ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εξειδίκευση οντολογικής γνώσης με χρήση μη δομημένων γλωσσικών περιγραφών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΤΟΥ

ΝΙΚΟΛΑΟΥ Λ. ΚΟΛΙΤΣΑ

Επιβλέπων : Γιώργος Στάμου
Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2014

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

Εξειδίκευση οντολογικής γνώσης με χρήση μη δομημένων γλωσσικών περιγραφών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΤΟΥ

ΝΙΚΟΛΑΟΥ Λ. ΚΟΛΙΤΣΑ

Επιβλέπων : Γιώργος Στάμου
Επίκουρος Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 4η Φεβρουαρίου 2014.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Γεώργιος Στάμου	Ανδρέας Σταφυλοπάτης	Στέφανος Κόλλιας
Επίκουρος Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2014

(Υπογραφή)

.....

ΝΙΚΟΣ ΚΟΛΙΤΣΑΣ

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

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Γιώργο Στάμου για την ευκαιρία που μου έδωσε να ασχοληθώ με αυτή τη διπλωματική εργασία, την καθοδήγηση και βοήθεια που μου παρείχε κατά την εκπόνησή της. Επίσης, θα ήθελα να ευχαριστήσω τον ερευνητή Αλέξανδρο Χορταρά για τη συνεργασία που είχαμε και για τις πολύτιμες συμβουλές του. Τέλος θα ήθελα να ευχαριστήσω την οικογένειά μου που με στηρίζει όλα αυτά τα χρόνια υλικά και πνευματικά.

Περίληψη

Ο σκοπός αυτής της διπλωματικής εργασίας είναι η δημιουργία μιας εφαρμογής που θα βοηθά τον χρήστη στην ανάπτυξη και εξειδίκευση οντολογιών. Αυτό το επιτυγχάνει με δύο τρόπους: πρώτον, παρέχει στο χρήστη γραφικό περιβάλλον μέσω του οποίου μπορεί να διαχειρίζεται και να τροποποιεί την οντολογία· και δεύτερον, εφαρμόζοντας τεχνολογίες επεξεργασίας φυσικής γλώσσας και αυτόματης εξαγωγής όρων προτείνει στο χρήστη περιεχόμενο που θα του φανεί χρήσιμο για τον ορισμό νέων κλάσεων.

Το προτεινόμενο περιεχόμενο, που υποβοηθά την διαδικασία ορισμού νέων κλάσεων, δημιουργείται με τον εξής τρόπο: αρχικά γίνεται αναζήτηση σε βάση δεδομένων για εύρεση στιγμιότυπων της κλάσης που θέλουμε να εξειδικεύσουμε. Στη συνέχεια, βρίσκονται οι γλωσσικές περιγραφές αυτών των στιγμιότυπων και δημιουργείται μια Συλλογή Κειμένων. Πάνω σε αυτή την Συλλογή Κειμένων κάνουμε αυτόματη εξαγωγή όρων (όροι που είναι αντιπροσωπευτικοί και συνοψίζουν το νόημα και τις έννοιες που περιέχουν) και τους προτείνουμε στον χρήστη. Η εξαγωγή όρων γίνεται εφαρμόζοντας διάφορες μεθόδους που έχουν προταθεί από την επιστημονική κοινότητα για το σκοπό αυτό (simple term frequency, average term frequency in corpus, TF-IDF, RIDF, Weirdness, C-value, GlossEx, TermEx). Για να εμπλουτίσουμε ακόμα περισσότερο το προτεινόμενο περιεχόμενο, παρέχουμε υποδομή που μπορεί να αντλεί επιπλέον γνώση από δομημένες περιγραφές των στιγμιότυπων, εκτελώντας τα κατάλληλα ερωτήματα στη βάση και επεξεργάζοντας τα αποτελέσματα.

Μέσω του γραφικού περιβάλλοντος που παρέχει η εφαρμογή, ο χρήστης μπορεί να ορίσει νέες κλάσεις της οντολογίας δημιουργώντας απλά ένα σχήμα γράφου και χωρίς να χρειάζεται να γράψει ούτε κώδικα, ούτε εκφράσεις σε OWL. Ο γράφος αυτός αντιστοιχίζεται αυτόματα σε OWL έκφραση και αν περάσει τους ελέγχους ορθότητας, δημιουργείται η καινούργια κλάση. Η εκφραστικότητα που παρέχεται για τον ορισμό των νέων κλάσεων είναι αυτή της OWL EL, που αντιστοιχεί στην γλώσσα Περιγραφικής Λογικής $\mathcal{EL}$++. Επιλέχθηκε η γλώσσα αυτή γιατί η εκφραστικότητά της σε συνδυασμό με τις πολύ καλές υπολογιστικές της ιδιότητες (reasoning σε πολυωνυμικό χρόνο) την καθιστούν ιδιαίτερα δημοφιλή.

Επίσης, μέσω του γραφικού περιβάλλοντος ο χρήστη μπορεί να δει και να τροποποιήσει την εξεταζόμενη οντολογία με τους ακόλουθους τρόπους: οπτικοποίηση της ιεραρχίας των κλάσεων, προβολή των χρησιμοποιούμενων ιδιοτήτων της οντολογίας και δυνατότητα δημιουργίας καινούργιων (Object και Data Properties), διαχείριση των ατόμων (Individuals) της οντολογίας, έλεγχος συνέπειας Βάσης Γνώσης και εκτέλεση Sparql-DL ερωτημάτων.

Λέξεις Κλειδιά: <<οντολογία, OWL, OWL EL, Περιγραφικές Λογικές, αυτόματη εξαγωγή όρων>>

Abstract

The goal of this diploma thesis is to develop an application that will help users create or expand ontologies. This is achieved in two ways: firstly, by providing a graphical user interface through which one can manage and modify the ontology, and secondly, by proposing content to the user that will be helpful for the definition of the new classes. This content is created by using natural language processing technologies and automatic term extraction.

Specifically, the procedure for creating this proposed content is this: At first, the database is searched for instances of the class that we want to particularize. Then, we find the linguistic descriptions of these instances and we create a text Corpus. By applying automatic term extraction in this corpus, we find the most important and relevant terms and we propose them to the user. This automatic term extraction is done by applying the most prominent and acceptable methods, proposed by the scientific community for this purpose (simple term frequency, average term frequency in corpus, TF-IDF, RIDF, Weirdness, C-value, GlossEx, TermEx). In order to further enrich the proposed content, we provide infrastructure, for extracting additional knowledge from the semi-structured descriptions of the instances, by executing suitable queries to database and processing the results.

Using the graphical user interface, one can define new classes of the ontology simply by creating a shape of a graph, i.e. without writing code or complicated expressions in some ontology language like OWL. This graph is automatically mapped onto the corresponding OWL expression and if it passes the validation's tests, then the new class is created and stored. The expressivity provided for the definition of the new class is equivalent to the OWL EL's expressivity, i.e. the $\mathcal{EL}$++ Description Logic. This DL was chosen, cause due to its expressivity and its excellent computational characteristics (reasoning in polynomial time), it has become very popular.

Furthermore, this application enables users do the following via its graphical user interface: class hierarchy visualization, definition of new classes, visualization of ontology's properties (Object and Data Properties) and possibility to create new, individuals management, consistency checking, Sparql-DL querying.

Keywords: << ontology, OWL, OWL EL, Description Logics, automatic term extraction>>

Πίνακας περιεχομένων

1

Εισαγωγή

1.1 Αναπαράσταση γνώσης, δημιουργία οντολογιών, αυτόματη εξαγωγή όρων

Ένα από τα βασικά προβλήματα της επιστήμης των υπολογιστών είναι η δυνατότητα αναπαράστασης της ανθρώπινης γνώσης και η μοντελοποίηση του κόσμου σε μορφή κατανοητή και επεξεργάσιμη από τους υπολογιστές. Αφού γίνει αυτό, ο υπολογιστής αποκτά την δυνατότητα να εκτελέσει περίπλοκη συλλογιστική και να εξάγει νέα συμπεράσματα και καινούργια γνώση με ευφυή τρόπο που μοιάζει στην ανθρώπινη σκέψη.

Η W3C, που είναι ο οργανισμός για την ανάπτυξη και προτυποποίηση τεχνολογιών για το Παγκόσμιο Ιστό, έχει δημιουργήσει δύο γλώσσες αναπαράστασης γνώσης για το διαδίκτυο: την RDF(S) και την OWL. Όμως η ανάπτυξη νέων οντολογιών για έναν τομέα, αλλά και η περιγραφή του κόσμου και των αντικειμένων του σε αυτές τις γλώσσες, έχει αποδειχτεί μια χρονοβόρα, δύσκολη και δαπανηρή διαδικασία, καθώς απαιτείται η συνεργασία πολλών ατόμων με διαφορετικές ειδικεύσεις. Συγκεκριμένα, απαιτείται η συμμετοχή ειδικών στον τομέα μοντελοποίησης (domain experts), που καλούνται να μεταφέρουν την εξειδικευμένη γνώση τους στην αναπτυσσόμενη

οντολογία, καθώς και μηχανικών γνώσης, που καλούνται να αποτυπώσουν αυτή τη γνώση σε μια τυπική γλώσσα όπως η OWL Αυτό έχει σαν αποτέλεσμα, στην πλειοψηφία των περιπτώσεων, για λόγους ευκολίας, να επιλέγονται απλούστερες μορφές αναπαράστασης γνώσης, όπως είναι το απλό κείμενο σε φυσική γλώσσα (αδόμητη πληροφορία).

Τέτοιες περιπτώσεις αδόμητης ή ημιδομημένης πληροφορίας είναι τα αρχεία κειμένου, τα αρχεία txt, τα αρχεία excel, τα αρχεία html που αναρτώνται στο διαδίκτυο, η πληροφορία που εισάγεται σαν κείμενο σε συστήματα βάσεων δεδομένων κ.α.. Όμως, σε αυτή τη μορφή η γνώση δεν είναι κατανοητή από τους υπολογιστές και δεν μπορεί να αξιοποιηθεί σε ολόκληρο το εύρος της για εξαγωγή χρήσιμων συμπερασμάτων.

Προκειμένου να λυθεί αυτό το πρόβλημα, η επιστημονική κοινότητα έχει στρέψει το ενδιαφέρον της σε τομείς όπως της επεξεργασίας φυσικής γλώσσας και της αυτόματης εξαγωγής όρων. Με τα εργαλεία που έχουν κατασκευαστεί σε αυτούς τους τομείς, προσπαθούμε να εξάγουμε γνώση με αυτόματο τρόπο από αδόμητες γλωσσικές περιγραφές και να την αξιοποιήσουμε για να αναπτύξουμε νέες οντολογίες ή να επεκτείνουμε ήδη υπάρχουσες, δημιουργώντας νέες εξειδικευμένες κλάσεις. Την εξαγόμενη γνώση μπορούμε επίσης να την χρησιμοποιήσουμε για να ταξινομήσουμε αντικείμενα του κόσμου μας σε κατηγορίες (κλάσεις). Δηλαδή σύμφωνα με τα χαρακτηριστικά που εντοπίζουμε στις γλωσσικές περιγραφές των αντικειμένων, μπορούμε να διαπιστώσουμε ότι είναι στιγμιότυπα κάποιων συγκεκριμένων κλάσεων.

1.2 Αντικείμενο διπλωματικής

Η εφαρμογή που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής, έχει σαν στόχο να διευκολύνει τη διαδικασία ανάπτυξης-επέκτασης οντολογιών και την κατηγοριοποίηση αντικειμένων σε κλάσεις, εξάγοντας αυτόματα γνώση από αδόμητες και ημιδομημένες περιγραφές αντικειμένων. Για να το πετύχει αυτό, προσπαθεί να απλοποιήσει, αυτοματοποιήσει, και ενώσει τα διάφορα στάδια της ανάπτυξης. Δηλαδή το στάδιο της άντλησης γνώσης πάνω στον τομέα (από γλωσσικές περιγραφές), και το στάδιο κωδικοποίησής της σε τυπική γλώσσα.

Συγκεκριμένα η εφαρμογή που δημιουργήθηκε μπορεί να χωριστεί σε δύο μεγάλα κομμάτια:

- Το κομμάτι διαχείρισης οντολογιών

- Το κομμάτι επεξεργασίας φυσικής γλώσσας και άντληση πληροφορίας, χρήσιμης για την επέκταση της οντολογίας

Το κομμάτι διαχείρισης οντολογιών ασχολείται με τον τρόπο με τον οποίο μπορεί να αναπαρασταθεί και να διαχειριστεί μια οντολογία εύκολα και γρήγορα από τους χρήστες της, με γραφικό τρόπο και με πλήρη εκφραστικότητα και έλεγχο. Συγκεκριμένα δημιουργείται μια γραφική εφαρμογή με την οποία ο χρήστης μπορεί να επιτελέσει τις παρακάτω λειτουργίες:

- Οπτικοποίηση της ιεραρχία των κλάσεων της οντολογίας δυναμικά καθώς μεταβάλλεται (προστίθενται ή διαγράφονται κλάσεις)
- Ορισμός καινούργιων κλάσεων με εύκολο, διαισθητικό, γραφικό τρόπο, χωρίς να πρέπει να γράψει κώδικα (πχ κώδικα java για το owl api ή κώδικα σε κάποια γλώσσα περιγραφικής λογικής) απλά και μόνο σχεδιάζοντας έναν γράφο. Η εκφραστικότητα που παρέχουμε είναι αυτή της OWL EL.
- Προβολή των ιδιοτήτων της οντολογίας και δυνατότητα ορισμού νέων (Ιδιότητες Αντικειμένων και Ιδιότητες Τύπων Δεδομένων)
- Διαχείριση των ατόμων (Individuals) της οντολογίας. Ποια άτομα είναι μέλη της κάθε κλάσης, εισαγωγή καινούργιων ατόμων ως μέλη κλάσεων, αυτόματη ταξινόμηση αντικειμένων σε νέες εξειδικευμένες κλάσεις με βάση την πληροφορία που αντλούμε από τις γλωσσικές περιγραφές τους.
- Έλεγχος συνέπειας της οντολογίας, και δυνατότητα εκτέλεσης Sparql-DL ερωτημάτων.

