτονες του (NodeNeighborsPanel), τόσο τα παιδία όσο και τους γονείς του.

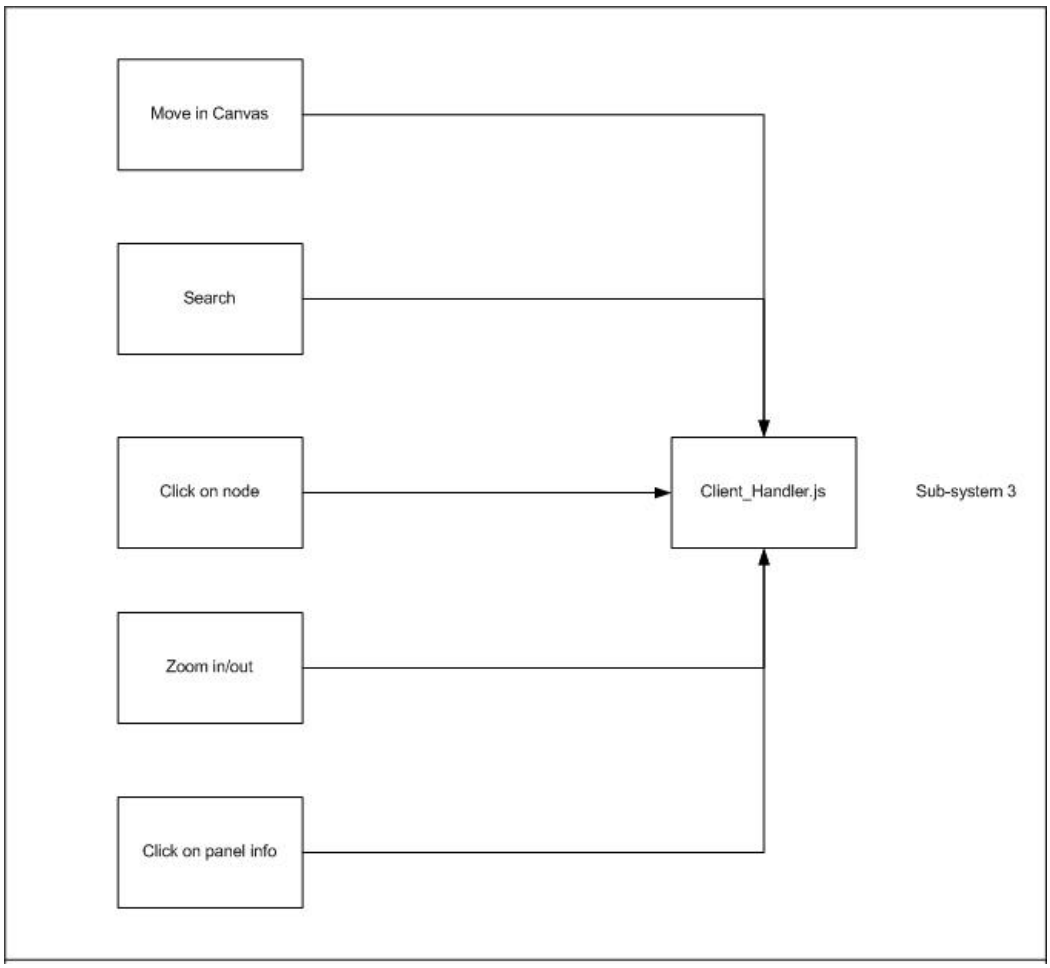


Εικόνα 3: Η jsp σελίδα καλεί την κατάλληλη Java κλάση ανάλογα με την ενέργεια του χρήστη που καλείται να διαχειριστεί

4.2.3 Υπηρεσίες Client

Το υποσύστημα αυτό προσπαθεί να κάνει όσο το δυνατόν ευκολότερη την περιήγηση του χρήστη στον γράφο. Προσφέρει στον χρήστη την δυνατότητα να κινηθεί μέσα στον καμβά είτε χρησιμοποιώντας τις scrollbars είτε χρησιμοποιώντας το πληκτρολόγιο (arrow keys, page up/down) είτε κρατώντας πατημένο το αριστερό κουμπί του ποντικού και κινώντας το ποντίκι στον καμβά. Το κεντρικό παράθυρο της web εφαρμογής μέσα στο οποίο εμφανίζεται ο γράφος (καμβάς) έχεις τις διαστάσεις ολόκληρου του γράφου όπως αυτός δημιουργήθηκε από το πρώτο υποσύστημα, παρόλο που αυτός δεν εμφανίζεται πότε ολόκληρος. Έτσι για κάθε θέση μέσα σε αυτό το παράθυρο, στην οποία μετακινείται ο χρήστης, υπολογίζουμε τις συντεταγμένες του, λαμβάνοντας υπόψη μας την θέση των scrollbars και τις τρέχουσες διαστάσεις του παραθύρου, και στην συνέχεια ζητάμε από τον server να μας επιστρέψει μόνο

το τμήμα του γράφου που εμφανίζεται σε αυτό το παράθυρο. Με αυτόν τον τρόπο ο χρήστης θα δει και πάλι ακριβώς την ίδια εικόνα αν επιστρέψει σε κάποιο τμήμα του γράφου, παρόλο που αυτή θα ζητηθεί εκ νέου από την βάση δεδομένων. Έχουμε ακόμα εξασφαλίζει ότι θα στέλνεται μόνο ένα ερώτημα στον server για κάθε μετακίνηση του χρήστη σε ένα νέο παράθυρο για να μην δημιουργείται ανώφελη επιβάρυνση στον server. Η εφαρμογή επιτρέπει επιπλέον στον χρήστη να μεταβεί σε οποιονδήποτε κόμβο εμφανίζεται στο αριστερό panel της με ένα απλό click του ποντικιού είτε αυτός έχει προέλθει από κάποια αναζήτηση είτε αποτελεί γείτονα κάποιου κόμβου. Επιτρέπει επιπλέον στον χρήστη να διαλέγει ανάμεσα σε δύο σύνολα δεδομένων (Yago και DBpedia) και να αλλάζει το ποσοστό εστίασης του γράφου είτε χρησιμοποιώντας τα κουμπιά του panel είτε με ctrl+z+mouse wheel.



Εικόνα 4: Ενέργειες του χρήστη που αποτελούν το έναυσμα για την επικοινωνία της εφαρμογής με τον server

4.3 Περιγραφή Κλάσεων

Τα παραπάνω υποσυστήματα όπως αναφέραμε χρησιμοποιούν κάποιες Java κλάσεις για να επιτύχουν τις ζητούμενες λειτουργίες. Στην συνέχεια παρουσιάζουμε αναλυτικότερα τις κλάσεις αυτές. Οι κλάσεις 4.3.1-4 αποτελούν μέρος του πακέτου InputDataProcessing το οποίο αφορά την επεξεργασία των δεδομένων εισόδου (πρώτο υποσύστημα), ενώ οι κλάσεις 4.3.5-7 αφορούν τις υπηρεσίες server (δεύτερο υποσύστημα).

4.3.1 RDFToDot

- Διαβάζει τα δεδομένα εισόδου και κατασκευάζει το Jena μοντέλο τους.
- Για κάθε τριπλέτα των δεδομένων επεξεργαζόμαστε τα υποκείμενο, κατηγορούμενο και αντικείμενο ώστε να προκύψουν τα δύο αρχεία εξόδου
- Το πρώτο αρχείο εξόδου είναι της μορφής subject_localname -> object_localname, στην κατάλληλη μορφή για το graphviz (dot μορφή).
- Το δεύτερο αρχείο εξόδου περιέχει μία γραμμή πληροφορίας για κάθε τριπλέτα των δεδομένων εισόδου, η οποία καταγράφει τα subject_id, subject_name, subject_localname, edge_name, edge_localname, object_id, object_name και object_localname σε csv μορφή. Το αρχείο αυτό στην συνέχεια (στην κλάση CreateImportFileForDB όπου θα έχουμε επιπλέον στοιχεία) θα συμπληρωθεί με τις συντεταγμένες των κόμβων ώστε να γίνει import στην βάση δεδομένων.

4.3.2 SplitGraphvizInput

- Παίρνει ως είσοδο το πρώτο αρχείο εξόδου της κλάσης RDFToDot.
- Χωρίζει το αρχείο, ανάλογα με το μέγεθος του σε μικρότερα κομμάτια ώστε να μπορεί να τα διαχειριστεί το graphviz με την διαθέσιμη μνήμη.

4.3.3 MergeGraphvizOutput

- Δέχεται ως είσοδο τα αρχεία εξόδου του graphviz.
- Υπολογίζει το offset για κάθε αρχείο με βάση το μέγεθος του γράφου που αναφέρεται στην αρχή του κάθε αρχείου.
- Για κάθε εγγραφή κόμβου στα αρχεία εισόδου απομονώνονται οι x, y συντεταγμένες του και δημιουργούνται δύο εγγραφές στους αντίστοιχους hash maps.
- Όταν ολοκληρωθεί η διαδικασία για όλους τους κόμβους τα hash maps γράφονται σε αρχείο.

4.3.4 CreateImportFileForDB

- Δέχεται ως είσοδο τα hash maps που φτιάχτηκαν από την κλάση MergeGraphvizOutput και το δεύτερο αρχείο εξόδου της κλάσης RDFToDot
- Το αρχείο ενημερώνεται με την προσθήκη των συντεταγμένων για κάθε κόμβο οπότε και παίρνει την τελική του μορφή, subject_id, subject_name, subject_localname, x_1, y_1, edge_name, edge_localname, object_id, object_name, object_localname, x_2, y_2. Το αρχείο αυτό είναι που γίνεται import στην βάση δεδομένων.