Το κομμάτι άντλησης πληροφορίας από επεξεργασία φυσικής γλώσσας έχει σαν στόχο να υποβοηθήσει την διαδικασία επέκτασης μιας οντολογίας. Η λογική είναι η εξής: έστω ότι θέλουμε να εξειδικεύσουμε την κλάση Α (δηλαδή να ορίσουμε μια υποκλάση της) θα μας ήταν πολύ χρήσιμο να πάρουμε όλα τα στιγμιότυπα της κλάσης αυτής και για το καθένα να ψάξουμε να βρούμε την περιγραφή του (απλό κείμενο σε φυσική γλώσσα). Ύστερα να συλλέξουμε όλες αυτές τις περιγραφές, δημιουργώντας μία Συλλογή Κειμένων (Corpus), και πάνω σε αυτή την Συλλογή Κειμένων να εφαρμόσουμε έξυπνους αλγόριθμους εύρεσης σημαντικών όρων (automatic term extraction from text). Οι όροι αυτοί που θα επιλεγούν είναι αντιπροσωπευτικοί των κειμένων και συνοψίζουν το νόημα τους και τις έννοιες που περιλαμβάνουν. Έτσι είναι πολύ πιθανό να μας φανούν χρήσιμοι για τον ορισμό της

νέας υποκλάσης μας. Οι όροι που συλλέγουμε στην εφαρμογή μας είναι ονοματικές φράσεις (noun phrases) και επίθετα (adjectives).

Για παράδειγμα έστω ότι έχουμε μία οντολογία για τις καλές τέχνες και θέλουμε να επεκτείνουμε την κλάση "πίνακας ζωγραφικής". Με την εφαρμογή αυτή, μπορούμε να συνδεθούμε σε οποιαδήποτε βάση δεδομένων και να αναζητήσουμε καταχωρήσεις της βάσης που είναι πίνακες ζωγραφικής. Βρίσκοντας τις καταχωρήσεις που μας ενδιαφέρουν, παίρνουμε τα κείμενα που τις περιγράφουν και από αυτή την Συλλογή Κειμένων που δημιουργείται, αντλούμε τους πιο χρήσιμους και σχετικούς όρους. Για παράδειγμα θα βρίσκαμε όρους της μορφής {Leonardo Da Vinci, Vincent Van Gogh, Pablo Picasso, Expressionism, Futurism, Impressionism, ink, hot wax, water color, historical events, mythology, portraits κλπ }. Αυτούς τους όρους μπορούμε στη συνέχεια να τους χρησιμοποιήσουμε είτε σαν απλά λεκτικά, είτε σαν άτομα, είτε σαν κλάσεις, για τον ορισμό των νέων εξειδικευμένων κλάσεων της οντολογίας μας. Επίσης, αντικείμενα που έχουμε αποθηκευμένα στη βάση δεδομένων και καταχωρημένα σαν στιγμιότυπα γενικών κλάσεων (πχ στιγμιότυπα της κλάσης "πίνακας ζωγραφικής"), μπορούμε πλέον να τα κατηγοριοποιήσουμε ημιαυτόματα σε πιο ειδικές κλάσεις, με βάση τις ιδιότητές τους που τις εξάγουμε από τις γλωσσικές τους περιγραφές (πχ στιγμιότυπο της κλάσης "πίνακας ζωγραφικής του Leonardo Da Vinci που ανήκει στο ρεύμα του εξπρεσιονισμού")

Η εφαρμογή χρησιμοποιεί διάφορους αλγόριθμους για την αυτόματη εξαγωγή όρων. Μερικοί από αυτούς είναι: simple term frequency, average term frequency in corpus, TF-IDF, RIDF, Weirdness, C-value, GlossEx, TermEx.

Η υλοποίηση της εφαρμογής μπορεί να βρεθεί στο σύνδεσμο: https://www.dropbox.com/sh/qvd3b01wmp8vdhm/eAA8bvw5L_

1.3 Οργάνωση κειμένου

Η δομή της διπλωματικής είναι η ακόλουθη:

Κεφάλαιο 1: Το κεφάλαιο αυτό αποτελεί εισαγωγή στο αντικείμενο που θα μας απασχολήσει. Παρουσιάζει το πρόβλημα που πραγματεύεται καθώς και τις λύσεις που δίνονται από την εργασία.

Κεφάλαιο 2: Παρουσιάζεται το θεωρητικό υπόβαθρο που είναι απαραίτητο για την κατανόηση του θέματος της διπλωματικής.

Κεφάλαιο 3: Περιγράφεται η δομή της εφαρμογής, τα υποσυστήματα από τα οποία αποτελείται και οι αρμοδιότητές τους.

Κεφάλαιο 4: Παρουσιάζεται ο τρόπος χρήσης της εφαρμογής σε πραγματικά δεδομένα.

Κεφάλαιο 5: Παρουσιάζεται αναλυτικά η λειτουργικότητα της εφαρμογής και πώς εκτελούνται οι διάφορες εργασίες. Μπορεί να χρησιμοποιηθεί και σαν εγχειρίδιο χρήσης.

Κεφάλαιο 6: Παρουσιάζονται ορισμένες λεπτομέρειες υλοποίησης. Τα πρώτα υποκεφάλαια είναι χρήσιμα για όποιον χρήστη επιθυμεί να επεκτείνει την εφαρμογή με δικούς του αλγόριθμους ταξινόμησης και επεξεργασίας των αποτελεσμάτων.

Κεφάλαιο 7: Επίλογος

Κεφάλαιο 8: Βιβλιογραφία

2

Θεωρητικό υπόβαθρο

2.1 Οντολογίες και Σημασιολογικός Ιστός

Στην εποχή μας η ποσότητα της διαθέσιμης πληροφορίας που υπάρχει στο διαδίκτυο είναι τεράστια και συνεχίζει να αυξάνεται με ραγδαίους ρυθμούς. Η πλειοψηφία αυτής της πληροφορίας είναι αδόμητη, δηλαδή σε φυσική γλώσσα και επομένως μη κατανοητή από τους υπολογιστές (machine understandable). Έτσι παρά αυτή την υπερπληθώρα πληροφοριών που υπάρχει διαθέσιμη και ελεύθερη στο διαδίκτυο δεν μπορούν να φτιαχτούν έξυπνες εφαρμογές που θα την αξιοποιούν και θα προσφέρουν στους χρήστες τους υπηρεσίες υψηλότερου επιπέδου. Στόχος του Σημασιολογικού Ιστού [1][2] είναι να αλλάξει ακριβώς αυτό, δηλαδή τον τρόπο ανάρτησης της πληροφορίας στο διαδίκτυο που πλέον θα γίνεται με δομημένο τρόπο και τυπική σημασιολογία.

2.1.1 Περιγραφικές Λογικές

Οι περιγραφικές λογικές (Description Logics)[3] είναι μια οικογένεια γλωσσών που χρησιμοποιούνται ευρέως για την αναπαράσταση της ανθρώπινης γνώσης και την μοντελοποίηση του κόσμου. Ένας λόγος που συνέβαλε στην επικράτηση των

περιγραφικών λογικών είναι η χρήση τους στον Σημασιολογικό Ιστό. Συγκεκριμένα το θεωρητικό υπόβαθρο που παρέχουν στη γλώσσα OWL (Web Ontology Language) που είναι η καθιερωμένη γλώσσα του Σημασιολογικού Ιστού από το World Wide Web Consortium (W3C)[4].

Οι Περιγραφικές Λογικές (ΠΛ) είναι αποφασίσιμα τμήματα της Λογική Πρώτης Τάξης (First Order Logic) και επομένως έχουν τυπική μαθηματική σημασιολογία. Έτσι δεν υπάρχει αμφισημία στις δηλώσεις, σε αντίθεση με τη φυσική γλώσσα, όπου μια πρόταση μπορεί να έχει περισσότερες από μια ερμηνείες. Είναι ένας καλά μελετημένος κλάδος και υπάρχουν ορθοί (sound) και πλήρεις (complete) αλγόριθμοι για την εξαγωγή συμπερασμάτων (reasoning). Η πολυπλοκότητα και συνεπώς η ταχύτητα εκτέλεσης του reasoning εξαρτάται από την εκφραστικότητα της γλώσσας. Για αυτό υπάρχουν τόσες πολλές διαφορετικές γλώσσες περιγραφικής λογικής.

2.1.1.1 Δομικά στοιχεία Περιγραφικών λογικών

Στις ΠΛ υπάρχουν τριών ειδών οντότητες: οι έννοιες (entities), οι ρόλοι (roles) και τα ονόματα ατόμων (individual names). Οι έννοιες αντιπροσωπεύουν σύνολα ατόμων, οι ρόλοι αντιπροσωπεύουν δυαδικές σχέσεις ανάμεσα στα άτομα, και τα ονόματα ατόμων αντιπροσωπεύουν αντικείμενα του κόσμου που μελετούμε.

Στις ΠΛ περιγράφουμε τον κόσμο που μοντελοποιούμε μέσα από αξιώματα. Δηλαδή δηλώσεις που πρέπει να ισχύουν στον κόσμο μας. Στις ΠΛ, σε αντίθεση με άλλα συστήματα αποθήκευσης γνώσης όπως είναι οι βάσεις δεδομένων, ισχύει η υπόθεση του ανοικτού κόσμου (Open World Assumption). Δηλαδή δεχόμαστε ότι με τα αξιώματα που εισάγουμε στη βάση γνώσης περιγράφουμε μόνο ένα μέρος της αλήθειας. Αυτό σημαίνει ότι η απουσία κάποιας δήλωσης δεν σημαίνει κιόλας ότι δεν ισχύει. Δηλαδή αν δεν έχουμε δηλώσει ότι το άτομο John ανήκει στην έννοια Άνθρωπος δεν σημαίνει ότι μπορούμε να είμαστε και σίγουροι ότι δεν είναι άνθρωπος.

Μια άλλη διαφοροποίηση από τα υπόλοιπα συστήματα και τις βάσεις δεδομένων είναι ότι εδώ δεν ισχύει η υπόθεση μοναδικού ονόματος (Unique Name Assumption). Δηλαδή δύο διαφορετικά ονόματα ατόμων πχ Γιώργος και Γιάννης μπορεί να αντιστοιχούν στο ίδιο αντικείμενο του μοντέλου.

Όπως επισημάνθηκε προηγουμένως, ο λόγος που επιλέγουμε να αναπαραστήσουμε την γνώση μας σε ΠΛ και όχι σε φυσική γλώσσα είναι ότι με τις ΠΛ έχουμε τυπική σημασιολογία. Η σημασιολογία όμως αυτή δεν προϋπάρχει αλλά πρέπει εμείς να την ορίσουμε. Για παράδειγμα, η έννοια Πατέρας στη φυσική γλώσσα έχει συγκεκριμένη ερμηνεία που όλοι γνωρίζουμε (όλοι οι άντρες που έχουν ένα τουλάχιστον παιδί). Όμως η αντίστοιχη έννοια "Πατέρας" σε ΠΛ, δεν σημαίνει τίποτα από μόνη της και θα μπορούσαμε να πούμε ότι είναι υποέννοια της έννοιας Αυτοκίνητο χωρίς να υπάρχει σφάλμα.

Μια ΠΛ ερμηνεία (interpretation) $\mathcal{I}$ ορίζεται από ένα ζεύγος $(\Delta^I , .^I)$, όπου Δ^I είναι ένα μη κενό σύνολο που ονομάζεται χώρος ερμηνείας (domain of interpretation) και περιέχει στοιχεία που ονομάζονται αντικείμενα (objects), και $.^I$ είναι μια συνάρτηση ερμηνείας (interpretation function) που ερμηνεύει κάθε ατομική έννοια A ως ένα υποσύνολο A^I του Δ^I $(A^I \subseteq \Delta^I)$ και κάθε ρόλο R ως ένα υποσύνολο R^I του Δ^I x Δ^I $(R^I \subseteq \Delta^I \text{ x } \Delta^I)$.

2.1.1.2 Ατομικές έννοιες, σύνθετες έννοιες, κατασκευαστές

Ατομικές έννοιες (atomic concepts) είναι οι απλές έννοιες που ορίζονται από μόνες τους (χωρίς την χρήση κατασκευαστών ή άλλων ατομικών εννοιών) και συνήθως αναπαριστώνται με τα γράμματα A, B. Σύνθετες έννοιες (complex concepts) είναι οι έννοιες που ορίζονται με βάση άλλες ατομικές έννοιες και τη χρήση διαφόρων κατασκευαστών και συμβολίζονται συνήθως με τα γράμματα C, D. Δύο έννοιες με ξεχωριστή σημασία είναι οι εξής:

- καθολική έννοια ($\top$): η έννοια αυτή περιέχει όλα τα αντικείμενα του χώρου ερμηνείας. Δηλαδή ισχύει $\top^I = \Delta^I$
- κενή έννοια ($\bot$): η έννοια αυτή ερμηνεύεται ως το κενό σύνολο και δεν περιέχει κανένα αντικείμενο του χώρου ερμηνείας. $\bot^I = \varnothing$

Παρατίθενται μερικοί συνηθισμένοι κατασκευαστές που χρησιμοποιούνται για τη σύνθεση περίπλοκων εννοιών. Υπάρχει η σύμβαση το όνομα της ΠΛ να δίνεται σύμφωνα με τους κατασκευαστές που χρησιμοποιεί και την εκφραστικότητα που έχει.

Επομένως παρατίθεται και το αντίστοιχο όνομα που προσδίδει ο κάθε κατασκευαστής.

- Κατασκευαστής ένωσης (union) $\sqcup$: Δημιουργεί την ένωση δύο εννοιών πχ $C \sqcup D$ με σημασιολογία $(C \sqcup D)^I = C^I \cup D^I$. Προσδίδει το γράμμα $\mathcal{U}$ στο όνομα της ΠΛ.
- Κατασκευαστής τομής (intersection) $\sqcap$: Δημιουργεί την τομή δύο κλάσεων πχ $C \sqcap D$ με σημασιολογία $(C \sqcap D)^I = C^I \cap D^I$. Όλες οι ΠΛ υποστηρίζουν αυτό τον κατασκευαστή.
- Συμπλήρωμα (complement) $\neg$: στη νέα έννοια ανήκουν όλα τα αντικείμενα του χώρου ερμηνείας που δεν ανήκουν στην αρχική έννοια. πχ $(\neg C)^I = \Delta^I \setminus C^I$. Όταν ο κατασκευαστής αυτός επιτρέπεται να μπει μπροστά από περίπλοκες έννοιες και όχι μόνο ατομικές έννοιες, τότε προσδίδει το γράμμα C στο όνομα της ΠΛ.
- Πλήρης υπαρξιακός περιορισμός (existential restriction) $\exists R.C$: η σημασιολογία του είναι $(\exists R.C)^I = \{a \in \Delta^I \mid \exists b \in \Delta^I . (a,b) \in R^I \text{ και } b \in C^I\}$. Προσδίδει το γράμμα $\mathcal{E}$ στο όνομα της ΠΛ. Υπάρχει και ο περιορισμένος υπαρξιακός περιορισμός (limited existential restriction) $\exists R.\top$
- Καθολικός περιορισμός (universal restriction) $\forall R.C$: η σημασιολογία του είναι $(\forall R.C)^I = \{a \in \Delta^I \mid \forall b \in \Delta^I . (a,b) \in R^I \rightarrow b \in C^I\}$
- Περιορισμός πληθυκότητας (number restriction). Αυτός ο κατασκευαστής αυξάνει πολύ την εκφραστικότητα των ΠΛ. Αποτελείται από δυο επιμέρους κατασκευαστές: τον $\leq nR$ (at-most) και τον $\geq nR$ (at-least). Η τυπική σημασιολογία τους είναι: $(\leq nR)^I = \{a \in \Delta^I \mid \#\{b \mid (a,b) \in R^I\} \leq n\}$ (δηλαδή στην έννοια αυτή ανήκουν αντικείμενα του κόσμου όπου συνδέονται μέσω της σχέσης R με το πολύ n αντικείμενα) και
  $(\geq nR)^I = \{a \in \Delta^I \mid \#\{b \mid (a,b) \in R^I\} \geq n\}$ αντίστοιχα. Αυτός ο κατασκευαστής συμβολίζεται με το γράμμα $\mathcal{N}$.