4.3.5 GraphVisualisation

- Καλείται από την jsp σελίδα με βάση τις τρέχουσες συντεταγμένες της οθόνης του χρήστη στον γράφο.
- Στέλνει ερώτημα στην βάση ώστε να εντοπιστούν όλες οι τριπλέτες που είτε κάποιος κόμβος τους βρίσκεται στην οθόνη είτε η ακμή τους περνάει από αυτήν.
- Επεξεργάζεται την απάντηση από την βάση δεδομένων και φτιάχνει τις εντολές του mxgraph που θα δημιουργήσουν τον γράφο στην οθόνη του χρήστη.
- Επιστρέφει μια συμβολοσειρά, έτοιμη να εκτελεστεί.

4.3.6 SearchInGraph

- Καλείται από την jsp σελίδα κάθε φορά που ο χρήστης κάνει αναζήτηση με βάση κάποια λέξη κλειδί.
- Στέλνει ερώτημα στην βάση ώστε να εντοπιστούν κόμβοι που περιέχουν την λέξη κλειδί. Το ερώτημα στην βάση επιστρέφει συγκεκριμένο αριθμό κόμβων για να αποφευχθούν απαντήσεις με εκατομμύρια κόμβους (κυρίως λόγω αναζητήσεων με μικρές ή συχνές λέξεις κλειδιά) και να προστατευτεί ο χρήστης από μια αχανή και χωρίς νόημα απάντηση.
- Επεξεργάζεται την απάντηση από την βάση δεδομένων και φτιάχνει τις εντολές που θα δημιουργήσουν τον πίνακα στο panel της εφαρμογής.
- Επιστρέφει μια συμβολοσειρά, έτοιμη να εκτελεστεί.

4.3.7 NodeNeighborsPanel

- Καλείται από την jsp σελίδα κάθε φορά που ο χρήστης κάνει click σε κάποιον κόμβο του καμβά.
- Στέλνει ερώτημα στην βάση ώστε να εντοπιστούν κόμβοι που είναι γείτονες του.

- Επεξεργάζεται την απάντηση από την βάση δεδομένων και φτιάχνει τις εντολές που θα δημιουργήσουν τον πίνακα στο panel της εφαρμογής.
- Επιστρέφει μια συμβολοσειρά, έτοιμη να εκτελεστεί.

4.4 Βάση Δεδομένων

Το σχεσιακό μοντέλο της βάσης δεδομένων που χρησιμοποιούμε αποτελείται από έναν μόνο πίνακα κάθε γραμμή του οποίου αντιστοιχίζεται σε μία τριπλέτα του αρχικού αρχείου εισόδου. Επειδή η εφαρμογή έχει σχεδιαστεί για να ανταποκρίνεται σε αρχεία εκατομμυρίων τριπλετών θεωρήθηκε ότι είναι προτιμότερο να συγκεντρωθεί όλη η πληροφορία σε έναν πίνακα ώστε να αποφευχθεί η ανάγκη για join πινάκων την ώρα που ο χρήστης περιμένει την πληροφορία.

Για να διευκολυνθούν τα SQL ερωτήματα που ο server στέλνει στην βάση δεδομένων έχουν χρησιμοποιηθεί 5 indexes.

- Ο πρώτος index είναι πάνω στο id_1 ενώ ο δεύτερος πάνω στο id_2. Και οι δύο χρησιμοποιούνται για την πιο γρήγορη ανάκτηση πληροφοριών όσο αφορά τους γείτονες κάποιου κόμβου για την εμφάνιση των πληροφοριών στο panel.
- Ο τρίτος index είναι τύπου spatial πάνω στην στήλη edge_rectangle. Η στήλη αυτή κατασκευάζεται από τις συντεταγμένες x, y των δύο κόμβων και είναι ουσιαστικά το ορθογώνιο του περικλείει την κάθε ακμή. Το index αυτό μας βοηθάει να προσδιορίσουμε τις τριπλέτες που πρέπει κάθε φορά να εμφανίζονται στον χρήστη.
- Οι δύο τελευταίοι indexes είναι τύπου fulltext πάνω στις στήλες panel_label_1 και panel_label_2 και συμβάλουν ώστε να γίνεται πιο αποδοτικά η αναζήτηση με βάση κάποια λέξη- κλειδί.

Triplets_table	
I1	id_1
I4	panel_label_1
	graph_label_1
	x_1
	y_1
	panel_edge_label
	graph_edge_label
I2	id_2
I5	panel_label_2
	graph_label_2
	x_2
	y_2
I3	edge_rectangle

Σημειώνεται ότι τα x_1, y_1, x_2, y_2 που έχουν αποθηκευμένα τις συντεταγμένες των δύο κόμβων δεν μπορούν μην έχουν τιμή (να είναι null). Αυτό συμβαίνει γιατί σε αυτά στηρίζεται κατά το import η δημιουργία του edge_rectangle πάνω στο οποίο υπάρχει spatial index και δεν μπορεί να μην έχει τιμή. Ο πίνακας στην βάση κατασκευάζεται με την εντολή

Create Table triplets_table (id_1 INT ,panel_label_1 TEXT,graph_label_1 TEXT,x_1 FLOAT NOT NULL,y_1 FLOAT NOT NULL, panel_edge_label TEXT, graph_edge_label TEXT,id_2 INT,panel_label_2 TEXT,graph_label_2 TEXT,x_2 FLOAT,y_2 FLOAT, edge_rectangle GEOMETRY NOT NULL, index id1_panel (id_1), index id2_panel (id_2), spatial index edge_index (edge_rectangle), fulltext index index_1 (panel_label_1), fulltext index index_2 (panel_label_2)) ENGINE=MyISAM;

Ενώ τα δεδομένα εισάγονται με την εντολή

load data local infile ‘file_to_import.csv' into table triplets_table fields terminated by ',' enclosed by "" lines terminated by '\n' (id_1, panel_label_1, graph_label_1, x_1, y_1, panel_edge_label, graph_edge_label ,id_2, panel_label_2, graph_label_2, x_2, y_2) SET edge_rectangle=GeomFromText(CONCAT('POLYGON((',x_1,' ',y_1,',',x_1,' ', y_2,',',x_2,' ',y_2,',',x_2,' ',y_1,',',x_1,' ',y_1,'))'));

5

Υλοποίηση

5.1 Λεπτομέρειες υλοποίησης

Ένα από τα θεμελιώδη ζητήματα της εργασίας ήταν πως θα μπορέσουμε να έχουμε στον γράφο μία τοπολογία που να συμβάλει στην κατανόηση της προβαλλόμενης πληροφορίας προσφέροντας στον χρήστη έναν γράφο με την κατάλληλη πυκνότητα κόμβων και όλες τις απαραίτητες πληροφορίες. Ένα ακόμα σημείο που ήταν καθοριστικής σημασίας ήταν ο όσο το δυνατό μεγαλύτερος περιορισμός του χρόνου ανταπόκρισης του server. Τέλος, θέλοντας να προσφέρουμε στον χρήστη την δυνατότητα της εύκολης περιήγησης στα δεδομένα δημιουργήσαμε μια σειρά επιπρόσθετων εργαλείων για να πλαισιώσουν τον γράφο.

5.1.1 Ονόματα Κόμβων

Τα ονόματα των κόμβων τα διαχειριστήκαμε με δύο τρόπος. Για τα resources στον γράφο εμφανίζονται μόνο τα local names, παραλείπονται πλήρως τα name spaces. Ολόκληρο το όνομα του κόμβου εμφανίζεται στο panel της εφαρμογής είτε όταν ο χρήστης πατήσει πάνω στον συγκεκριμένο κόμβο είτε κατά την αναζήτηση με λέξη-κλειδί Όταν ο κόμβος είναι συμβολοσειρά τότε εμφανίζονται στον γράφο μόνο οι 20 πρώτοι χαρακτήρες της. Αν η συμβολοσειρά είναι μεγαλύτερη τότε η συνέχεια της θα εμφανιστεί στο panel αν ο χρήστης το επιλέξει.

5.1.2 Μεταβλητές Graphviz

Το Graphviz είναι ένα προγραμματιστικό εργαλείο που υποστηρίζει μια πληθώρα διαφορετικών λειτουργιών και για να τις υποστηρίξει πλήρως προσφέρει και μία μεγάλη γκάμα μεταβλητών για να μπορεί ο χρήστης να καθορίζει την μορφή του γράφου με βάση τις ανάγκες της εφαρμογής του. Για τις δικές μας ανάγκες θέλαμε να εξασφαλίσουμε ότι δεν θα υπάρχουν επικαλύψεις στους κόμβους του γράφου αλλά θέλαμε να αποφύγουμε και έναν γράφο πολύ αραιό που θα έκανε την ανάγνωση του πολύ δύσκολη. Έτσι μετά από διαφορετικούς συνδυασμούς καταλήξαμε στις ranksep=3,K=2.5, overlap=prism ως τις μόνες μεταβλητές που έπρεπε να αλλάξουμε από τις default τιμές τους. Ακόμα, ανάμεσα στις διαφορετικές τοπολογίες που προσφέρει επιλέχθηκε η sfdp. Μπορεί να μην προσφέρει το καλύτερο δυνατό αποτέλεσμα, σε ότι αφορά την τοπολογία του γράφου -αυτό προκύπτει από την χρήση της βιβλιοθήκης neato- έχει όμως την δυνατότητα διαχείρισης πολύ μεγαλύτερου αριθμού κόμβων από οποιαδήποτε άλλη βιβλιοθήκη. Θεωρήθηκε λοιπόν σκόπιμο να θυσιαστεί λίγο η ποιότητα του τελικού αποτελέσματος, κυρίως λόγω της ανομοιομορφίας στην κατανομή των κόμβων αφού υπάρχουν περιοχές με μεγάλη πυκνότητα, ώστε να αυξηθεί η δυνατότητα της εφαρμογής να διαχειρίζεται με ευκολία εκατομμύρια κόμβους.