- Προσοντούχος περιορισμός πληθυκότητας (qualified number restriction) $\leq$nR.C και $\geq$nR.C: είναι επέκταση του προηγούμενου κατασκευαστή και συμβολίζεται με το γράμμα Q. Η νέα σημασιολογία που έχει είναι $(\leq nR.C)^I = \{a \in \Delta^I \mid \#\{b \mid (a,b) \in R^I \text{ και } b \in C^I\} \leq n\}$ και

  $(\geq nR.C)^I = \{a \in \Delta^I \mid \#\{b \mid (a,b) \in R^I \text{ και } b \in C^I\} \geq n\}$

2.1.1.3 Αξιώματα οντολογίας

Όπως εξηγήθηκε η περιγραφή του κόσμου γίνεται μέσα από τα αξιώματα. Τα αξιώματα συνήθως τα χωρίζουμε σε τρεις κατηγορίες: Ισχυρισμούς (assertional axioms), Ορολογίες (terminological axioms) και αξιώματα ρόλων (relational axioms). Το σύνολο όλων των ισχυρισμών ονομάζεται Σώμα Ισχυρισμών (Abox) και συμβολίζεται με $\mathcal{A}$. Αντίστοιχα το σύνολο αξιωμάτων ορολογίας ονομάζεται Tbox και συμβολίζεται με $\mathcal{T}$, ενώ το σύνολο των αξιωμάτων ρόλων ονομάζεται Rbox και συμβολίζεται με $\mathcal{R}$.

Abox Axioms

Τα αξιώματα του Abox καταγράφουν τη γνώση που έχουμε για τα άτομα του κόσμου που μοντελοποιούμε. Συγκεκριμένα έχουμε δύο είδη ισχυρισμών :

- Ισχυρισμοί εννοιών (concepts assertions): Με τα αξιώματα αυτά μπορούμε να δηλώσουμε ότι ένα άτομο είναι μέλος μιας συγκεκριμένης έννοιας πχ a:C ή C(a) δείχνει ότι το άτομο a είναι μέλος της έννοιας C. Η τυπική σημασιολογία είναι $(a:C)^I = a^I \in C^I$.
- Ισχυρισμοί ρόλων (role assertions): Με τα αξιώματα αυτά δείχνουμε ότι δύο συγκεκριμένα άτομα συνδέονται μεταξύ τους με μια σχέση πχ (a,b):R ή R(a,b) δείχνει ότι το άτομο a συνδέεται με το b με μια σχέση τύπου R. Η τυπική σημασιολογία είναι $((a,b):R)^I = (a^I,b^I) \in R^I$.

Tbox Axioms

Τα αξιώματα αυτής της κατηγορίας περιγράφουν τις σχέσεις που υπάρχουν μεταξύ των εννοιών. Συγκεκριμένα έχουμε δύο είδη αξιωμάτων:

- Αξιώματα υπαγωγής (subsumption axioms) $C \sqsubseteq D$: Η τυπική σημασιολογία είναι $(C \sqsubseteq D)^I = C^I \subseteq D^I$. Για παράδειγμα Mother $\sqsubseteq$ Human δείχνει ότι η έννοια Mother είναι υποέννοια της Human, επομένως κάθε μέλος της έννοιας Mother είναι επίσης μέλος της έννοιας Human.
- Αξιώματα ισοδυναμίας (equivalence axioms) $C \equiv D$: Η τυπική σημασιολογία είναι $(C \equiv D)^I = (C^I = D^I)$.

Παράδειγμα Tbox:

Woman $\equiv$ Person $\sqcap$ Female

Mother $\equiv$ Woman $\sqcap \exists$ hasChild.Person

MotherWithManyChildren $\equiv$ Mother $\sqcap \geq 3$ hasChild

Rbox Axioms

Τα αξιώματα αυτά περιγράφουν τις ιδιότητες των ρόλων.

- Αξιώματα υπαγωγής ρόλων (role inclusion axioms): πχ parentOf $\sqsubseteq$ ancestorOf Αυτή η δήλωση δείχνει ότι ο ρόλος parentOf είναι υπορόλος του ancestorOf που σημαίνει ότι κάθε ζευγάρι ατόμων που συνδέονται με τη σχέση parentOf συνδέεται και με τη σχέση ancestorOf. Αν για παράδειγμα ισχύει το αξίωμα parentOf(john,george) τότε θα ισχύει επίσης και το ancestorOf(john,george). Τυπική σημασιολογία: $(R \sqsubseteq S)^I = R^I \subseteq S^I$
- Αξιώματα ισοδυναμίας ρόλων (role equivalence axioms): $R \equiv S$ έχει τυπική σημασιολογία $(R \equiv S)^I = (R^I = S^I)$

Για να εκφράσουμε ακόμα περισσότερες ιδιότητες ρόλων, μπορούμε να χρησιμοποιήσουμε σύνθεση ρόλων (role composition). Έτσι μπορούμε να κατασκευάσουμε ρόλους όπως ο uncelOf. Διαισθητικά, αν ο Jim είναι αδερφός του John και ο John είναι πατέρας του George, τότε ο Jim είναι θείος του George. Αυτή η σχέση μεταξύ των ρόλων brotherOf, parentOf και uncleOf μπορεί να αποτυπωθεί με αυτό το αξίωμα: brotherOf $\circ$ parentOf $\sqsubseteq$ uncleOf .

Η σύνθεση ρόλων μπορεί να εμφανίζεται μόνο στο αριστερό μέρος ενός αξιώματος υπαγωγής ρόλων. Επίσης η χρήση της περιορίζεται από διάφορους κανόνες προκειμένου να παραμείνει η ΠΛ αποφασίσιμη.

Ένας άλλος κατασκευαστής είναι ο disjoint roles (ξένοι ρόλοι). Για παράδειγμα κανείς δεν μπορεί να είναι πατέρας και παιδί του ίδιου ατόμου. Οπότε μπορούμε να δηλώσουμε ότι οι ρόλοι parentOf και childOf είναι ξένοι: Disjoint(parentOf,childOf).

Άλλα αξιώματα που μπορούν να χρησιμοποιηθούν για την περιγραφή ιδιοτήτων των ρόλων είναι τα εξής: δήλωση μεταβατικών ρόλων, συμμετρικών, αντισυμμετρικών, αντίστροφων και άλλα. Όμως όλα αυτά είναι απλώς συντακτικές συντομεύσεις που μπορούν να παραχθούν από τα τρία βασικά αξιώματα (role inclusion axioms, role composition και disjoint roles).

	Syntax	Semantics
Individuals:		
individual name	a	a^I
Roles:		
atomic role	R	R^I
inverse role	R^-	$\{\langle x,y\rangle \mid \langle y,x\rangle \in R^I\}$
universal role	U	$\Delta^I \times \Delta^I$
Concepts:		
atomic concept	A	A^I
intersection	$C \sqcap D$	$C^I \cap D^I$
union	$C \sqcup D$	$C^I \cup D^I$
complement	$\neg C$	$\Delta^I \setminus C^I$
top concept	$\top$	Δ^I
bottom concept	$\bot$	$\emptyset$
existential restriction	$\exists R.C$	$\{x \mid \text{some } R^I\text{-successor of } x \text{ is in } C^I\}$
universal restriction	$\forall R.C$	$\{x \mid \text{all } R^I\text{-successors of } x \text{ are in } C^I\}$
at-least restriction	$\geqslant n\, R.C$	$\{x \mid \text{at least } n\ R^I\text{-successors of } x \text{ are in } C^I\}$
at-most restriction	$\leqslant n\, R.C$	$\{x \mid \text{at most } n\ R^I\text{-successors of } x \text{ are in } C^I\}$
local reflexivity	$\exists R.Self$	$\{x \mid \langle x,x\rangle \in R^I\}$
nominal	$\{a\}$	$\{a^I\}$
where $a, b \in \mathsf{N}_I$ are individual names, $A \in \mathsf{N}_C$ is a concept name, $C, D \in \mathbf{C}$ are concepts, $R \in \mathbf{R}$ is a role		

2.1-1 Συντακτικό και σημασιολογία κατασκευαστών

	Syntax	Semantics
ABox:		
concept assertion	$C(a)$	$a^I \in C^I$
role assertion	$R(a,b)$	$\langle a^I, b^I \rangle \in R^I$
individual equality	$a \approx b$	$a^I = b^I$
individual inequality	$a \not\approx b$	$a^I \neq b^I$
TBox:		
concept inclusion	$C \sqsubseteq D$	$C^I \subseteq D^I$
concept equivalence	$C \equiv D$	$C^I = D^I$
RBox:		
role inclusion	$R \sqsubseteq S$	$R^I \subseteq S^I$
role equivalence	$R \equiv S$	$R^I = S^I$
complex role inclusion	$R_1 \circ R_2 \sqsubseteq S$	$R_1^I \circ R_2^I \subseteq S^I$
role disjointness	$Disjoint(R,S)$	$R^I \cap S^I = \emptyset$

2.1-2 Συντακτικό και σημασιολογία αξιωμάτων

2.1.1.4 Βάση Γνώσης

Μια Βάση Γνώσης Σ αποτελείται από την τριάδα $(\mathcal{T},\mathcal{R},\mathcal{A})$: ένα Tbox που συμβολίζεται με $\mathcal{T}$, ένα Rbox που συμβολίζεται με $\mathcal{R}$ και ένα Abox που συμβολίζεται με $\mathcal{A}$.

Μια ερμηνεία $\mathcal{I}$ ικανοποιεί (είναι μοντέλο) μια βάση γνώσης Σ, και συμβολίζεται με $\mathcal{I} \models \Sigma$, αν και μόνο αν η ερμηνεία $\mathcal{I}$ ικανοποιεί (είναι μοντέλο) τα $\mathcal{T},\mathcal{R}$ και $\mathcal{A}$. Επίσης λέμε ότι η Βάση Γνώσης Σ είναι ικανοποιήσιμη, και γράφουμε $\Sigma \not\models \perp$, αν και μόνο αν υπάρχει ερμηνεία $\mathcal{I}$ που ικανοποιεί την Σ.

2.1.1.5 Υπηρεσίες εξαγωγής Συμπερασμάτων (Reasoning Services)

Μια Βάση Γνώσης δεν προσφέρει μόνο την δυνατότητα να αποθηκεύει απλά γνώση, αλλά και να εκτελεί περίπλοκη συλλογιστική και να εξάγει υπονοούμενη γνώση. Συγκεκριμένα οι υπηρεσίες που προσφέρει είναι οι εξής[5][6]:

- Έλεγχος συνέπειας Βάσης Γνώσης (Knowledge Base Consistency): Αυτή η διαδικασία ελέγχει αν η εξεταζόμενη Βάση Γνώσης είναι συνεπής, δηλαδή αν υπάρχει ερμηνεία $\mathcal{I}$ που να ικανοποιεί όλα τα αξιώματα και τους ισχυρισμούς.

- Ικανοποιησιμότητα έννοιας (Concept Satisfiability): Απαντάει στο ερώτημα αν μια έννοια C είναι ικανοποιήσιμη στη Βάση Γνώσης Σ. Επομένως ελέγχει αν υπάρχει ερμηνεία $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$, που να είναι μοντέλο της Σ (δηλαδή να ισχύει $\mathcal{I} \models \Sigma$) και σε αυτή την ερμηνεία να υπάρχει αντικείμενο $d \in \Delta^I$ που να είναι μέλος της έννοιας C (δηλαδή $d \in C^I$).
- Υπαγωγή έννοιας (Concept Subsumption): Δεδομένου των εννοιών C, D και της Βάσης Γνώσης Σ, η διαδικασία αυτή αποφασίζει αν η έννοια C υπάγεται (είναι υποσύνολο) της έννοιας D. Συγκεκριμένα ισχύει $C \sqsubseteq D$ αν δεν υπάρχει καμία ερμηνεία $\mathcal{I}$ μοντέλο της Σ όπου να ισχύει $d \in \Delta^I$, $d \in C^I$ και $d \in (\neg D)^I$.
- Έλεγχος στιγμιότυπων (Instance Checking): Δεδομένου του ατόμου α, της έννοιας C και της Βάσης Γνώσης Σ, αποφασίζει ότι το άτομο α είναι μέλος της έννοιας C, ανν για κάθε ερμηνεία $\mathcal{I}$ που είναι μοντέλο της Σ, ισχύει ότι $\alpha^I \in C^I$. Άλλες παραλλαγές αυτής της υπηρεσίας είναι η Ανάκτηση Ατόμων (Retrieval of individuals) όπου βρίσκει όλα τα άτομα που είναι στιγμιότυπα της κλάσης C και η υπηρεσία Realization of an Individual όπου για το άτομο α βρίσκει όλες τις κλάσεις στις οποίες είναι στιγμιότυπο.