5.1.3 Δημιουργία γράφου

Κάθε φορά που το παράθυρο που ενδιαφέρει τον χρήστη βρίσκεται εκτός του γράφου που ήδη έχει φορτωθεί στην οθονη καλείται μέσω Ajax ο server με παραμέτρους την πάνω αριστερή και την κάτω δεξιά γωνία του ζητούμενου παραθύρου. Είναι αναγκαίο να στέλνονται και οι δύο γωνίες αφού το μέγεθος του παραθύρου είναι άγνωστο στον server και μπορεί να βρεθεί μόνο στον client με την βοήθεια της jQuery. Στην συνέχεια ο server συνδέεται στον server της MySQL, χρησιμοποιώντας τα σωστά username και password και στέλνει ερώτημα στην βάση δεδομένων ζητώντας της όλες τις ακμές που διασχίζουν ή περιλαμβάνονται μέσα στο παράθυρο. Το SQL query αυτού του ερωτήματος είναι

SELECT id_1, graph_label_1, x_1, y_1, graph_edge_label, id_2, graph_label_2, x_2, y_2 from triplets_table WHERE ST_Intersects (edge_rectangle, GeomFromText ('polygon((x_1, y_1, x_2, y_2))'))

Το ερώτημα αυτό εκμεταλλεύεται τόσο το spatial index που έχω πάνω στο edge_rectangle όσο και την δυνατότητα που προσφέρει η MySQL από την έκδοση 5.6 και μετά για να γίνονται χωρικά ερωτήματα πάνω στην μορφή των σχημάτων και όχι των ορθογωνίων που τα περικλείουν (ST).

Στην συνέχεια επεξεργαζόμαστε κάθε γραμμή του αποτελέσματος ώστε να φτιάξουμε τις ζητούμενες εντολές του mxgraph. Το mxgraph όμως δεν κάνει κανέναν έλεγχο σε ότι αφορά

την εμφάνιση πολλαπλών φορών του ίδιου κόμβου, οπότε ο έλεγχος αυτός έπρεπε να γίνει προγραμματιστικά. Επιλέχθηκε ο έλεγχος αυτός να γίνεται σε αυτό το σημείο όπου γίνεται η γενικότερη επεξεργασία των δεδομένων που έστειλε η βάση δεδομένων και όχι κατά την εμφάνιση του γράφου στον browser από την mxgraph τόσο για λόγους ταχύτητας και μείωσης του κόστους επικοινωνίας όσο και για λόγους καθαρότητας του κώδικα που επιστρέφει στον χρήστη.

5.1.4 Διαγραφή γράφου

Όταν ο χρήστης μετακινείται από μία οθόνη και φορτώνεται ένα νέο τμήμα του γράφου για λόγους μνήμης και αποδοτικότητας το κομμάτι που πλέον δεν χρειάζεται πρέπει να διαγραφεί. Το mxgraph παρέχει στον χρήστη την δυνατότητα είτε να διαγράψει επιλεκτικά τους κόμβους που θέλει από τον γράφο και τις όποιες ακμές είτε να χρησιμοποιήσει την επιθετική μέθοδο graph.removeCells(graph.getChildVertices(graph.getDefaultParent())). Με αυτήν την εντολή διαγράφονται όλοι οι κόμβοι και οι ακμές του αρχικού κόμβου parent που αποτελεί την βάση πάνω στην οποία χτίζεται όλος ο γράφος. Αρχικά είχε αφεθεί στον server η επιλογή των κόμβων που έπρεπε να διαγραφούν αλλά για να μειωθεί ο χρόνος ανταπόκρισης του server και το κόστος επικοινωνίας που περιελάμβανε η μεγαλύτερη απάντηση που στελνόταν επιλέχθηκε τελικά η δεύτερη μέθοδος ως αποδοτικότερη.

5.1.5 Zoom

Ο χρήστης έχει την δυνατότητα να εστιάσει σε κάποιο κομμάτι του γράφου που τον ενδιαφέρει ή να ζητήσει μια πιο γενική εικόνα του. Υπάρχουν δύο δυνατότητες εστίασης στον γράφο είτε από τα κουμπιά στο panel είτε με τον συνδυασμό ctrl+z+mousewheel. Εκτός από τον ίδιο τον γράφο που αλλάζει κλίμακα κατά την αλλαγή στην εστίαση αλλάζουν και οι διαστάσεις του div στο οποίο βρίσκεται ο γράφος αλλά και οι συντεταγμένες με τις οποίες στέλνεται το ερώτημα στην βάση δεδομένων ώστε να διατηρείται η πλήρης λειτουργικότητα της εφαρμογής σε οποιοδήποτε επίπεδο εστίασης και να παραμένει η εικόνα του γράφου που έχει ο χρήστης σε συνέπεια με την πληροφορία στην βάση δεδομένων.

5.1.6 Panning

Η κλασική συμπεριφορά για το panning που προσφέρεται τόσο από την mxgraph όσο και από την jQuery αφοράς στην δυνατότητα να κινείς ένα αντικείμενο με την χρήση του left mouse button ως προς το div μέσα στο οποίο βρίσκεται. Αυτή η συμπεριφορά δεν ήταν η κατάλληλη για την εφαρμογή μας αφού δημιουργούσε αντίφαση μεταξύ των όσων έβλεπε ο χρήστης με τα όσα υπήρχαν στην βάση δεδομένων –βασική αρχή της εφαρμογής μας ήταν ότι ανεξάρτητα από τις όποιες ενέργειες έκανε ο χρήστης η μορφή του γράφου στον browser θα

έμενε πάντα σε αντιστοιχία με την βάση δεδομένων για να διευκολύνεται η κατανόηση της δομής της πληροφορίας. Έτσι αντικαταστήσαμε την συμπεριφορά αυτή ώστε κρατώντας το του left mouse button πατημένο ο χρήστης να μπορεί να κινεί τον γράφο ως προς το παράθυρο στο οποίο βρίσκεται κάνοντας παράλληλα το κατάλληλο scroll στο div.

5.1.7 Μετακίνηση σε επιλεγμένο κόμβο

Για κάθε ετικέτα κόμβου που εμφανίζεται στο panel –είτε η πληροφορία έχει προέλθει μέσα από κάποια αναζήτηση με λέξη-κλειδί είτε μέσα από τους γειτονικούς κόμβους κάποιου επιλεγμένου κόμβου- ο χρήστης έχει την δυνατότητα κάνοντας απλώς ένα click πάνω της να μεταβαίνει στον κόμβο που αντιστοιχεί σε αυτήν. Η πληροφορία για το που βρίσκεται ο κόμβος έρχεται από τον server μαζί με την υπόλοιπη πληροφορία για το panel και το μόνο που χρειάζεται όταν ο χρήστης επιλέξει τον κόμβο είναι να γίνει το div, scroll στο κατάλληλο σημείο –χωρίς να καλέσουμε ξανά τον server.

5.2 Πλατφόρμες και προγραμματιστικά εργαλεία

Η ανάπτυξη του συστήματος μπορεί να χωριστεί σε τρία ανεξάρτητα στάδια. Το πρώτο είναι ο Java κώδικας που γράφτηκε σε NetBeans[7]. Το δεύτερο στάδιο είναι ο JavaScript και JSP κώδικας ο οποίος αναπτύχθηκε σε Dreamweaver[8] και τέλος η βάση δεδομένων που αναπτύχθηκε σε MySQL[9]. Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκαν ορισμένα προγραμματιστικά εργαλεία. Συγκεκριμένα, χρησιμοποιήθηκε η βιβλιοθήκη Jena[10] για να γίνει η πρώτη επεξεργασία των δεδομένων εισόδου, το Graphviz [11]για να καθοριστεί η τοπολογία του γράφου, το mxgraph[12] που είναι αυτό που επιτρέπει στον χρήστη να βλέπει τον γράφο και τέλος το JQuery[13] για να υποστηριχθούν οι διάφορες λειτουργίες της ιστοσελίδας. Η web εφαρμογή δεν απαιτεί καμία εγκατάσταση ούτε έχει υψηλές απαιτήσεις σε hardware. Το μόνο που χρειάζεται είναι ένας browser. Η εφαρμογή δουλεύει σε Internet Explorer (έκδοση 9 και πάνω), Mozilla Firefox και Google Chrome. Ακολουθεί αναλυτικότερη

[7] https://netbeans.org/

[8] http://www.adobe.com/products/dreamweaver.html

[9] http://www.mysql.com/

[10] http://jena.apache.org/

[11] http://www.graphviz.org/

[12] http://www.jgraph.com/mxgraph.html

[13] http://jquery.com/

περιγραφή των προγραμματιστικών εργαλείων που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής.

5.2.1 Apache Jena

Η Apache Jena είναι ένα Java framework που συμβάλει στην δημιουργία εφαρμογών για τον Σημασιολογικό Ιστό. Η βιβλιοθήκη της Jena παρέχει μία μεγάλη συλλογή εργαλείων και Java βιβλιοθηκών που βοηθάνε στην ανάπτυξη εφαρμογών Σημασιολογικού Ιστού και διασυνδεδεμένων δεδομένων, προγραμματιστικών εργαλείων και servers. Βοηθάει ακόμα τους προγραμματιστές να αναπτύξουν κώδικα που χειρίζεται RDF, RDFS, RDFa, OWL και SPARQL συμφωνα με τις δημοσιοποιημένες προτάσεις της W3C.

Το Jena Framework περιλαμβάνει:

- Ένα API (Application Programming Interface-Διεπαφή Προγραμματισμού Εφαρμογών) για την ανάγνωση, επεξεργασία και δημιουργία RDF δεδομένων σε μορφή XML, N-triplets και Turtle.
- Ένα API οντολογιών για τον χειρισμό OWL και RDFS οντολογιών.
- Μία διεπαφή βασισμένη σε κανόνες για ερωτηματα σε RDF και OWL πηγές δεδομένων.
- Μια ποικιλία από στρατηγικές αποθήκευσης που επιτρέπουν σε μεγάλο αριθμό από RDF τριπλέτες να αποθηκεύονται αποτελεσματικά στην μνήμη ή στον σκληρό δίσκο.
- Διακομιστές που επιτρέπουν την δημοσίευση των RDF δεδομένων σε άλλες εφαρμογές χρησιμοποιώντας μία ποικιλία πρωτοκόλλων, περιλαμβανομένου και του SPARQL.

5.2.2 Graphviz Software

Το Graphviz είναι ένα ανοιχτό λογισμικό για την οπτικοποίηση γράφων. Η οπτικοποίηση γράφων είναι ένας τρόπος αναπαράστασης δομημένων πληροφοριών ως διαγράμματα. Έχει σημαντικές εφαρμογές σε δίκτυα, προγραμματισμό υπολογισμών, βάσεις δεδομένων και σχεδιασμό ιστοσελίδων καθώς και διεπαφές άλλων τεχνικών τομέων.

Τα προγράμματα του Graphviz που είναι υπεύθυνα για τον σχεδιασμό των γράφων παίρνουν ως είσοδο μια περιγραφή τους σε μία απλή γλώσσα κειμένου και φτιάχνουν διαγράμματα σε μία σειρά από χρήσιμες μορφές όπως εικόνες και SVG, PDF, Postscript αλλά δίνουν και την δυνατότητα εμφάνισης του γράφου σε έναν διαδραστικό περιηγητή. Το Graphviz προσφέρει μια μεγάλη συλλογή από μεταβλητές που επιτρέπουν στον χρήστη να επηρεάζει από την

μορφολογία του γράφου μέχρι τα χρώματα και τις γραμματοσειρές για τους κόμβους και τις ακμές.

5.2.3 MxGraph

Το MxGraph είναι ένα πρόσθετο της JavaScript που μπορεί να προστεθεί σε οποιαδήποτε html σελίδα με την χρήση μόνο μίας include εντολής αφου είναι αποκλειστικά client-side εφαρμογή. Με αυτό τον τρόπο δίνεται άμεσα πρόσβαση στο πιο πλήρες πρόσθετο που επιτρέπει την κατασκευή διαγραμμάτων απευθείας στον browser. Το mxGraph είναι μία βιβλιοθήκη που μπορεί να αξιοποιηθεί για την δημιουργία εφαρμογών που παρουσιάζουν διαδραστικά διαγράμματα και γράφους. Δεν είναι σχεδιασμένη σαν μία εφαρμογή έτοιμη να χρησιμοποιηθεί αλλά σαν ένα προγραμματιστικό εργαλείο που παρέχει όλες τις συχνά απαιτούμενες συναρτήσεις για να σχεδιάσεις και να αλληλεπιδράσεις με ένα διάγραμμα.

5.2.4 JQuery

Η jQuery είναι μία γρήγορη, μικρή και πλούσια σε χαρακτηριστικά βιβλιοθήκη JavaScript. Κάνει λειτουργίες όπως η διαχείριση και η διάσχιση μίας html σελίδας, τον χειρισμό συμβάντων, το animation αλλά και την επικοινωνία μέσω Ajax πιο απλές. με έναν εύκολο τρόπο χειρισμού και παρέχει την δυνατότητα δημιουργίας ενός κώδικα που θα δουλεύει για όλους τους browsers.Η jQuery βιβλιοθήκη δίνει την δυνατότητα στους προγραμματιστές να δημιουργήσουν plug-ins πάνω από την JavaScript βιβλιοθήκη. Αυτό επιτρέπει στους προγραμματιστές να αλληλέπιδράσουν αφαιρετικά σε χαμηλό επίπεδο δημιουργώντας animations, δυναμικές σελίδες και εφαρμογές.

6

Έλεγχος

6.1 Έλεγχος χρόνου ανταπόκρισης της εφαρμογής

Η βασική λειτουργία της εφαρμογής είναι η εμφάνιση του γράφου στον browser. Υπάρχουν δύο βασικά σενάρια ελέγχου τα οποία προσπαθούν να προσομοιώσουν την περιήγηση ενός χρήστη στον γράφο. Στο πρώτο σενάριο ελέγχου θεωρούμε ότι ο χρήστης επιλέγει να κινηθεί σε τυχαία κομμάτια του γράφου που μπορεί να έχουν ή να μην έχουν κοινά σημεία. Στο δεύτερο σενάριο θεωρούμε ότι ο χρήστης περιηγείται «σειριακά» τον γράφο δηλαδή μετακινείται από το ένα παράθυρο σε ένα διπλανό κρατώντας και κάποιο τμήμα του προηγούμενου. Κατά την διάρκεια του πειραματικού ελέγχου θα μετρήσουμε τον χρόνο ανταπόκρισης της εφαρμογής και για τα δύο σενάρια για μια σειρά από διαφορετικές παραμέτρους. Τέλος θα ελέγξουμε την ανταπόκριση της εφαρμογής στην λειτουργία αναζήτησης.

Ο server που χρησιμοποιεί η εφαρμογή, και στον οποίον βρίσκεται και η βάση δεδομένων έχει 8 GB RAM και λειτουργικό σύστημα Ubuntu 12.04.2 ενώ οι παρακάτω μετρήσεις αφορούν client με 4 GB RAM, Intel® Core™ i3 CPU @ 2.53 GHz και λειτουργικό σύστημα Windows 7, ενώ η εφαρμογή έτρεχε σε Mozilla Firefox.

Δημιουργήθηκε μία συνάρτηση που δέχεται ως είσοδο το μέγεθος του παραθύρου (ύψος και πλάτος), τον αριθμό των επαναλήψεων για την εκτέλεση του σεναρίου και τέλος το ποσοστό του νέου παραθύρου που θα προστίθεται σε κάθε επανάληψη στο παλιό (αν δοθεί μηδέν τότε

επιλέγονται τυχαία παράθυρα). Για κάθε εκτέλεση του σεναρίου θα καταγράφονται ο χρόνος που χρειάζεται ο server για να απαντήσει στο Ajax request, ο αριθμός των κόμβων και των ακμών που περιέχει η απάντηση, το μέγεθος της απάντησης (bytes) και ο χρόνος που χρειάζεται για να γίνει evaluate η απάντηση στον τελικό χρήστη. Όλοι οι χρόνοι εμφανίζονται σε milliseconds.

Στην συνέχεια, επιλέξαμε ορισμένα μεγέθη παραθύρου από 200x200 έως 3000x3000 pixels θεωρώντας ότι η εφαρμογή θα πρέπει να ανταποκρίνεται σε όλα τα μεγέθη οθονών.

Έπειτα επιλέξαμε να εκτελέσουμε το κάθε σενάριο ελέγχου 100 φορές ώστε να μπορούμε να έχουμε μία ασφαλή μέση τιμή τόσο για το χρόνο ανταπόκρισης της εφαρμογής όσο και για το μέγεθος της απάντησης.

Εκτός από το σενάριο όπου τα παράθυρα θα επιλέγονται τυχαία, επιλέξαμε να εκτελέσουμε τρία ακόμα σενάρια όπου το νέο παράθυρο θα έχει 20%, 50% και 80% κοινά σημεία με το παλιό, θεωρώντας πολύ πιθανό ο χρήστης να επιλέγει να κινηθεί είτε πολύ λίγο – μεταφέροντας μια λεπτομέρεια που τον ενδιαφέρει στο κέντρο του παραθύρου, είτε πολύ – προσπαθώντας να ακολουθήσει κάποιο μονοπάτι, είτε χρησιμοποιώντας το panning που επιτρέπει άνετη κίνηση περίπου στο μισό του παραθύρου.

Τα δύο σενάρια έχουν εκτελεστεί τόσο για το yago facts dataset που είναι περισσότερο από 2 εκατομμύρια τριπλέτες (2.664.405) όσο και για το dbpedia dataset που είναι πάνω από 5 εκατομμύρια τριπλέτες (5.905.257).

Σε όλους τους πίνακες που ακολουθούν για λόγους αναγνωσιμότητας θα παρουσιάζονται μόνο οι πίνακες με τους μέσους όρους των 100 επαναλήψεων και όχι αναλυτικά οι τιμές που έχουν καταγραφεί.

6.2 Παρουσίαση ελέγχου για χρόνο εμφάνισης γράφου

6.2.1 Yago facts dataset

Θα ξεκινήσουμε παρουσιάζοντας τα δεδομένα για το yago facts dataset. Στον πρώτο πίνακα παρουσιάζονται τα αποτελέσματα από τα τυχαία παράθυρα.

Window Size	Server response time	Nodes	Edges	Response Size	Rendering time	Total Time
200x200	272.69	110.89	70.31	23885	152.48	425.17
600x600	447.18	213.46	154.18	49105	309.78	756.96
800x800	407.89	157.36	131.4	38447	246.2	654.09
1000x1000	499.58	203.55	156.84	48142	316.91	816.49
1500x1500	664.18	231.4	195.8	57149	381.81	1045.99
2000x2000	628.85	312.72	262.52	77065	423.12	1051.97
2500x2500	1155.58	357.13	335.11	92908	570.74	1726.32
3000x3000	1441.62	437.00	408.31	113376	798.72	2240.34

Πίνακας 1: Τυχαία παράθυρα, yago facts dataset

Παρατηρούμε ότι οι τιμές για τον χρόνο που χρειάζεται η Ajax response αυξάνεται όσο αυξάνεται και το μέγεθος του παραθύρου, μαζί με αυτόν αυξάνονται τόσο οι ακμές όσο και οι κόμβοι που το παράθυρο περιλαμβάνει και φυσικά οι χαρακτήρες της απάντησης.

Στην συνέχεια παρουσιάζεται ο αντίστοιχος πίνακας για το σενάριο κατά το οποίο μετά από κάθε επανάληψη το νέο παράθυρο αποτελείται από 20% νέο καμβά και 80% από το τμήμα του γράφου που είχε ήδη εμφανιστεί στην οθόνη.