Όλες αυτές οι υπηρεσίες δεν είναι ανεξάρτητες μεταξύ τους. Συγκεκριμένα αν η ΠΛ γλώσσα που εξετάζουμε είναι κλειστή ως προς την άρνηση (δηλαδή αν το συμπλήρωμα μιας έννοιας είναι και αυτό έννοια της γλώσσας μας), τότε όλες αυτές οι βασικές υπηρεσίες συλλογισμού μπορούν να αναχθούν στον έλεγχο συνέπειας της Βάσης Γνώσης. Ακολουθούν μερικά παραδείγματα αναγωγών:

- Η Υπαγωγή Έννοιας ανάγεται σε Ικανοποιησιμότητα έννοιας. Για παράδειγμα η έννοια C υπάγεται στην D $(C \sqsubseteq D)$ ανν η έννοια $(C \sqcap \neg D)$ είναι μη ικανοποιήσιμη.
- Η ικανοποιησιμότητα μιας έννοιας ανάγεται σε έλεγχο συνέπειας της Βάσης Γνώσης. Πχ η έννοια C είναι ικανοποιήσιμη με βάση τη Σ ανν η $\Sigma' = (\mathcal{T}, \mathcal{R}, \mathcal{A} \cup \{C(\alpha_n)\})$ είναι συνεπής. Εδώ το άτομο α_n είναι ένα νέο άτομο όπου δεν συναντάται στη Σ.
- Ο έλεγχος στιγμιότυπων μπορεί να αναχθεί σε συνέπεια Βάσης Γνώσης ως εξής: το άτομο α είναι στιγμιότυπο της έννοιας C αν η $\Sigma' = (\mathcal{T}, \mathcal{R}, \mathcal{A} \cup \{(\neg C)(\alpha)\})$ είναι μη συνεπής.

Αυτοί οι μετασχηματισμοί μπορούν να γίνουν σε γραμμικό χρόνο.

Για τον έλεγχο της συνέπειας της Βάσης Γνώσης, και επομένως και για οποιαδήποτε άλλη υπηρεσία εξαγωγής συμπερασμάτων που ανάγεται σε αυτή, χρησιμοποιούνται οι αλγόριθμοι tableaux. Οι αλγόριθμοι αυτοί έχουν διαφορετική πολυπλοκότητα ανάλογα με την εκφραστικότητα της γλώσσας ΠΛ που εξετάζουμε. Δουλεύουν ως εξής: έχουν ένα σύνολο κανόνων τους οποίους εφαρμόζουν στο Abox και απλοποιούν τις εκφράσεις του. Ο αλγόριθμος τερματίζει όταν πλέον δεν υπάρχει κανένας διαθέσιμος κανόνας που να μπορεί να εφαρμοστεί ή έχει εμφανιστεί κάποια σύγκρουση αξιωμάτων.

2.1.1.6 Δημοφιλείς Περιγραφικές Λογικές

$\mathcal{AL}$

Μια από τις πιο βασικές ΠΛ είναι η γλώσσα $\mathcal{AL}$ (attribute language). Υποστηρίζει μόνο βασικούς κατασκευαστές και είναι η βάση για την δημιουργία νέων πιο εκφραστικών γλωσσών Περιγραφικής Λογικής. Οι περιγραφές εννοιών στη γλώσσα $\mathcal{AL}$ ορίζονται επαγωγικά από την ακόλουθη αφηρημένη σύνταξη (abstract syntax):

$$C, D \rightarrow A \mid \top \mid \bot \mid \neg A \mid C \sqcap D \mid \forall R.C \mid \exists R.\top$$

$\mathcal{ALC}$

Η γλώσσα αυτή επεκτείνει την $\mathcal{AL}$ δίνοντας την δυνατότητα στον κατασκευαστή Συμπλήρωμα ($\neg$) να μπαίνει μπροστά από περίπλοκες έννοιες. Έτσι όμως έμμεσα είναι σαν να εισάγουμε και άλλους κατασκευαστές όπως αυτόν της ένωσης και του πλήρη υπαρξιακού περιορισμού, διότι $C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$ και $\exists R.C \equiv \neg \forall R.\neg C$.

Οπότε το πραγματικό όνομα της $\mathcal{ALC}$ σύμφωνα με του κατασκευαστές που υποστηρίζει είναι $\mathcal{ALUEC}$. Εξαιτίας όμως αυτής της μεγαλύτερης εκφραστικότητας αυξάνεται πολύ και η πολυπλοκότητα του reasoning. Παρακάτω παρατίθεται ένας πίνακας με τις πολυπλοκότητες του reasoning για διάφορες γλώσσες [7].

	concept satisfiability	subsumption	knowledge base satisfiability	instance checking
$\mathcal{AL}$	P	P	P	P
$\mathcal{ALN}$	P	P	P	P
$\mathcal{ALE}$	coNP	NP	coNP	PSPACE
$\mathcal{ALR}$	coNP	NP	coNP	NP
$\mathcal{ALU}$	NP	coNP	NP	coNP
$\mathcal{ALC}$	PSPACE	PSPACE	PSPACE	PSPACE

Πίνακας 2.1.1-1 πολυπλοκότητα εξαγωγής συμπερασμάτων

2.1.2 Γλώσσες αναπαράστασης γνώσης στο Σημασιολογικό Ιστό

Το νόημα του Σημασιολογικού Ιστού είναι τα δεδομένα και γενικά η πληροφορία που δημοσιεύεται στο διαδίκτυο να αναπαριστάται με τρόπο κατανοητό και επεξεργάσιμο από τους υπολογιστές. Αυτό μπορεί να οδηγήσει σε δημιουργία έξυπνων εφαρμογών σε τομείς όπως το ηλεκτρονικό εμπόριο, τις μηχανές αναζήτησης, τη διαχείριση και ανακάλυψη γνώσης από το διαδίκτυο και άλλα. Όπως έγινε φανερό από τη θεωρητική μελέτη των προηγούμενων κεφαλαίων, οι Περιγραφικές Λογικές είναι η κατάλληλη γλώσσα για αυτήν την αναπαράσταση. Προκειμένου όμως να είναι συμβατές με τις χρησιμοποιούμενες τεχνολογίες του σημερινού Ιστού, πρέπει να αναθεωρηθούν και τροποποιηθούν ορισμένα συστατικά τους. Η W3C έχει δημιουργήσει δύο γλώσσες αναπαράστασης γνώσης για το διαδίκτυο: την RDF(S) και την OWL.

2.1.2.1 RDF (Resource Description Framework)

Η γλώσσα αναπαράστασης γνώσης RDF[8][9] παρέχει μια πολύ περιορισμένη και βασική εκφραστικότητα. Συγκεκριμένα προορίζεται για δημιουργία μεταδεδομένων για πόρους του διαδικτύου. Πόρος δεν σημαίνει αποκλειστικά κάτι που μπορεί να ανακτηθεί άμεσα από το διαδίκτυο, πχ ιστοσελίδα ή ηλεκτρονικό αρχείο, αλλά οτιδήποτε αναπαριστούμε με ένα URI reference (URIref). Uriref είναι ένα URI το

οποίο πιθανόν συνοδεύεται και από ένα fragment identifier πχ http://www.exampleontology.org/concepts#concept1. Εδώ το concept1 είναι το fragment ενώ το υπόλοιπο είναι το URI. Με τη γλώσσα αυτή περιγράφουμε έναν πόρο κάνοντας δηλώσεις που έχουν τη μορφή τριάδων. Κάθε τριάδα αποτελείται από το υποκείμενο (subject), την ιδιότητα (property) και το αντικείμενο (object). Το υποκείμενο είναι ο πόρος που περιγράφουμε, η ιδιότητα είναι μια ιδιότητα που έχει ο πόρος και αντικείμενο είναι η τιμή που έχει ο πόρος για τη συγκεκριμένη ιδιότητα.

Αν θέλαμε να αντιστοιχίσουμε την εκφραστικότητα αυτής της γλώσσας σε Περιγραφικές Λογικές, θα λέγαμε ότι μπορούμε να δημιουργήσουμε ισχυρισμούς ρόλων και σχέσεις στιγμιότυπου ανάμεσα σε κάποιο πόρο και σε κάποια έννοια. Συγκεκριμένα κάθε τριάδα s p o που δηλώνουμε μπορεί να θεωρηθεί ισχυρισμός της μορφής p(s,o), που σημαίνει ότι το άτομο s έχει την τιμή ο στην ιδιότητα (ρόλο) p. Αν η ιδιότητα p είναι το στοιχείο rdf:type τότε είναι σαν να δηλώνουμε ότι ο πόρος s είναι στιγμιότυπο της κλάσης ο.

2.1.2.2 RDFS

Η γλώσσα αυτή παρέχει ένα επιπλέον λεξιλόγιο και επεκτείνει την εκφραστικότητα της RDF. Συγκεκριμένα μπορούμε να ορίσουμε νέες κλάσεις (rdfs:Class) και σχέσεις υπαγωγής ανάμεσά τους (rdfs:subClassOf) δημιουργώντας έτσι μια ιεραρχία κλάσεων. Επίσης μπορούμε να δημιουργήσουμε περιγραφές για τις ιδιότητες που χρησιμοποιούμε. Δηλαδή μπορούμε να περιγράψουμε σχέσεις υπαγωγής ανάμεσα σε δύο ιδιότητες (rdfs:subPropertyOf), να ορίσουμε το πεδίο ορισμού (rdfs:domain) και το πεδίο τιμών (rdfs:range) μιας ιδιότητας. Τέλος μας παρέχει τα στοιχεία rdfs:comment και rdfs:label με τα οποία μπορούμε να περιγράψουμε κάποιο πόρο με ελεύθερο κείμενο, δίνοντάς του έτσι κάποιο εναλλακτικό όνομα ή μια περιγραφή που εξηγεί τι είναι ο πόρος σε μορφή κατανοητή από τον άνθρωπο.

2.1.2.3 OWL (Web Ontology Language)

Όπως φαίνεται από τα προηγούμενα, η rdf και η rdfs παρέχουν μια πολύ βασική και περιορισμένη εκφραστικότητα. Αποτελούν όμως θεμέλιο για την ανάπτυξη πιο εκφραστικών γλωσσών ανωτέρου επιπέδου, όπως είναι οι γλώσσες του λογικού

επιπέδου και συγκεκριμένα της OWL[10][11]. Το πρότυπο της OWL καθορίζει τρείς διαφορετικές εκδοχές της, καθεμία με διαφορετική εκφραστικότητα, επομένως και υπολογιστική πολυπλοκότητα του reasoning.

- OWL Lite: Είναι μια υπογλώσσα της OWL DL και επομένως υποστηρίζει μόνο ορισμένους κατασκευαστές της. Σχεδιάστηκε για χρήστες που δεν επιθυμούν μεγάλη εκφραστικότητα αλλά γρήγορο reasoning. Στην πραγματικότητα όμως αποδείχτηκε ότι οι συντακτικοί περιορισμοί που εισάγει δεν μειώνουν την πραγματική εκφραστική δύναμη καθώς οι περισσότεροι κατασκευαστές της OWL DL μπορούν να προσομοιωθούν από περίπλοκες εκφράσεις της OWL Lite. Οπότε δυσκολεύει την κατασκευή οντολογιών με συντακτικούς περιορισμούς και δεν προσφέρει σε αντάλλαγμα καλύτερα υπολογιστικά χαρακτηριστικά. Αυτό έχει οδηγήσει στο να μην είναι ιδιαίτερα δημοφιλής επιλογή. Συγκριτικά με τις Περιγραφικές Λογικές η εκφραστικότητα που παρέχει είναι ισοδύναμη με την $\mathcal{SHIF}(\mathcal{D})$
- OWL DL: Σχεδιάστηκε να παρέχει την μέγιστη εκφραστικότητα, ενώ παράλληλα παραμένει υπολογιστικά πλήρης και αποφασίσιμη. Υποστηρίζει όλους τους κατασκευαστές της OWL και είναι εκφραστικά ισοδύναμη με την ΠΛ $\mathcal{SROIQ}(\mathcal{D})$
- OWL Full: Έχει διαφορετική σημασιολογία από την OWL Lite και την OWL DL και σχεδιάστηκε ώστε να είναι πλήρως συμβατή με το RDF Schema. Αυτό σημαίνει ότι δεν διαχωρίζει το σύνολο των κλάσεων, των ιδιοτήτων και των ατόμων. Επομένως ένα στοιχείο A μπορεί να χρησιμοποιείται ταυτόχρονα σαν κλάση, σαν άτομο και σαν ιδιότητα. Αυτή η μεταμοντελοποίηση που υποστηρίζει την καθιστά μη αποφασίσιμη και δεν υπάρχει κανένας αλγόριθμος που να μπορεί να χρησιμοποιηθεί για reasoning.

2.1.2.4 OWL2 EL

Όπως έχει ειπωθεί, ο συλλογισμός στην OWL είναι πολύ δύσκολος, αφού οι αλγόριθμοι συλλογιστικής έχουν πολύ μεγάλη πολυπλοκότητα. Για το λόγο αυτό έχουν κατασκευαστεί ειδικά βατά υποσύνολα της γλώσσας με καλές υπολογιστικές ιδιότητες. Η OWL EL[12][13] είναι ένα από αυτά. Συγκεκριμένα επιτρέπει πολυωνυμικούς αλγόριθμους για τις παρακάτω υπηρεσίες συλλογισμού: συνέπεια,

ταξινόμηση και έλεγχο στιγμιότυπων. Η OWL EL αντιστοιχεί στην ΠΛ $\mathcal{EL}$++. Τα κύρια χαρακτηριστικά της γλώσσας είναι ότι απαγορεύει τον κατασκευαστή καθολικού περιορισμού (universal restriction πχ $\forall$R.C) και της ένωσης ($\sqcup$) ενώ επιτρέπει τον πλήρη υπαρξιακό περιορισμό (existential restriction πχ $\exists$R.C) και την τομή ($\sqcap$). Η γλώσσα αυτή χρησιμοποιείται για την μοντελοποίηση μεγάλων αλλά μη σύνθετων εκφραστικά οντολογιών που αποτελούνται κυρίως από ορολογίες (terminological data). Έχει συχνή εφαρμογή σε επιστήμες της ζωής (life sciences). Μερικές μεγάλες οντολογίες που έχουν αναπτυχθεί σε EL είναι οι εξής:

- SNOMED CT (Systematized Nomenclature of Medizine, Clinical Terms)[14]: Είναι μια οντολογία για το σύστημα υγείας των Ηνωμένων πολιτειών, της Αγγλίας και μερικών άλλων χωρών. Αποτελείται από πάνω από μισό εκατομμύριο κλάσεις.
- NCI (Thesaurus of the National Cancer Institute)[15]: Οντολογία πάνω στην έρευνα κατά του καρκίνου. Περιέχει πάνω από 50000 κλάσεις.
- The Gene Ontology[16]: Είναι μια οντολογία για τα γονίδια (αποθηκεύει τα γονίδια και τα χαρακτηριστικά τους για το ανθρώπινο είδος όσο και για άλλα) και αποτελείται από 25000 κλάσεις.

Παρακάτω παρατίθενται οι σημαντικότεροι κατασκευαστές της OWL EL σε αντιστοιχία με τις αντίστοιχες εκφράσεις σε ΠΛ.