Window Size	Server response time	Nodes	Edges	Response Size	Rendering time	Total Time
200x200	135.87	45.73	23.92	8887	60.75	196.62
600x600	141.81	75.65	52.75	16469	124.53	266.34
800x800	1010.71	589.15	435.76	138352	868.78	1879.49
1000x1000	728.89	333.11	274.13	81655	510.79	1239.68
1500x1500	160.00	107.47	73.09	23698	121.37	281.37
2000x2000	756.35	482.06	401.07	119149	648.29	1404.64
2500x2500	1048.29	537.57	433.66	131554	774.43	1822.72
3000x3000	414.28	286.84	224.15	69573	397.99	812.27

Πίνακας 2: 20% νέος καμβάς, yago facts dataset

Σε αυτόν τον πίνακα παρατηρούμε ότι δεν υπάρχει η ίδια ομοιομορφία μεταξύ του μεγέθους του παραθύρου και του αριθμού των τριπλετών που περιέχει. Αυτό συμβαίνει γιατί παρά τις προσπάθειες μας για ένα ομοιογενή γράφο, υπάρχουν σημεία τα οποία είναι είτε πολύ πιο πυκνά –κάποιος κόμβος που έχει πολλές ακμές βρίσκεται σε αυτά, είτε πολύ αραιά – κυρίως ανάμεσα στα επιμέρους τμήματα του Graphviz που έχουμε συνενώσει. Για αυτά τα σεναρια ελέγχου το πρώτο παράθυρο επιλέγεται τυχαία και τα επόμενα βρίσκονται ακριβώς δίπλα του, με δεδομένο ότι προστίθεται ένα μικρό τμήμα μόνο νέου παραθύρου είναι προφανές ότι στην περίπτωση του 800x800 βρεθήκαμε σε κάποιο πυκνό κομμάτι και στην περίπτωση του 1500x1500 σε κάποιο αραιό γεγονός που επηρέασε τις μετρήσεις.

Ακολουθεί ο αντίστοιχος πίνακας για το σενάριο κατά το οποίο μετά από κάθε επανάληψη το νέο παράθυρο αποτελείται από 50% νέο καμβά και 50% από το τμήμα του γράφου που είχε ήδη εμφανιστεί στην οθόνη.

Window Size	Server response time	Nodes	Edges	Response Size	Rendering time	Total Time
200x200	191.92	140.97	79.22	29509	186.02	377.94
600x600	258.39	193.61	115.36	41121	280.42	538.81
800x800	326.12	163.19	96.77	34495	230.46	556.58
1000x1000	430.68	86.08	70.79	20799	130.66	561.34
1500x1500	378.83	300.8	213.15	68551	375.19	754.02
2000x2000	1463.96	598.89	622.86	165527	972.85	2436.81
2500x2500	816.09	457.49	405.21	117636	818.42	1634.51
3000x3000	958.81	365.43	336.04	91154	958.81	1535.64

Πίνακας 3: 50% νέος καμβάς, yago facts dataset

Σε αυτόν τον πίνακα παρατηρούμε ότι παρόλο που δεν έχουμε την ομοιομορφία του πίνακα με τα τυχαία παράθυρα έχουμε μία πολύ καλύτερη εικόνα από τον προηγούμενο. Αυτό συμβαίνει γιατί το παράθυρο ανανεώνεται κατά 50% οπότε ο καμβάς δεν μένει σε πυκνές ή αραιές περιοχές για πολλές επαναλήψεις ώστε να επηρεάσει το τελικό αποτέλεσμα.

Ο αντίστοιχος πίνακας για το σενάριο κατά το οποίο μετά από κάθε επανάληψη το νέο παράθυρο αποτελείται από 80% νέο καμβά και 20% από το τμήμα του γράφου που είχε ήδη εμφανιστεί στην οθόνη, παρατίθεται εδώ.

Window Size	Server response time	Nodes	Edges	Response Size	Rendering time	Total Time
200x200	379.16	236.74	139.48	50206	364.89	744.05
600x600	157.17	69.73	43.01	14578	102.97	260.14
800x800	225.60	87.71	52.98	18662	127.20	352.8
1000x1000	686.07	329.18	223.00	73879	591.33	1277.4
1500x1500	143.28	83.22	48.16	17309	63.58	206.86
2000x2000	372.34	125.52	72.69	26120	185.40	557.74
2500x2500	541.76	318.33	244.37	76396	500.85	1042.61
3000x3000	320.22	244.21	175.03	55671	331.97	652.19

Πίνακας 4: 80% νέος καμβάς, yago facts dataset

Και εδώ παρατηρείται μια ανομοιομορφία μεταξύ του μεγέθους του παραθύρου και τον αριθμό των τριπλετών που περιέχει γιατί καλύπτεται μια μεγάλη περιοχή του γράφου με αποτέλεσμα να περιέχονται περιοχές με ανομοιογενή κατανομή κόμβων.

Στο διάγραμμα που ακολουθεί παρουσιάζεται η σχέση μεταξύ του χρόνου εμφάνισης του γράφου και του μεγέθους της απάντησης.

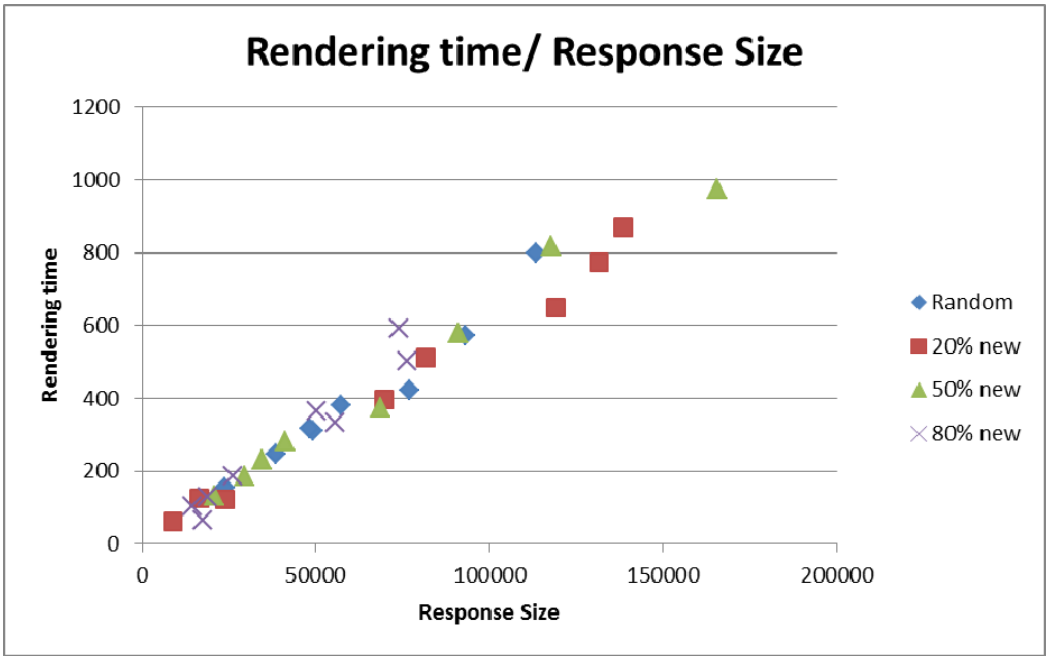


Figure 1: Rendering time/ Response Size, yago facts dataset

Είναι εντυπωσιακό να παρατηρήσουμε ότι ο λόγος των δύο τιμών παραμένει σταθερός και ανεξάρτητος του μεγέθους της απάντησης. Η ίδια ακριβώς εικόνα παρατηρείται και για τον

λόγο Evaluation time/Edges κάτι που ήταν αναμενόμενο μετά τα παραπάνω αποτελέσματα, αφού οι εντολές της mxgraph έχουν συγκεκριμένη μορφή και οι ετικέτες των κόμβων δεν διαφέρουν πολύ ως προς το μέγεθος.

Στην συνέχεια παρουσιάζουμε το διάγραμμα Server response time/Triplets, ώστε να δούμε κατά πόσο ο χρόνος ανταπόκρισης της εφαρμογής εξαρτάται από το μέγεθος της απάντησης.

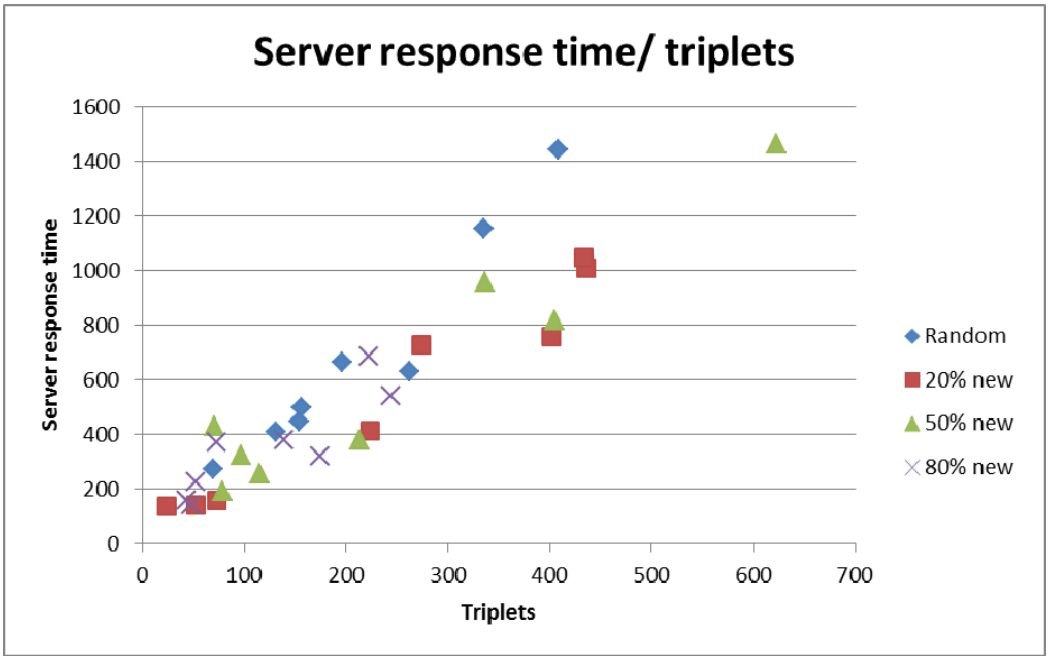


Figure 2: Server response time/ Triplets, yago facts dataset

Είναι σημαντικό να παρατηρήσουμε ότι ο λόγος Server response time/ Triplets δεν παρουσιάζει την ίδια ομοιομορφία που συναντήσαμε στον λόγο Rendering time/ Response Size, αντιθέτως όσο αυξάνει το μέγεθος της απάντησης ο λόγος μικραίνει κάτι που είναι αναμενόμενο αν αναλογιστούμε ότι κάποιος χρόνος χρειάζεται για να γίνει η επικοινωνία ακόμα και όταν η απάντηση είναι κενή, επομένως όσο μεγαλύτερη η απάντηση τόσο πιο καλά κατανέμεται αυτό το κόστος.

Στην συνέχεια παρουσιάζεται το διάγραμμα που αφορά τον αριθμό των τριπλετών και πως αυτός διαμορφώνεται σε σχέση με το μέγεθος του παραθύρου.

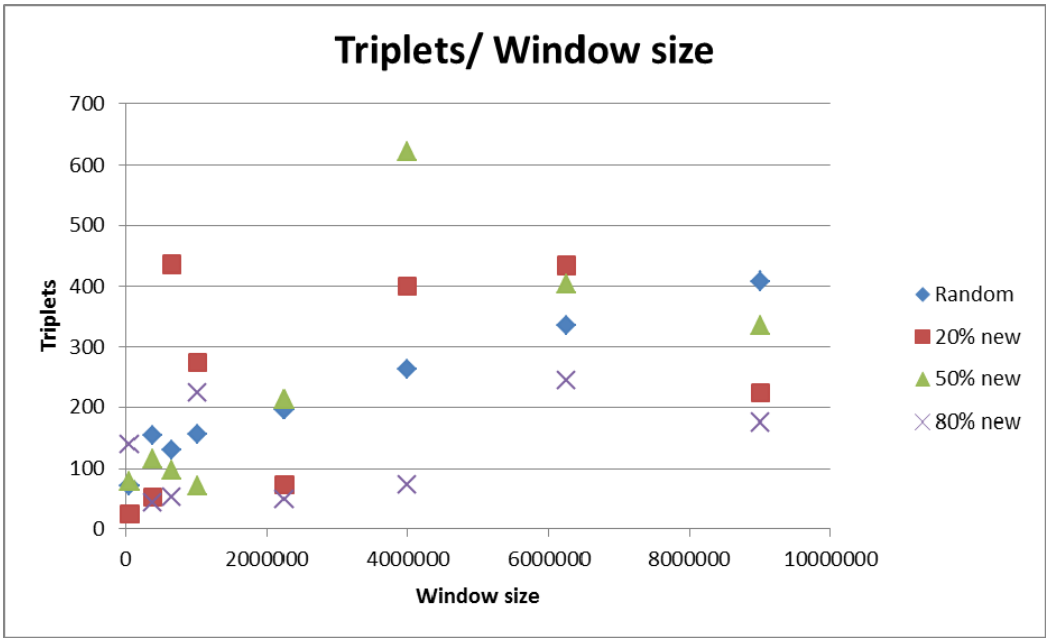


Figure 3: Triplets/Window size, yago facts dataset

Παρατηρούμε ότι στα τυχαία παράθυρα η αύξηση των τριπλετών σε σχέση με το μέγεθος του παραθύρου είναι σχεδόν γραμμική. Το yago dataset έχει μεγάλες παρεκκλίσεις σε ότι αφορά τα άλλα σενάρια ελέγχου, ενώ οι τιμές για το σενάριο ελέγχου όπου προστίθεται κατά 80% νέος καμβάς είναι πιο συνεπείς από ότι στα άλλα δύο σενάρια ελέγχου για γειτονικά παράθυρα.

Τέλος παρουσιάζουμε το διάγραμμα του συνολικού χρόνου ως προς το μέγεθος του παραθύρου.

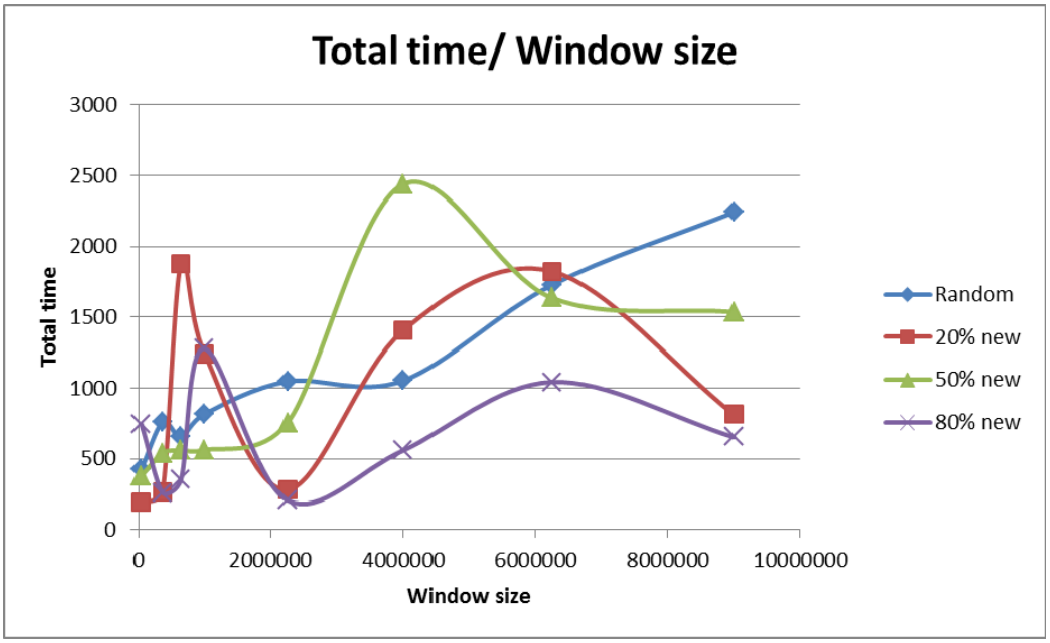


Figure 4: Total time/Window size, yago facts dataset

6.2.2 Dbpedia dataset

Οι αντίστοιχες μετρήσεις έγιναν και για το dbpedia dataset και τα αποτελέσματα παρουσιάζονται στους ακόλουθους πίνακες.

Window Size	Server response time	Nodes	Edges	Response Size	Rendering time	Total Time
200x200	5525.31	168.63	117.66	37440	255.79	5781.1
600x600	6036.26	192.94	153.92	45849	321.2	6357.46
800x800	6759.05	206.76	161.08	48408	326.16	7085.21
1000x1000	7056.37	195.39	165.77	47541	313.96	7370.33
1500x1500	4823.14	324.11	310.58	83938	527.16	5350.3
2000x2000	8314.63	411.73	397.43	107226	695.29	9009.92
2500x2500	8007.66	449.72	431.20	116920	721.37	8729.03
3000x3000	9847.30	536.92	523.08	141859	915.63	10762.93

Πίνακας 5: Τυχαία παράθυρα, dbpedia dataset

Από τον πρώτο κιόλας πίνακα βλέπουμε ότι ενώ ο χρόνος για το yago facts dataset δεν υπερβαίνει σε καμία περίπτωση το 1.5 sec για το dbpedia dataset οι χρόνοι είναι πολύ μεγαλύτεροι. Αυτό συμβαίνει γιατί το dataset της dbpedia έχει πολύ περισσότερες ακμές ανά κόμβο σε σχέση με το yago facts με αποτέλεσμα, παρόλο που έχει 2.5 φορές περισσότερες τριπλέτες, να καταλαμβάνει μόνο το 1/5 του χώρου. Αυτό σημαίνει όχι μόνο ότι τα παράθυρα έχουν μεγαλύτερο αριθμό από τριπλέτες αλλά και ότι το spatial index είναι πολύ επιβαρυμένο και χρειάζεται να ελεγχθούν περισσότερες τριπλέτες από την βάση δεδομένων. Παρόλο τον αυξημένο χρόνο παρατηρείται και εδώ μια ομοιομορφία σε ότι αφορά τον χρόνο ανταπόκρισης του server σε σχέση με το μέγεθος του παραθύρου.

Window Size	Server response time	Nodes	Edges	Response Size	Rendering time	Total Time
200x200	2881.84	93.94	57.70	19923	126.39	3008.23
600x600	3207.97	45.86	25.78	9174	57.45	3265.42
800x800	5255.45	305.97	176.48	66042	387.09	5642.54
1000x1000	5472.99	264.36	154.59	57337	351.12	5804.11
1500x1500	7525.72	206.71	122.35	44931	288.28	7814.00
2000x2000	6417.60	249.3	141.59	53336	333.53	6751.13
2500x2500	7179.37	188.45	107.18	40096	278.23	7457.60
3000x3000	3945.55	212.00	146.94	48335	307.18	4252.73

Πίνακας 6: 20% νέος καμβάς, dbpedia dataset

Παρατηρούμε ότι εδώ δεν έχουμε την ανομοιομορφία που παρατηρήθηκε στο yago dataset αφού ο γράφος είναι πολύ πυκνός σε όλη του την έκταση και δεν υπάρχουν οι ίδιες περιοχές ανομοιογένειας.

Window Size	Server response time	Nodes	Edges	Response Size	Rendering time	Total Time
200x200	6549.26	235.04	135.54	50218	311.43	6860.69
600x600	3365.61	37.42	21.02	7298	48.12	3413.73
800x800	2682.96	147.19	84.63	31345	191.58	2874.54
1000x1000	3443.72	157.98	90.42	33656	206.24	3649.96
1500x1500	9901.46	223.54	135.57	49251	307.97	10209.43
2000x2000	7780.93	223.51	128.42	48029	301.13	8082.06
2500x2500	8658.37	183.92	105.46	39318	282.04	8940.41
3000x3000	6990.64	306.96	203.07	69484	450.47	7441.11

Πίνακας 7: 50% νέος καμβάς, dbpedia dataset

Για πρώτη φορά έχουμε μία τιμή χρόνου στην ανταπόκριση του server που αγγίζει τα 10 sec, παρατηρώντας προσεχτικά τις αναλυτικές μετρήσεις προκύπτει ότι σχεδόν για κανένα παράθυρο ο χρόνος ανταπόκρισης δεν έπεσε κάτω από τα 5 sec ενώ υπήρξαν κάποια συνεχόμενα παράθυρα με χρόνο κοντά στα 20 sec. Τα παράθυρα αυτά όμως επέστρεφαν περίπου 500 κόμβους και 300 ακμές, όγκος δεδομένων που είναι διπλάσιος ακόμα και από τις μέσες τιμές ακόμα και για παράθυρα 3000x3000.