Κατασκευαστές σύνθετων εννοιών

ObjectIntersectionOf(C_1…C_n)	$C_1 \sqcap \ldots \sqcap C_n$
ObjectOneOf(o)	{o}
ObjectSomeValuesFrom(R C)	$\exists$R.C
ObjectHasValue(R o)	$\exists$R.{o}
ObjectExistsSelf(R)	$\exists$R.Self

Απαγορεύεται η χρήση των ακόλουθων στοιχείων της OWL : ObjectUnionOf, ObjectComplementOf, ObjectOneOf με πολλά στοιχεία, ObjectAllValuesFrom, ObjectMaxCardinality, ObjectMinCardinality, ObjectExactCardinality.

Αξιώματα Κλάσεων

SubClassOf(C D)	$C \sqsubseteq D$
EquivalentClasses($C_1 \ldots C_n$)	$C_1 \equiv \ldots \equiv C_n$
DisjointClasses($C_1 \ldots C_n$)	$C_i \sqcap C_j \sqsubseteq \bot$, $1 \leq i < j \leq n$

Απαγορεύεται η χρήση του DisjointUnion

Αξιώματα Ιδιοτήτων

SubObjectPropertyChain($R_1 \ldots R_n$)	$R_1 \circ \ldots \circ R_n$
SubObjectPropertyOf(R S)	$R \sqsubseteq S$
EquivalentObjectProperties($R_1 \ldots R_n$)	$R_1 \equiv \ldots \equiv R_n$
ObjectPropertyDomain(R C)	$\exists R.\top \sqsubseteq C$
ObjectPropertyRange(R C)	$\top \sqsubseteq \forall R.C$
TransitiveObjectProperty(R)	Tr(R) σημασιολογία: $\{(a^I,b^I), (b^I,c^I)\} \subseteq R^I \rightarrow (a^I,c^I) \in R^I$
ReflexiveObjectProperty(R)	$\top \sqsubseteq \exists R.Self$

Απαγορεύεται η χρήση των ακόλουθων στοιχείων της OWL: DisjointObjectProperties, IrreflexiveObjectProperty, InverseObjectProperties, FunctionalObjectProperty, SymmetricObjectProperty, AsymetricObjectProperty

Αξιώματα Οντολογίας

SameIndividual($o_1 \ldots o_n$)	$o_1 = \ldots = o_n$
DifferentIndividuals($o_1 \ldots o_n$)	$o_i \neq o_j$, $1 \leq i < j \leq n$
ClassAssertion(a C)	a:C
ObjectPropertyAssertion(R a b)	(a,b):R
NegativeObjectPropertyAssertion(R a b)	$\{a\} \sqsubseteq \neg(\exists R.\{b\})$ σημασιολογία: $(a^I,b^I) \notin R^I$

Τέλος παρουσιάζονται τα αξιώματα που αφορούν τύπους δεδομένων. Η σημασιολογία αυτών των αξιωμάτων εύκολα αντιστοιχίζεται με τα αξιώματα των προηγούμενων πινάκων. Η διαφορά εδώ είναι ότι οι ιδιότητες δεν είναι ιδιότητες αντικειμένων, αλλά ιδιότητες τύπων δεδομένων και δεν περιγράφουν σχέσεις μεταξύ δύο αντικειμένων, αλλά σχέσεις μεταξύ ενός αντικειμένου και ενός λεκτικού.

Τα ακόλουθα στοιχεία είναι διαθέσιμα: DataSomeValuesFrom, DataHasValue, DataOneOf, SubDataPropertyOf, EquivalentDataProperties, DataPropertyDomain, DataPropertyRange, DataPropertyAssertion, NegativeDataPropertyAssertion, FunctionalDataProperty.

2.2 Αυτόματη εξαγωγή όρων από κείμενο

Το κεφάλαιο αυτό αναλύει την μεθοδολογία ανάλυσης κειμένων φυσική γλώσσας και τους αλγορίθμους εξαγωγής αντιπροσωπευτικών όρων.

Η λειτουργία αυτή είναι πολύ σημαντική και πολύ χρήσιμη, καθώς όπως αναλύθηκε στο προηγούμενο κεφάλαιο η πλειοψηφία της πληροφορίας στο διαδίκτυο είναι αδόμητη και μη κατανοητή από τους υπολογιστές και το όνειρο του Σημασιολογικού Ιστού αργεί να ολοκληρωθεί. Είναι πολύ πιο εύκολο να ζητήσεις από τους απλούς χρήστες του διαδικτύου, να ανεβάσουν το περιεχόμενό τους στη γλώσσα που μιλούν, παρά να απαιτήσεις να το εκφράσουν σε μορφή οντολογίας και διασυνδεδεμένων δεδομένων. Έτσι, μεγάλο μέρος της επιστημονικής κοινότητας έχει στραφεί στην έρευνα πάνω στην επεξεργασία φυσικής γλώσσας (natural language processing) και στην αυτόματη εξαγωγή πληροφορίας (information extraction).

2.2.1 Επιμέρους στάδια επεξεργασίας φυσικής γλώσσας

Η επεξεργασία της φυσική γλώσσας χωρίζεται σε διάφορα μικρά στάδια και υπάρχουν πολλά επιμέρους εργαλεία, ελεύθερου λογισμικού και μη, που τα υλοποιούν (Stanford NLP Group[17], Apache OpenNLP[18], Dragon Toolkit[19]). Παρακάτω παρατίθενται τα πιο σημαντικά στάδια.

2.2.1.1 Stop word list

Είναι μια λίστα από λέξεις που συνήθως δεν προσδίδουν κάποια ιδιαίτερη πληροφορία στο κείμενο για αυτό και αφαιρούνται από αυτό, πριν αρχίσει η επεξεργασία του.

Η λίστα με τις λέξεις αυτές δεν είναι καθιερωμένη, έτσι διαφορετικές εφαρμογές μπορεί να χρησιμοποιούν διαφορετική λίστα και μερικές εφαρμογές να μην περιλαμβάνουν καθόλου αυτό το στάδιο. Μερικές τέτοιες λέξεις είναι για παράδειγμα οι ακόλουθες: *the, is, at, which, on, yours, would, when, whereas, lot, yes* κλπ. Μπορούμε να χρησιμοποιήσουμε μια γενική λίστα με stopwords ή να δημιουργήσουμε μια καινούργια για την συγκεκριμένη Συλλογή Κειμένων (εξετάζοντας τμήμα του corpus και εξάγοντάς τες με το χέρι). Εδώ πρέπει να βρεθεί ένας συμβιβασμός μεταξύ ακρίβειας και ανάκλησης. Όσο περισσότερες λέξεις συμπεριλάβουμε στην stoplist, τόσο βελτιώνουμε την ακρίβεια, αλλά χάνουμε στην ανάκληση, καθώς στα κείμενά μας μπορεί να περιέχονται όροι με απρόβλεπτες λέξεις που εξαιτίας της stoplist να απορρίπτονται. Ιδιαίτερα σε μηχανές αναζήτησης η αφαίρεση αυτών των λέξεων είναι ακόμα πιο σημαντική για να έχουμε καλά αποτελέσματα.

2.2.1.2 Λημματοποίηση (lemmatization) και stemming

Οι δύο αυτές διαδικασίες είναι παρόμοιες και χρησιμοποιούνται για να επιλύσουν το εξής σημαντικό πρόβλημα: μέσα σε ένα κείμενο είναι πολύ πιθανό να συναντήσουμε την ίδια λέξη, πολλές φορές, αλλά σε διαφορετική μορφή (λ.χ. σε διαφορετικό γένος, πτώση, αριθμό για ουσιαστικά και διαφορετικό χρόνο, έγκλιση, αριθμό, πρόσωπο για ρήματα). Είναι πολύ σημαντικό να μπορούμε να αναγνωρίσουμε σε όλες αυτές τις περιπτώσεις ότι πρόκειται για την ίδια λέξη. Επίσης, υπάρχει ειδική μέριμνα για τα κεφαλαία και τα πεζά γράμματα.

Αυτή η λειτουργία είναι πολύ σημαντική, καθώς μας βοηθά να μετρήσουμε σωστά την συχνότητα εμφάνισης της κάθε λέξης στο κείμενο και έτσι να την αξιολογήσουμε πόσο αντιπροσωπευτική είναι. Αν για παράδειγμα συναντήσουμε τη λέξη run με συχνότητα 5, την λέξη runs με συχνότητα 3, τη λέξη ran με 4 και την running με 3 τότε όλες αυτές οι λέξεις ανάγονται στην λέξη run με ολική συχνότητα 15. Εκτός από την μέτρηση της συχνότητας των λέξεων, η λειτουργία αυτή χρησιμοποιείται και στις

μηχανές αναζήτησης. Αν ο χρήστης μιας μηχανής αναζήτησης ψάξει την λέξη cars προφανώς θα περιμένει να του επιστραφούν κείμενα που περιέχουν κάποια από τις ακόλουθες λέξεις: car,cars,Car κλπ.

Περιγραφή του τρόπου λειτουργίας του λημματοποιητή.

Η λημματοποίηση είναι μια διαδικασία που δέχεται σαν είσοδο μια λέξη και επιστρέφει το λημματικό τύπο στον οποίο αντιστοιχεί. Για να είναι πιο αποτελεσματική η διαδικασία αυτή, εκτός από την λέξη παίρνει και σαν παράμετρο το μέρος του λόγου αυτής της λέξης (είναι ρήμα, ουσιαστικό, επίθετο ή επίρρημα). Το μέρος του λόγου το βρίσκουμε με άλλα εργαλεία (parsing and tagging) που θα εξηγηθούν αναλυτικά σε επόμενη παράγραφο. Δηλαδή λαμβάνει υπ' όψιν ολόκληρη την πρόταση μέσα στην οποία εμφανίζεται η λέξη καθώς και τους γραμματικούς κανόνες της γλώσσας που χρησιμοποιούμε. Έτσι, η διαδικασία γίνεται πιο πολύπλοκη αλλά ταυτόχρονα και πιο αποτελεσματική. Στη συνέχεια, ξέροντας τον τύπο της λέξης, την αναζητεί σε κατάλληλα λεξικά που διαθέτει ενσωματωμένα και βρίσκει το πρωτότυπο λήμμα στο οποία αντιστοιχεί.

Για παράδειγμα ο dragon lemmatizer κάνει λημματοποίηση της λέξης με την κλήση της παρακάτω μεθόδου:

String lemmatize(String word, int pos)

Όπου word είναι η λέξη προς λημματοποίηση και pos είναι ένας ακέραιος που δείχνει το μέρος του λόγου (ρήμα, επίθετο κλπ). Διαθέτει έναν φάκελο με ειδικά αρχεία στα οποία έχει αποθηκευμένα τα λεξικά. Έχει ένα λεξικό για τα ρήματα, ένα για τα ουσιαστικά, ένα για τα επίθετα και ένα για τα επιρρήματα. Έτσι, όταν του δώσουμε την λέξη ran με ετικέτα ρήματος, θα ψάξει στο λεξικό με τα ρήματα και θα βρει ότι αντιστοιχεί στο run.

Περιγραφή του τρόπου λειτουργίας του stemmer

Ο stemmer, σε αντίθεση με τον lemmatizer, δεν λαμβάνει υπ' όψιν του το πλαίσιο μέσα στο οποίο εμφανίζεται η λέξη. Αυτό τον καθιστά πιο γρήγορο και πιο εύκολο στην υλοποίηση. Το αντίτιμο είναι η μικρότερη ακρίβεια. Για παράδειγμα, δεν μπορεί να διακρίνει περιπτώσεις στις οποίες αλλάζει το νόημα της λέξεις ανάλογα με το μέρος του λόγου. Η λέξη "meeting" μπορεί να είναι το ρήμα "to meet" ή να είναι το ουσιαστικό "meeting". Ο stemmer δίνει και στις δύο φορές το ίδιο αποτέλεσμα "meet", ενώ ο lemmatizer διαφορετικό.

Πρόταση: I have a meeting tomorrow with my colleagues.

Stemmer: I have a meet tomorrow with my colleague.

Lemmatizer: I have a meeting tomorrow with my colleague.

Πρόταση: I am meeting my colleagues tomorrow.

Stemmer: I be meet my colleague tomorrow.

Lemmatizer: I be meet my colleague tomorrow.

Ο πιο κοινός αλγόριθμος για την υλοποίηση του stemming στα Αγγλικά είναι ο αλγόριθμος του Porter[20]. Εδώ δεν χρησιμοποιούμε αναζητήσεις σε λεξικά αλλά κυρίως ευριστικές μεθόδους που προσπαθούν να κόψουν τις καταλήξεις των λέξεων και πολλές φορές κόβουν και τα παραγωγικά επιθήματα. Ο αλγόριθμος είναι αρκετά μεγάλος αλλά η γενική του φιλοσοφία είναι η εξής:

Αποτελείται από 5 διαφορετικά στάδια το ένα μετά το άλλο. Κάθε στάδιο έχει ένα σύνολο κανόνων από το οποίο επιλέγουμε ποιόν θα εφαρμόσουμε. Υπάρχουν διάφορες τεχνικές επιλογής κανόνα, μια από αυτές είναι η επιλογή εκείνου που εφαρμόζεται στο μεγαλύτερο επίθημα. Για παράδειγμα, στο πρώτο στάδιο έχουμε το παρακάτω σύνολο κανόνων με την προαναφερθείσα τεχνική επιλογής.

Rule			Example		
SSES	→	SS	caresses	→	caress
IES	→	I	ponies	→	poni
SS	→	SS	caress	→	caress
S	→		cats	→	cat

Εικόνα 2.2-1 ο αλγόριθμος του Porter

Ακολουθεί ένα παράδειγμα εφαρμογής του αλγορίθμου του Porter σε μία πρόταση:

Πρόταση: Such an analysis can reveal features that are not easily visible.

Stemming: Such an analys can reve feature that ar not eas vis.