Window Size	Server response time	Nodes	Edges	Response Size	Rendering time	Total Time
200x200	5873.45	151.64	88.16	32271	194.36	6067.81
600x600	5177.25	83.57	45.19	17048	107.24	5284.49
800x800	4298.28	216.3	123.17	46515	282.57	4580.85
1000x1000	6335.83	230.54	133.20	49839	300.76	6636.59
1500x1500	7175.61	163.47	92.13	34621	213.25	7388.86
2000x2000	8311.12	225.39	131.32	48789	319.23	8630.35
2500x2500	8718.58	292.23	214.41	68162	418.16	9136.74
3000x3000	7120.26	270.18	204.35	63712	403.83	7524.09

Πίνακας 8: 80% νέος καμβάς, dbpedia dataset

Στην συνέχεια παρουσιάζουμε το διάγραμμα που μας δίνει την συσχέτιση μεταξύ του μεγέθους της απάντησης και του χρόνου που χρειάστηκε για να δημιουργηθεί ο γράφος.

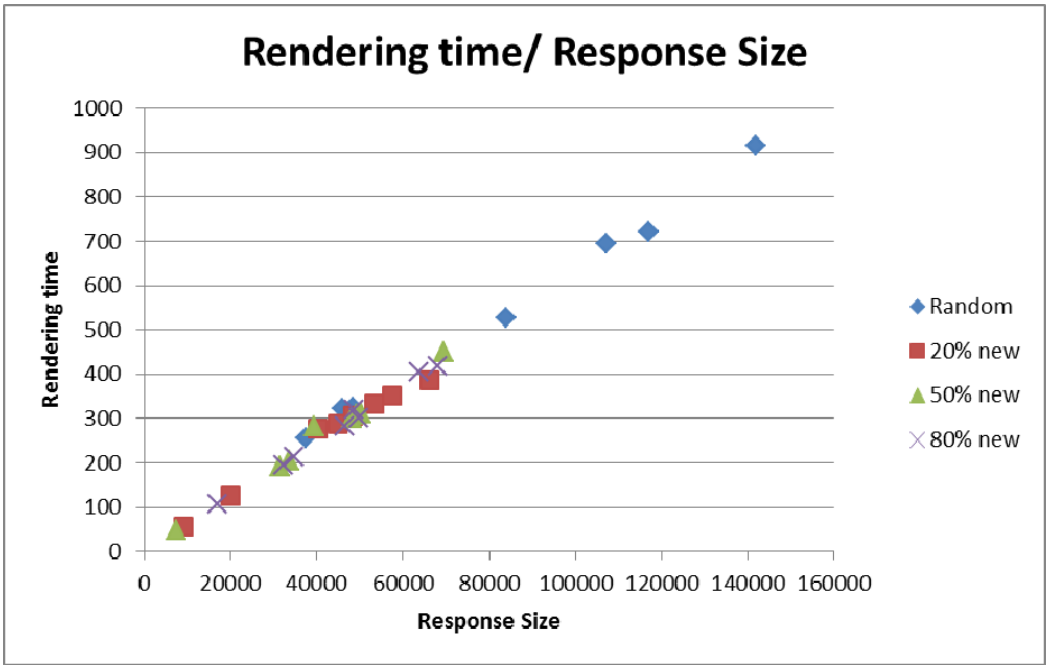


Figure 5: Rendering time/ Response Size, dbpedia dataset

Στην συνέχεια παρουσιάζουμε το αντίστοιχο διάγραμμα Server response time/Triplets, ώστε να δούμε κατά πόσο ο χρόνος ανταπόκρισης της εφαρμογής εξαρτάται από το μέγεθος της απάντησης.

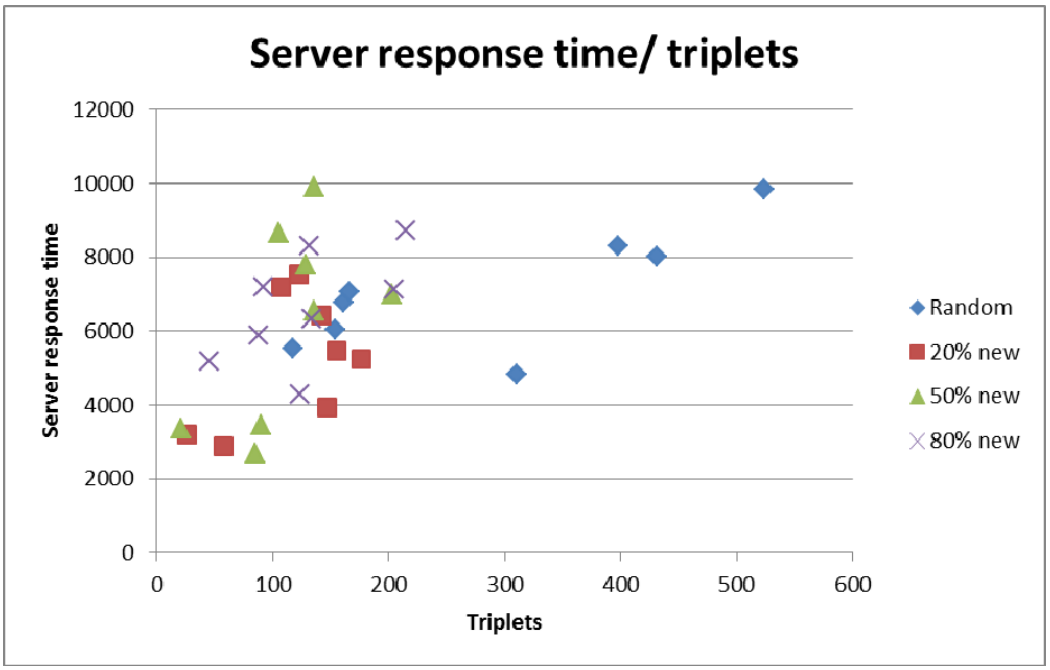


Figure 6: Server response time/ Triplets, dbpedia dataset

Είναι σημαντικό να παρατηρήσουμε ότι και για αυτό το dataset ο λόγος Server response time/ Triplets δεν παρουσιάζει την ίδια ομοιομορφία που συναντήσαμε στον λόγο Rendering time/ Response Size. Ακόμα, είναι εμφανές ότι για το dbpedia dataset δεν έχουμε καλή λειτουργία του spatial index αφού για τα γειτονικά παράθυρα απαιτείται περισσότερος χρόνος για να επιστραφεί μικρότερος αριθμός τριπλετών σε σχέση με τα τυχαία παράθυρα.

Στην συνέχεια παρουσιάζεται το διάγραμμα που αφορά τον αριθμό των τριπλετών και πως αυτός διαμορφώνεται σε σχέση με το μέγεθος του παραθύρου.

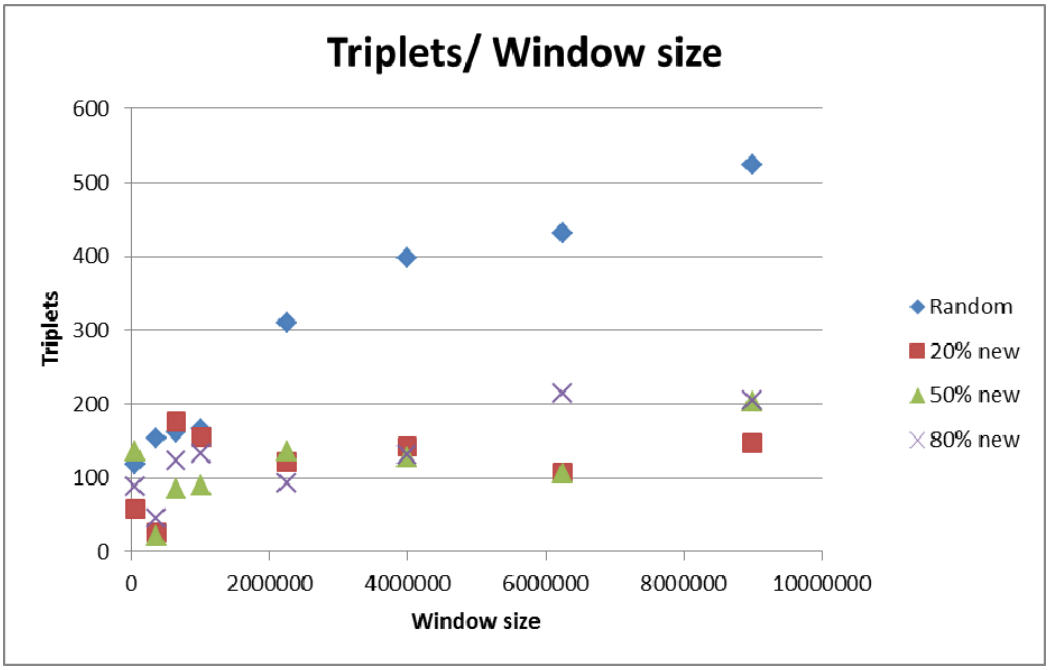


Figure 7: Triplets/Window size, dbpedia dataset

Παρατηρούμε ότι στα τυχαία παράθυρα η αύξηση των τριπλετών σε σχέση με το μέγεθος του παραθύρου είναι και εδώ σχεδόν γραμμική. Το dbpedia dataset εμφανίζει μεγαλύτερη ομοιομορφία στην κατανομή των δεδομένων από το yago dataset, ενώ και στις δύο περιπτώσεις οι τιμές για το σενάριο ελέγχου όπου προστίθεται κατά 80% νέος καμβάς είναι πιο συνεπείς από ότι στα άλλα δύο σενάρια ελέγχου για γειτονικά παράθυρα.

Τέλος παρουσιάζουμε το διάγραμμα του συνολικού χρόνου ως προς το μέγεθος του παραθύρου.

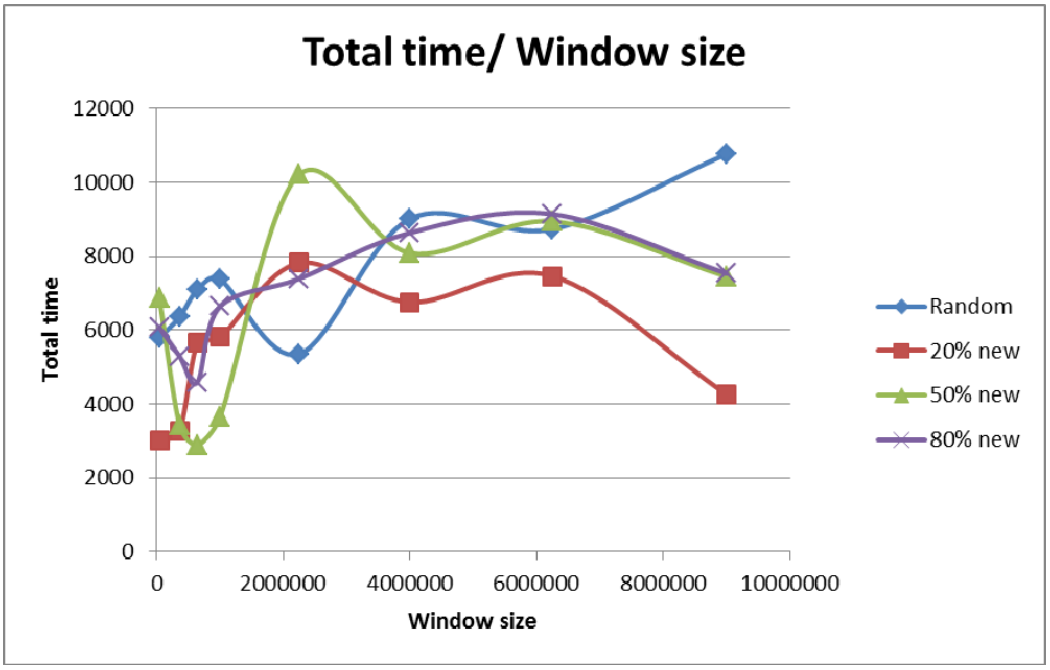


Figure 8: Total time/Window size, dbpedia dataset

6.3 Παρουσίαση ελέγχου για αναζήτηση με λέξη-κλειδί

Σε ότι αφορά την αναζήτηση με λέξη-κλειδί υπάρχουν δύο βασικά σενάρια ελέγχου. Το πρώτο αφορά αναζητήσεις με λέξεις-κλειδιά τα οποία δεν υπάρχουν στο σύνολο δεδομένων που εξετάζουμε ενώ το δεύτερο αφορά λέξεις-κλειδιά που υπάρχουν στα δεδομένα μας. Για το δεύτερο σενάριο επιλέχθηκαν τρεις κατηγορίες λέξεων-κλειδιών. Η πρώτη κατηγορία αφορά πολύ συχνούς όρους (πχ dbpedia και yago αντίστοιχα) που υπάρχουν εκατομμύρια φορές στο σύνολο δεδομένων, η δεύτερη λιγότερο συχνές λέξεις (πχ love, peace, money) που εμφανίζονται μερικές χιλιάδες φορές αλλά και με ονόματα και αριθμούς που εμφανίζονται ελάχιστες φορές (λιγότερες και από το όριο κάθε φορά είχαμε θέσει στην βάση δεδομένων για το μέγεθος της απάντησης που θα επέστρεφε).

Για κάθε τέτοια αναζήτηση μεταβάλαμε το όριο για το ερώτημα στην βάση δεδομένων στις 50, 100, 500 και 1000 απαντήσεις.

Για αυτά τα σενάρια ελέγχου δεν χρειάστηκε να γίνει μεγάλος αριθμός επαναλήψεων γιατί από τις πρώτες κιόλας επαναλήψεις ήταν εμφανές ότι ο χρόνος ανταπόκρισης δεν παρουσιάζει μεγάλες διακυμάνσεις.

6.3.1 Yago facts dataset

Αρχικά παρατηρήσαμε ότι ανεξάρτητα από το όριο στο SQL ερώτημα όταν η βάση δεν έχει αποτελέσματα να επιστρέψει χρειάζεται 5.5 έως 6.5 sec. Για τις συμβολοσειρές που

επιστρέφουν αποτελέσματα αν το όριο είναι μικρότερο από 100 εγγραφές ο server χρειάζεται 0.4 έως 0.5 sec για να ανταποκριθεί, αντίστοιχα για όριο αποτελεσμάτων 500 ο χρόνος κυμαίνεται από 1 έως 1.5 sec ενώ για 1000 ο χρόνος είναι από 2 έως 3 sec. Πρέπει να σημειώσουμε ότι οι αναζητήσεις με λέξεις από τις τρείς κατηγορίες δεν είχαν καμία ουσιαστική διαφορά στον χρόνο ανταπόκρισης του server.

6.3.2 Dbpedia dataset

Και εδώ παρατηρήθηκε ότι ανεξάρτητα από το όριο στο SQL ερώτημα όταν η βάση δεν έχει αποτελέσματα να επιστρέψει έχει σταθερό χρόνο απάντησης ο οποίος κυμαίνεται γύρω από τα 10 sec. Για τις συμβολοσειρές που έχουν να επιστρέψουν πολλά αποτελέσματα ο χρόνος ανταπόκρισης είναι 1 έως 2 sec αν το όριο είναι μικρότερο από 100 εγγραφές, φτάνει τα 5 sec όταν το όριο είναι 500 και αγγίζει τα 8 sec όταν το όριο γίνει 1000. Σε ότι αφορά αναζητήσεις με λέξεις-κλειδιά που επιστρέφουν λίγα μόνο αποτελέσματα, λιγότερα από πέντε, η αναζήτηση χρειάζεται περίπου 10 sec χρόνος που είναι, όπως ήταν αναμενόμενο, σταθερός και ανεξάρτητος από το όριο στην απάντηση.

6.4 Παρουσίαση ελέγχου για αναζήτηση γειτονικών κόμβων

Για λόγους πληρότητας πρέπει να αναφέρουμε ότι οι πληροφορίες για έναν κόμβο λαμβάνονται από τον server σε 0.1 με 0.2 sec από την στιγμή που επιλέγεται ο κόμβους, ανεξάρτητα από το μέγεθος του dataset. Αυτό συμβαίνει γιατί τα ids των κόμβων είναι σωστά κατανεμημένα και τα indexes πάνω σε αυτά λειτουργούν με μεγάλη επιτυχία.

7

Επίλογος

Στην συνέχεια παρουσιάζουμε μία σύνοψη της διπλωματικής εργασίας.

7.1 Σύνοψη και συμπεράσματα

Ολοκληρώνοντας την διπλωματική εργασία έχουμε δημιουργήσει ένα σύστημα που επεξεργάζεται διασυνδεδεμένα δεδομένα που ήταν αρχικά δομημένα στο RDF μοντέλο, τα μετατρέπει σε γράφο, αποθηκεύει την πληροφορία και τα αναπαράγει σε μία μορφή πιο εύχρηστη και κατανοητή για τον απλό χρήστη, όπως ήταν και ο αρχικός μας στόχος.

Το τμήμα της εφαρμογής που αφορά τον χρήστη, η web εφαρμογή, παρουσιάζει κατανοητά, με σαφήνεια και συνέπεια τα δεδομένα με την μορφή ενός γράφου, δίνοντας την δυνατότητα στον χρήστη να περιηγηθεί σε αυτά, να ανακτήσει πληροφορίες και να κατανοήσει την δομή τους, κάτι που καλύπτει πλήρως τους στόχους που είχαμε θέσει.

Σε ότι αφορά τον χρόνο ανταπόκρισης της εφαρμογής μπορούμε να πούμε ότι είμαστε ιδιαίτερα ικανοποιημένοι από την απόδοση της για το yago facts dataset, αφού όλοι οι χρόνοι βρίσκονται κάτω από τα 2.5 sec με την πλειονότητα τους να είναι κάτω από το 1.5 sec. Δυστυχώς δεν μπορούμε να πούμε το ίδιο και για το dbpedia dataset, αφού η αύξηση του όγκου των δεδομένων σε συνδυασμό με έναν πυκνό γράφο με πολλές ακμές, έκανε ορισμένα σενάρια ελέγχου να περάσουν το φράγμα των 10 sec.

8

Βιβλιογραφία

[1] Brunetti J.M., Auer S., Garcia R., The Linked Data Visualization Model" In Proceedings of International Semantic Web Conference (ISWC) 2012

[2] Dadzie A., Rowe M., Petrelli D., Hide the Stack: Toward Usable Linked Data" In Proceedings of Extended Semantic Web Conference (ESWC) 2011

[3] Motta E., Mulholland P., Peroni S., d'Aquin M., Gomez-Perez J., Mendez V., Zablith F., A Novel Approach to Visualizing and Navigating Ontologies" In Proceedings of International Semantic Web Conference (ISWC) 2011

[4] Dadzie A., Rowe M.,Approaches to visualizing Linked Data: A survey" Semantic Web 2(2), 2011

[5] Akrivi Katifori, Constantin Halatsis, George Lepouras, Costas Vassilakis, Eugenia G. Giannopoulou: Ontology visualization methods - a survey. ACM Computer Survey (CSUR) 39(4) (2007)