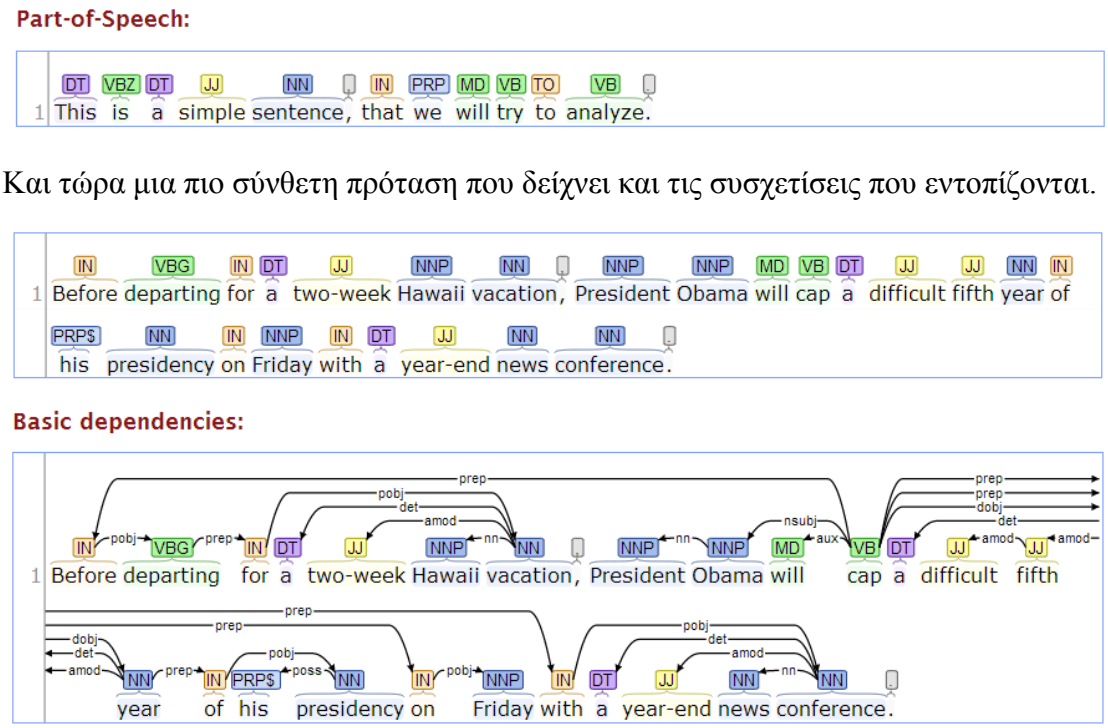
2.2.1.3 Part of Speech Tagging

Είναι ένα από τα πιο σημαντικά στάδια της επεξεργασίας κειμένων. Σε αυτό το στάδιο πρέπει να αναγνωριστεί κάθε λέξη του κειμένου, τι μέρος του λόγου είναι

(ουσιαστικό, ρήμα, επίθετο, επίρρημα, πρόθεση κλπ). Για να γίνει αυτή η επεξεργασία πρέπει να έχουν προηγηθεί δύο μικρά στάδια

- Sentence Detection: Στο στάδιο αυτό το κείμενο χωρίζεται σε προτάσεις. Αυτό γίνεται πολύ εύκολα απλά αναγνωρίζοντας τα σημεία στίξης. Έτσι μια πρόταση ορίζεται ως η ακολουθία χαρακτήρων μεταξύ δύο σημείων στίξης. Φυσικά γίνονται ορισμένοι έλεγχοι και δεν χωρίζει απλά όπου συναντά τον χαρακτήρα "." διότι τότε εκφράσεις όπως "Mr. Smith" ή "Elsevier N.V." θα χωριζόντουσαν λανθασμένα σε προτάσεις.
- Tokenization: Σε αυτό το στάδιο παίρνουμε μία-μία πρόταση και την χωρίζουμε στις επιμέρους λέξεις.

Ακολουθούν παραδείγματα POS tagging[1]



Εικόνα 2.2-2 Part Of Speech Tagging

2.2.1.4 Chunking

Μέχρι τώρα έχουμε κάνει τα εξής: χωρίσαμε το κείμενό μας σε προτάσεις, αφαιρέσαμε τις ασήμαντες λέξεις και διακρίναμε τι μέρος του λόγου είναι κάθε λέξη. ·Τώρα για να συνεχίσουμε και να κάνουμε εξαγωγή όρων (terms) και ονοματισμένων

[1] Τα παραδείγματα αυτά έγιναν με χρήση του εργαλείου Stanford CoreNLP

οντοτήτων (named entities) πρέπει να έχουμε παραπάνω πληροφορία που να αφορά την δομή της πρότασης. Κάτι τέτοιο μας το προσφέρει η συντακτική ανάλυση (parsing). Η συντακτική ανάλυση είναι μια πολύπλοκη, επομένως και χρονοβόρα διαδικασία που προσφέρει πλήρη γνώση της δομής της πρότασης. Στην συγκεκριμένη όμως περίπτωση δεν την χρειαζόμαστε όλη αυτή την πληροφορία και μπορούμε να αρκεστούμε σε απλή ομαδοποίηση των λέξεων (chunking) και συγκεκριμένα των ονοματικών φράσεων. Η ακόλουθη πρόταση ομαδοποιείται ως εξής:

Πρόταση: Rockwell International Corp.'s Tulsa unit said it signed a tentative agreement

Chunking: [NP Rockwell_NNP International_NNP Corp._NNP] [NP 's_POS Tulsa_NNP unit_NN] [VP said_VBD] [NP it_PRP] [VP signed_VBD] [NP a_DT tentative_JJ agreement_NN]

Βλέπουμε ότι οι ονοματικές φράσεις [Rockwell International Corp.] και [a tentative agreement] έχουν ομαδοποιηθεί.

2.2.2 Εύρεση αντιπροσωπευτικών όρων Συλλογής Κειμένων

Όπως ειπώθηκε και σε προηγούμενο κεφάλαιο στόχος μας είναι να επεξεργαστούμε μια Συλλογή Κειμένων και να συλλέξουμε τους πιο σημαντικούς όρους από αυτά. Όροι που συνοψίζουν το περιεχόμενο των κειμένων, το νόημα τους και τις έννοιες που περιλαμβάνουν. Έχει γίνει πολύ έρευνα πάνω σε αυτό το κομμάτι, για το πώς θα αποφασίσουμε ποιοι είναι αντιπροσωπευτικοί όροι, και παρακάτω παρατίθενται οι πιο σημαντικοί και διαδεδομένοι αλγόριθμοι. Όλοι τους χρησιμοποιούν για την λήψη αυτής της απόφασης κάποια αριθμητικά και στατιστικά δεδομένα που συλλέχθηκαν κατά την επεξεργασία των κειμένων. Παρακάτω περιγράφονται τι δεδομένα συλλέγονται και με ποιο τρόπο.

2.2.2.1 Συλλογή αριθμητικών και στατιστικών δεδομένων

Ο αλγόριθμος επεξεργασίας κειμένων μοιάζει κάπως έτσι:

```
for each document in the corpus do{
   split document into sentences;
   for each sentence do{
      String[] tokens = split_sentence_into_tokens;
      String[] posTagging =
            for_every_token_find_its_part_of_speech
      String[] candidates = chunkNounPhrases(posTagging)
      removeWeirdCharactersFromCandidates;
      candidatesToLowerCase;
      removeStopWordsFromCandidates;
      lemmatizeCandidates;
      storeCandidates;
   }
}
```

Στον αλγόριθμο που παρατίθεται παραπάνω φαίνονται τα διάφορα στάδια επεξεργασίας και πώς ακολουθεί το ένα μετά το άλλο. Τώρα θα εξηγήσουμε αναλυτικά το στάδιο "storeCandidates" που είναι η αποθήκευση της αριθμητικής και στατιστικής πληροφορίας που μας ενδιαφέρει.

Στο τέλος αυτής της διαδικασίας έχουμε κρατήσει όλους τους όρους (ονοματικές φράσεις) που συναντήσαμε στα κείμενα. Για κάθε όρο κρατάμε επιπλέον την συχνότητα εμφάνισής του (πλήθος εμφανίσεων), καθώς και μια λίστα με τα κείμενα όπου τον βρήκαμε. Επίσης, κρατάμε και την αντίστροφη πληροφορία, δηλαδή ποιους όρους περιέχει το κάθε κείμενο. Με αυτές τις βασικές πληροφορίες, μπορούμε να εφαρμόσουμε τους αλγόριθμους της επόμενης παραγράφου.

Γενικά υπάρχουν δύο κριτήρια αξιολόγησης μιας μεθόδου εξαγωγής όρων:

- Ακρίβεια (precision): ως ακρίβεια ορίζεται η πιθανότητα ένας όρος που εξάγεται από τον αλγόριθμό μας να είναι όντως σημαντικός όρος (όρος του επιστημονικού κλάδου που εξετάζει η Συλλογή Κειμένων).
- Ανάκληση (recall): τι ποσοστό των όρων που περιλαμβάνονται στη Συλλογή Κειμένων, είναι σε θέση να αναγνωρίσει ο αλγόριθμός μας

Αυτά τα δύο κριτήρια είναι πολλές φορές αντικρουόμενα που σημαίνει ότι η βελτίωση του ενός επιφέρει την μείωση του άλλου.

2.2.2.2 Αλγόριθμοι επιλογής όρων

Οι αλγόριθμοι αυτοί, λαμβάνοντας υπ' όψιν τα στατιστικά δεδομένα που έχουμε κρατήσει από την επεξεργασία κειμένου, πρέπει να αναθέσουν σε κάθε όρο ένα νούμερο (σκορ) που δείχνει πόσο σημαντικός είναι (term confidence). Αυτοί είναι οι πιο σημαντικοί και ευρέως χρησιμοποιούμενοι αλγόριθμοι.

Σημείωση: Οι ορισμοί αυτών των αλγορίθμων μπορεί να διαφέρουν λιγάκι από περίπτωση σε περίπτωση και εξαρτώνται από το που θέλουμε να τους εφαρμόσουμε. Για παράδειγμα, πολλοί από αυτούς τους αλγόριθμους εφαρμόζονται στις μηχανές αναζήτησης. Στις μηχανές αυτές, η λογική είναι η εξής: πάρε την συμβολοσειρά που έγραψε ο χρήστης σε ένα πεδίο, χώρισε την σε λέξεις και βρες ποια κείμενα είναι πιο σχετικά με αυτές τις λέξεις. Οπότε εκεί έχει νόημα η εύρεση της συχνότητας εμφάνισης της λέξης σε κάθε κείμενο. Αν συμβολίσουμε τον όρο t και το κείμενο d, τότε η συχνότητα του όρου θα ορίζετε ως $tf_{t,d}$ = πλήθος εμφανίσεων του όρου στο κείμενο αυτό. Στην δικιά μας όμως περίπτωση, όπου εξετάζουμε ποιοι όροι είναι σημαντικοί και αντιπροσωπευτικοί για όλη την Συλλογή Κειμένων, ως συχνότητα εμφάνισης του όρου λαμβάνεται η συνολική από όλα τα κείμενα.

Simple Term Frequency

Αυτός ο αλγόριθμος είναι ο πιο απλός και διαισθητικός καθώς σε κάθε όρο το σκορ που του αναθέτει είναι η συχνότητα εμφάνισής του. Δηλαδή αν βρούμε τον όρο "expensive car" 30 φορές στη Συλλογή Κειμένων, τότε του αναθέτουμε το σκορ 30. Επομένως, όσο πιο πολλές φορές συναντούμε έναν όρο, τόσο πιο σημαντικό τον θεωρούμε.

Average Term Frequency in Corpus

Το σκορ του κάθε όρου είναι η ολική συχνότητα εμφάνισής του στη Συλλογή Κειμένων προς το πλήθος των κειμένων στο οποίο τον συναντούμε.

$$\frac{\text{CorpusFrequency}}{\text{DocumentFrequency}}$$

Αν για παράδειγμα τον όρο "expensive car" τον συναντούμε 30 φορές στη Συλλογή Κειμένων και ο όρος αυτός εμφανίζεται σε 3 διαφορετικά κείμενα (στα υπόλοιπα κείμενα δεν εμφανίζεται καθόλου) τότε το σκορ του είναι 30/3 = 10

TF-IDF (Term Frequency- Inverse Document Frequency)

Μαθηματικός ορισμός:

$$\text{tf} = \frac{\text{πλήθος\_εμφανίσεων\_του\_όρου\_στη\_Συλλογή\_Κειμένων}}{\text{συνολικό\_πλήθος\_εμφανίσεων\_όλων\_των\_όρων\_στη\_Συλλογή\_Κειμένων}}$$

$$\text{df} = \text{πλήθος\_κειμένων\_όπου\_εμφανίζεται\_αυτός\_ο\_όρος}$$

$$\text{idf} = \log\left(\frac{N}{df}\right), \text{ όπου N είναι το πλήθος των κειμένων του Corpus}$$

$$\text{score} = \text{tf} * \text{idf}$$

Στον αλγόριθμο αυτό το σκορ ενός όρου αυξάνεται με τη συχνότητα εμφάνισής του στη Συλλογή Κειμένων, αλλά μειώνεται με την συχνότητα εμφάνισής του σε διαφορετικά κείμενα. Έτσι μπορούμε να αντιμετωπίσουμε το γεγονός ότι μερικές λέξεις εμφανίζονται γενικά πιο συχνά από κάποιες άλλες και στην ουσία δεν προσδίδουν κάποιο ιδιαίτερο νόημα (μπορεί να χρησιμοποιηθεί και για stop words filtering) [21].

Weirdness

Αυτός ο αλγόριθμος εξαγωγής όρων, προτάθηκε για πρώτη φορά στο (Ahmad, Gillam, & Tostevin 1999)[22] και λειτουργεί ως εξής:

Για να βρει τους αντιπροσωπευτικούς όρους μιας Συλλογής Κειμένων, μετράει την συχνότητα εμφάνισης του κάθε όρου στη συλλογή αυτή και μετά την συγκρίνει με τη συχνότητα εμφάνισης σε ένα Corpus αναφοράς. Δηλαδή για να λειτουργήσει αυτός ο αλγόριθμος, εκτός από την εξεταζόμενη Συλλογή Κειμένων (domain specific) χρειάζεται άλλη μια Συλλογή Κειμένων που θεωρείται γενικού περιεχομένου (general corpus, κείμενα από διάφορους επιστημονικούς και μη κλάδους). Αν η συχνότητα εμφάνισης ενός όρου και στις δύο Συλλογές Κειμένων είναι περίπου ίδια, τότε ο όρος θεωρείται ότι δεν είναι αντιπροσωπευτικός. Αντίθετα, αν ένας όρος εμφανίζεται στο domain specific corpus σε πολύ μεγαλύτερη συχνότητα απ’ ότι στο general corpus, τότε θεωρείται σημαντικός. Για παράδειγμα, έστω ότι εξετάζουμε μια Συλλογή Κειμένων πάνω στην Νανοτεχνολογία. Οι λέξεις νανοσωματίδια και νανοσωλήνες μπορεί να συναντιούνται σε συχνότητα 0,3%, ενώ στο general Corpus να συναντιούνται σε συχνότητα 0,001%. Το σκορ (confidence) του όρου αυτού θα είναι 0,3/0,001 = 300 και επομένως θεωρείται σημαντικός και συμπεριλαμβάνεται στα αποτελέσματα. Αντίθετα, όροι σαν αυτούς: “the”, ”and”, “to”, ”but”, “which” δηλαδή αντωνυμίες, σύνδεσμοι, προθέσεις κλπ συναντιόνται πολύ συχνά και στις δύο συλλογές και χωρίς μεγάλη διαφορά στην συχνότητα. Έτσι όλοι αυτοί οι όροι θα

έχουν σκορ κοντά στην μονάδα και επομένως σωστά δεν θα συμπεριλαμβάνονται στα αποτελέσματα.

Μαθηματικά το σκορ των όρων ορίζεται ως εξής:

$$\text{Weirdness} = \frac{\frac{ws}{ts}}{\frac{wg}{tg}}$$

όπου w_s = συχνότητα όρου στο specialist language corpus

w_g = συχνότητα όρου στο general language corpus

t_s = συνολικό πλήθος λέξεων στο specialist language corpus

t_g = συνολικό πλήθος λέξεων στο general language corpus

Ως Συλλογή Κειμένων γενικού περιεχομένου συχνά χρησιμοποιείται η British National Corpus (BNC)[23]. Πρόκειται για μια συλλογή γραπτού και προφορικού λόγου, 100 εκατομμυρίων λέξεων, από διάφορες πηγές, σχεδιασμένη να αντιπροσωπεύει τα σύγχρονα αγγλικά. Πηγές γραπτού λόγου είναι για παράδειγμα: εφημερίδες, περιοδικά, βιβλία, επιστημονικά κείμενα, γράμματα, σχολικά και πανεπιστημιακά βιβλία.

Υλοποιήσεις τέτοιων αλγορίθμων από διάφορα εργαλεία, προφανώς δεν αποθηκεύουν στην μνήμη τους όλη την BNC συλλογή για να κάνουν τις συγκρίσεις, αλλά χρησιμοποιούν μόνο ένα αρχείου που έχει μια λίστα με όλες τις λέξεις και την συχνότητα εμφάνισής τους. Τα αρχεία αυτά έχουν τη μορφή (πλήθος εμφανίσεων του όρου - όρος)

53868 government
51439 world
35596 national
35553 head
35147 party
35102 money
34788 company

Ο αλγόριθμος αυτός, χρησιμοποιήθηκε για πρώτη φορά για την κατασκευή ενός πρωτότυπου συστήματος ανάκτησης αρχείων (WILDER), που συμμετείχε στον διαγωνισμό U.S.Text Retrieval Competition (TREC)[24]. Το σύστημα στα πλαίσια του διαγωνισμού, λειτουργούσε ως εξής:

Διαχειριζόταν (είχε αποθηκευμένη στη μνήμη του) μια πολύ μεγάλη Συλλογή Κειμένων (την TREC-8 Συλλογή Κειμένων) και σύμφωνα με αυτήν έπρεπε να απαντά στα ερωτήματα των χρηστών. Τα ερωτήματα των χρηστών ήταν απλές συμβολοσειρές που δείχνουν τι θέλουν να αναζητήσουν (keyword search). Σύμφωνα με τα ερωτήματα των χρηστών έπρεπε να προτείνουν ποια κείμενα από όλη την συλλογή είναι πιο σχετικά με την αναζήτηση.

C-value

Αυτός ο αλγόριθμος προτάθηκε από το [25] και έχει σαν σκοπό την βελτίωση της ακρίβειας των εξαγόμενων όρων σε σχέση με τον απλό αλγόριθμο μέτρησης συχνότητας. Κυρίως δίνει έμφαση και προσπαθεί να βελτιώσει την εύρεση των εμφωλευμένων όρων (nested terms).

Ο αλγόριθμος αυτός συνδυάζει γλωσσολογική και στατιστική πληροφορία για την επιλογή των όρων, με έμφαση όμως στα στατιστικά δεδομένα. Η διαδικασία που ακολουθείται είναι η εξής:

Αρχικά γίνεται tagging της κάθε λέξη της Συλλογής Κειμένων. Μετά εφαρμόζουμε ένα γλωσσολογικό φίλτρο επιλογής όρων (linguistic filter). Θα ήταν επιθυμητό να μπορούμε να εξάγουμε όλους τους τύπους όρων (δηλαδή ονοματικές φράσεις, επιθετικές φράσεις, ρηματικές φράσεις), δηλαδή να μην εφαρμόζουμε κανένα γλωσσολογικό φίλτρο, όμως τότε μειώνεται πολύ η ακρίβεια των αποτελεσμάτων (εμφανίζονται όροι όπως το “of the”, “is a” κλπ). Επίσης, γνωρίζουμε ότι η πλειοψηφία των όρων αποτελείται από ουσιαστικά, επίθετα και μερικές φορές προθέσεις. Έτσι επιλέγεται φίλτρο που να αποδέχεται αυτούς τους τύπους όρων. Ακόμα όμως και τώρα, που αποφασίσαμε να χρησιμοποιήσουμε φίλτρα ονοματικών φράσεων, υπάρχει δυνατότητα να διαλέξουμε φίλτρα με διαφορετική εκφραστικότητα και επομένως διαφορετική ακρίβεια και ανάκληση. Για παράδειγμα τρία δυνατά φίλτρα είναι τα εξής:

1. Noun^{+} Noun

2. $(\text{Adj} \mid \text{Noun})^{+}$ Noun

3. $((\text{Adj} \mid \text{Noun})^{+} \mid ((\text{Adj} \mid \text{Noun})^{*}$

   $(\text{NounPrep})^{?}) (\text{Adj} \mid \text{Noun})^{*})$ Noun

Από τα οποία το πρώτο είναι το πιο αυστηρό (closed filter), ενώ το τελευταίο το πιο εκφραστικό (open filter).

Τώρα θα εξετάσουμε πώς υπολογίζεται το σκορ του κάθε όρου. Έστω ότι ο εξεταζόμενος όρος είναι ο a, τότε το σκορ του συμβολίζεται με termhood(a) (με πόση σιγουριά ισχυριζόμαστε ότι όντως είναι όρος). Υπάρχουν τέσσερα κριτήρια που λαμβάνονται υπ' όψιν στον υπολογισμό αυτής της τιμής:

1. Η ολική συχνότητα εμφάνισης του υποψήφιου όρου a στη Συλλογή Κειμένων.
2. Η συχνότητα εμφάνισης του όρου a ως υποσυμβολοσειρά ενός άλλου μακρύτερου υποψήφιου όρου. Για παράδειγμα ο όρος a = "real time", εμφανίζεται μέσα στη Συλλογή Κειμένων ως υποσυμβολοσειρά των επίσης υποψήφιων όρων: "real time clock", "real time expert system", "real time image generation".
3. Το πλήθος αυτών των μακρύτερων υποψήφιων όρων που τον περιέχουν.
4. Το μήκος του υποψήφιου όρου a (πλήθος λέξεων του όρου).

Ερμηνεία των κριτηρίων:

Η ορθότητα του πρώτου κριτηρίου είναι προφανής και έχει εξηγηθεί πολλές φορές. Το δεύτερο κριτήριο έρχεται για να βελτιώσει την ανίχνευση των εμφωλευμένων όρων. Για παράδειγμα, σε ένα corpus οφθαλμολογίας η συμβολοσειρά "soft contact lens" αποτελεί όρο. Αν η συμβολοσειρά αυτή συναντάται συχνά, τότε ο αλγόριθμος συχνότητας θα εξάγει ορθά αυτόν τον όρο, όμως θα εξάγει και τις υποσυμβολοσειρές "soft contact" και "contact lens", καθώς και αυτοί οι όροι συναντώνται σε συχνότητα ίδια ή μεγαλύτερη και περνούν το γλωσσολογικό φίλτρο. Όμως η συμβολοσειρά "soft contact" δεν αποτελεί όρο στην πραγματικότητα και λανθασμένα τον εξάγαμε. Μια λύση σε αυτό το πρόβλημα θα ήταν να υπολογίζαμε το σκορ (termhood) ενός όρου μόνο από τις φορές που εμφανίζεται από μόνος του και όχι ως υποσυμβολοσειρά. Δηλαδή ως εξής: συχνότητα εμφάνισής του πλην την συχνότητα εμφάνισής του σαν υποσυμβολοσειρά μεγαλύτερων υποψήφιων όρων.

$$\text{termhood}(\alpha) = f(\alpha) - \sum_{b \in T\alpha} f(b)$$

Όπου

α είναι ο υποψήφιος όρος,

f(.) είναι η συχνότητα εμφάνισης του όρου στη Συλλογή Κειμένων

T_α είναι το σετ των υποψήφιων όρων που περιέχουν το α σαν υποσυμβολοσειρά.

b ένας τέτοιος όρος (που περιέχει το α σαν υποσυμβολοσειρά)

Όμως ακόμα και τώρα δεν έχουμε λύσει όλα τα προβλήματα της εξαγωγής εμφωλευμένων όρων. Πολλές φορές μπορεί ένας όρος να εμφανίζεται μόνο σαν υποσυμβολοσειρά σε μια Συλλογή Κειμένων και σχεδόν ποτέ μόνος του. Σύμφωνα με τον προηγούμενο τύπο, δεν θα τον αναγνωρίζουμε σαν όρο, ενώ στην πραγματικότητα είναι. Για παράδειγμα, σε μια Συλλογή Κειμένων μπορεί να συναντούμε τους όρους "real time" και "floating point" μόνο σαν υποσυμβολοσειρές και ποτέ μόνες τους

real time clock
real time expertsystem
real time image generation
real time output
real time systems

floating point arithmetic
floating point constant
floating point operation
floating point routine

Εδώ, πρέπει να μπορούμε να διακρίνουμε ποιες από τις υποσυμβολοσειρές είναι όροι και ποιες όχι. Όλες οι υποσυμβολοσειρές της δεύτερης στήλης είναι οι εξής: "point arithmetic", "point constant", "point operation", "point routine" και "floating point". Από αυτές μόνο η "floating point" είναι όντως όρος. Αυτό που μπορούμε να χρησιμοποιήσουμε σαν κριτήριο, για να μπορέσουμε να διαχωρίσουμε την συμβολοσειρά "floating point" από τις υπόλοιπες, είναι ότι αυτή συναντάται σαν υποσυμβολοσειρά πολλών διαφορετικών συμβολοσειρών, ενώ οι άλλες συμβολοσειρές (μη όροι) συναντώνται μόνο σαν υποσυμβολοσειρές ενός μόνο μακρύτερου υποψήφιου όρου. Δηλαδή όσο αυξάνεται ο αριθμός των διαφορετικών συμβολοσειρών που περιέχουν έναν υποψήφιο όρο σαν εμφωλευμένο, τόσο αυξάνεται και η πεποίθησή μας ότι αποτελεί ανεξάρτητο όρο από μόνος του (τρίτο κριτήριο).

Τέλος, το τέταρτο κριτήριου, που είναι το μήκος ενός όρου (δηλαδή το πλήθος λέξεων που τον αποτελεί), εξηγείται ως εξής: μια συμβολοσειρά μεγάλου μήκους είναι λιγότερο πιθανό να εμφανιστεί f φορές σε μια Συλλογή Κειμένων σε σχέση με μια συμβολοσειρά μικρότερου μήκους. Για να εξισορροπήσουμε αυτή την διαφορά, πολλαπλασιάζουμε το σκορ με μια τιμή που είναι ανάλογη με το μήκος του όρου.

Ο τελικός τύπος υπολογισμού είναι:

$$C-value(\alpha)=\begin{cases}\log_2|\alpha|\cdot f(\alpha) & \alpha \text{ is not nested}\\ \log_2|\alpha|\left(f(\alpha)-\frac{1}{P(T_\alpha)}\sum_{b\in T\alpha} f(b)\right), & \text{otherwise}\end{cases}$$

Όπου

α είναι ο υποψήφιος όρος,

f(.) είναι η συχνότητα εμφάνισης του όρου στη Συλλογή Κειμένων

T_α είναι το σετ των υποψήφιων όρων που περιέχουν το α σαν υποσυμβολοσειρά πχ {“real time clock”, “real time expert system”, “real time image generation”}

$P(T_\alpha)$ είναι το πλήθος του συνόλου T_α.

$\log_2|\alpha|$ ο λογάριθμος του μήκους του υποψήφιου όρου (|α| = πλήθος λέξεων του όρου)

NC-value

Αυτή η μέθοδος είναι επέκταση της μεθόδου C-value και χρησιμοποιεί πληροφορία εξαγόμενη από τα συμφραζόμενα, για να αυξήσει την ακρίβειά της στην εξαγωγή όρων.

Πολλές φορές, για να καταλάβουμε την σημασία μίας λέξης, είναι χρήσιμο να κοιτάξουμε το πλαίσιο μέσα στο οποίο εμφανίστηκε (ουσιαστικά, ρήματα και επίθετα στην γειτονιά του όρου). Μάλιστα, σε επιστημονικά κείμενα, είναι συχνά γεγονός ότι συγκεκριμένες λέξεις τείνουν να εμφανίζονται κοντά σε όρους (είτε προηγούνται, είτε έπονται του όρου). Αυτοί οι όροι λέγονται term context words. Για παράδειγμα, σε μια Συλλογή Κειμένων ιατρικού περιεχομένου, επίθετα όπως το “consistent” και το “present”, ουσιαστικά όπως το “compound” και το “layer”, ρήματα όπως το “present”,”showing” και “containing”, εμφανίζονται πολύ συχνά κοντά σε ιατρικούς όρους. Έτσι, αν βρούμε μία από αυτές τις λέξεις κοντά σε έναν υποψήφιο όρο, τότε αυξάνεται η βεβαιότητά μας ότι αποτελεί πραγματικό όρο της Συλλογής Κειμένων. Παραδείγματα:

1. nouns (compound cellular naevus),
2. adjectives (blood vessels are present), and
3. verbs (composed of basaloid papillae).

Κριτήριο για να θεωρήσουμε μια λέξη σαν term context word είναι το πλήθος των όρων με τους οποίους εμφανίζεται. Σε όσο περισσότερους όρους εμφανίζεται δίπλα, τόσο πιο πιθανό είναι αυτή η λέξη να σχετίζεται με την εμφάνιση όρων. Οι λέξεις term context words που βρίσκουμε είναι για το συγκεκριμένο corpus που εξετάζουμε και μεταβάλλονται ανάλογα με το τομέα που μελετούμε. Αν μια λέξη είναι term context word ή όχι καθορίζεται από την τιμή που αποκτά από τον επόμενο τύπο:

$$\text{weight}(w) = \frac{t(w)}{n} \qquad (1)$$

όπου

w	είναι η λέξη που εξετάζουμε αν είναι term context word
weight(w)	το βάρος που αναθέτουμε στη λέξη αυτή
t(w)	πλήθος όρων που εμφανίζεται γειτονικά
n	το συνολικό πλήθος όρων που χρησιμοποιήθηκε για την εξαγωγή των term context words

Οπότε η μέθοδος NC-value είναι η εξής:

1. Εφαρμόζουμε πρώτα την μέθοδο C-value στη Συλλογή Κειμένων. Η έξοδος αυτής της μεθόδου είναι μια λίστα από υποψήφιους όρους, διατεταγμένη ως προς την τιμή C-value.
2. Μετά εξάγουμε τις term context words και τα βάρη τους. Για να γίνει αυτή η διαδικασία, πρέπει να έχουμε ορισμένους όρους και με βάση αυτούς να εφαρμόσουμε τον προηγούμενο τύπο (1). Οι όροι αυτοί πρέπει να είναι πραγματικοί όροι της Συλλογής Κειμένων και όχι λανθασμένες υποθέσεις, για να έχουμε καλά αποτελέσματα. Για την πλήρη αυτοματοποίηση όμως της διαδικασίας, δεν χρησιμοποιείται κάποια εξωτερική πηγή (πχ λεξικό ή έλεγχος από άνθρωπο) που να βεβαιώνει την ορθότητα των όρων, αλλά χρησιμοποιούνται οι όροι με την υψηλότερη βαθμολογία C-value. Δηλαδή παίρνουμε ένα μέρος (πχ 10 ή 20%) των όρων με την υψηλότερη βαθμολογία C-value και εφαρμόζουμε τον τύπο (1). Έτσι, γνωρίζουμε ότι η συντριπτική πλειοψηφία των όρων που χρησιμοποιούνται είναι ορθοί, καθώς όσο μεγαλύτερη τιμή C-value έχουν, τόσο πιο πιθανό είναι να αποτελούν πραγματικούς όρους του Corpus.

3. Το τρίτο στάδιο περιλαμβάνει ανακατάταξη της λίστας των υποψήφιων όρων, που παράχθηκε στο στάδιο 1, χρησιμοποιώντας τις term context words που βρέθηκαν στο 2^{o} στάδιο. Με αυτή την ανακατάταξη, χρησιμοποιώντας πληροφορία από τα συμφραζόμενα, πετυχαίνουμε το εξής: οι πραγματικοί όροι εμφανίζονται πιο κοντά στην αρχή της λίστας από ότι πριν. Δηλαδή η συγκέντρωση των πραγματικών όρων στην αρχή της λίστας αυξάνεται, ενώ στο τέλος της λίστας μειώνεται. Η ανακατάταξη αυτή γίνεται σύμφωνα με την τιμή NC-value που ανατίθεται σε κάθε υποψήφιο όρο. Ο τύπος υπολογισμού είναι:

$$\mathrm{NCvalue}(\alpha) = 0.8 \cdot \mathrm{Cvalue}(\alpha) + 0.2 \sum_{b \in C\alpha} f_{\alpha}(b)\mathrm{weight}(b)$$

Όπου

α είναι ο υποψήφιος όρος

C_{α} είναι ένα σετ με τις διαφορετικές συμφραζομένες λέξεις (context words) που έχουν εντοπιστεί στην γειτονιά του υποψήφιου όρου α

b είναι μια λέξη από το σύνολο C_{α}

$f_{\alpha}(b)$ είναι η συχνότητα που η context word b εμφανίζεται στη γειτονιά του όρου α

Ο πρώτος όρος είναι από την μέθοδο C-value και ο δεύτερος από την πληροφορία των συμφραζομένων. Για παράδειγμα, αν μια λέξη w εμφανίζεται 10 φορές με τη συμφραζόμενη λέξη (context word) c_1 , 20 φορές με την λέξη c_2 και 30 φορές με την λέξη c_3 και το βάρος της term context word c1 είναι w_1, της c_2 είναι w2 και της c_3 είναι w_3 τότε η NC-value τιμή θα είναι

NCvalue(w) = 0.8*Cvalue(w) + 0.2*(10w1 + 20w2 + 30w3)

3

Ανάλυση Απαιτήσεων Συστήματος

Στα πλαίσια αυτής της διπλωματικής εργασίας, σχεδιάστηκε και υλοποιήθηκε σύστημα που βοηθά τον χρήστη στην ανάπτυξη και εξειδίκευση οντολογιών.

Οι λειτουργικές απαιτήσεις του συστήματος είναι:

- Υποβοήθηση του χρήστη στον ορισμό νέων εξειδικευμένων κλάσεων της οντολογίας. Αυτό επιτυγχάνεται με την αυτόματη άντληση γνώσης από αδόμητες ή ημιδομημένες περιγραφές αντικειμένων που είναι καταχωρημένες σε βάσεις δεδομένων. Στην συνέχεια, προτείνεται αυτή τη γνώση στο χρήστη για τον ορισμό των νέων κλάσεων.
    - Εξαγωγή ονοματικών φράσεων και επιθέτων, εφαρμόζοντας τεχνολογίες επεξεργασίας φυσικής γλώσσας και αυτόματης εξαγωγής όρων σε Συλλογές Κειμένων.
    - Καταγραφή στατιστικής πληροφορίας για τους όρους αυτούς, καθώς και των θέσεων εμφάνισής τους (δυνατότητα ανάκτησης των προτάσεων μέσα στις οποίες εμφανίζεται ο όρος).
    - Υποδομή για άντληση επιπλέον γνώσης μέσω της εκτέλεση επιπρόσθετων ερωτημάτων στη βάση δεδομένων και επεξεργασίας των αποτελεσμάτων (σε αυτή την περίπτωση δεν κάνουμε επεξεργασία φυσικής γλώσσας επομένως στοχεύουμε σε δομημένη γνώση)

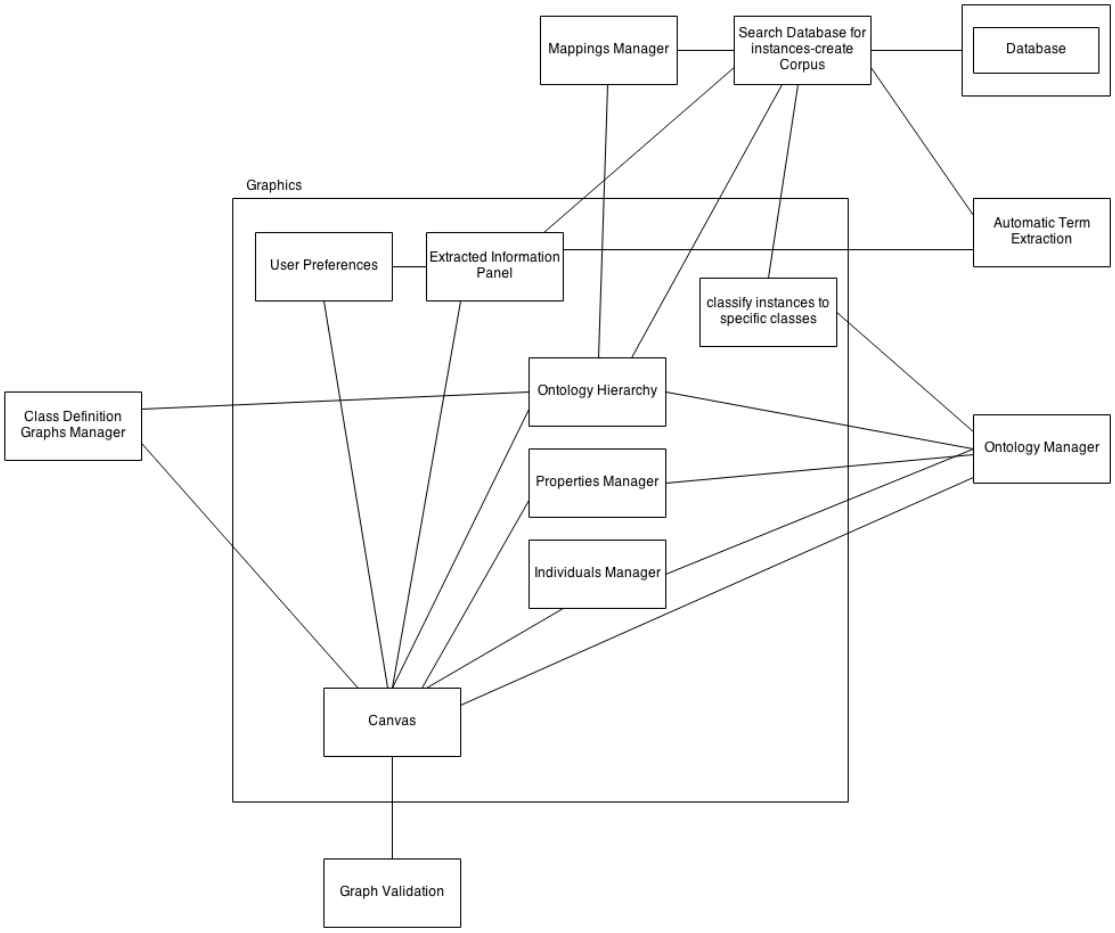
- Ημιαυτόματη ταξινόμηση των καταχωρημένων αντικειμένων σε νέες εξειδικευμένες κλάσεις της οντολογίας μας, σύμφωνα με τα χαρακτηριστικά τους που εντοπίζονται αυτόματα από τις γλωσσικές τους περιγραφές.
- Δυνατότητα πλήρους διαχείρισης και τροποποίησης της οντολογίας με γραφικό τρόπο.
    - Δυνατότητα ορισμού νέων κλάσεων δημιουργώντας σχήματα γράφων (χωρίς κώδικα και χωρίς εκφράσεις OWL)
    - Διαγραφή κλάσεων, τροποποίηση ορισμού κλάσεων, ανάκτηση του γράφου ορισμού μιας κλάσης
    - Δημιουργία νέων ιδιοτήτων (Object και Data Properties)
    - Δημιουργία νέων ατόμων, εύρεση στιγμιότυπων των κλάσεων και νέες εισαγωγές.
    - Έλεγχος συνέπειας βάσεις γνώσης, εκτέλεση SPARQL-DL ερωτημάτων.

Μη λειτουργικές απαιτήσεις:

- Όλες οι λειτουργίες να είναι προσβάσιμες μέσω του γραφικού περιβάλλοντος.
- Ευελιξία και εύκολη προσαρμογή σε όλες τις βάσεις δεδομένων.
- Εύκολα επεκτάσιμο με νέους αλγόριθμους επεξεργασίας και απεικόνισης των δεδομένων από τον χρήστη (υλοποιώντας μια διεπαφή και χωρίς να χρειάζεται πουθενά τροποποίηση ο κώδικα της εφαρμογής)
- Εξατομίκευση χρήστη. Η μορφή της εφαρμογής αλλά και ο τρόπος λειτουργίας της να μπορούν να παραμετροποιηθούν.

3.1 Domain Model Class Diagram

Εδώ παρουσιάζεται η αρχιτεκτονική του συστήματος, τα διάφορα μέρη που το αποτελούν και πώς αυτά αλληλεπιδρούν μεταξύ τους.



Εικόνα 3.1-1 Domain model class diagram

3.2 Περιγραφή Λειτουργιών

3.2.1 OntologyManager

Το δομικό αυτό στοιχείο αναλαμβάνει την διαχείριση της εξεταζόμενης οντολογίας. Επομένως, οποιοδήποτε αντικείμενο της εφαρμογής θέλει να μάθει κάτι για την δομή της οντολογίας, τις κλάσεις της, τις ιδιότητές της, τα άτομα ή να κάνει reasoning πάνω στην οντολογία, επικοινωνεί με τον OntologyManager. Για αυτό πρέπει να είναι εύκολα προσβάσιμος από οποιοδήποτε σημείο της εφαρμογής. Αυτό πετυχαίνεται εύκολα με το να υλοποιηθεί με το σχεδιαστικό μοτίβο Singleton.

3.2.2 MappingsManager

Αυτό το δομικό στοιχείο αναλαμβάνει να γνωρίζει για κάθε κλάση της οντολογίας τον τρόπο που θα αναζητήσουμε στιγμιότυπά της στη βάση δεδομένων (mapping).

Βρίσκοντας στιγμιότυπα των κλάσεων, βρίσκει ταυτόχρονα και τις γλωσσικές τους περιγραφές (κείμενα σε φυσική γλώσσα). Είναι σε θέση να απαντά στο ερώτημα αν μια συγκεκριμένη κλάση έχει ορισμένη δική της ξεχωριστή αντιστοίχιση. Παρέχει κατάλληλο γραφικό περιβάλλον για ορισμό και τροποποίηση των αντιστοιχίσεων με τη βάση. Επίσης, αναλαμβάνει να αποθηκεύει αυτή την πληροφορία σε μόνιμο μέσο, ώστε να είναι διαθέσιμη και στις επόμενες χρήσεις της εφαρμογής. Φυσικά, αναλαμβάνει και την αντίστροφη διαδικασία, δηλαδή την ανάκτηση της πληροφορίας από το μόνιμο μέσο αποθήκευσης. Υπάρχει μόνο ένα στιγμιότυπο αυτού του αντικειμένου για όλη την εφαρμογή και έχει σχεδιαστεί με το μόρφημα Singleton.

3.2.3 Class Definition Graphs Manager

Αυτό το δομικό στοιχείο αναλαμβάνει να κρατά αποθηκευμένους τους γράφους ορισμού κλάσεων. Δηλαδή για όλες τις κλάσεις που έχουν δημιουργηθεί με τη συγκεκριμένη εφαρμογή, αποθηκεύει τους γράφους ορισμού τους και έτσι μπορούμε να τους ανακτήσουμε οποιαδήποτε στιγμή και να τους απεικονίσουμε στον καμβά. Αναλαμβάνει, επίσης, την αποθήκευση αυτής της πληροφορίας και την ανάκτησή της από μόνιμο μέσο αποθήκευσης ώστε να διατηρείται και σε επόμενες χρήσεις της εφαρμογής.

3.2.4 Database

Η Βάση Δεδομένων απ’ όπου αντλούμε πληροφορίες. Δεν υπάρχει κάποιος περιορισμός ως προς το σχήμα βάσης που μπορούμε να χρησιμοποιήσουμε. Η εφαρμογή προς το παρόν τρέχει μόνο mysql.

3.2.5 Search Database for Instances-create Corpus

Αυτή η υπομονάδα αναλαμβάνει την αναζήτηση στη βάση δεδομένων για στιγμιότυπα κλάσεων. Όταν ο χρήστης επιλέξει αυστηρή αναζήτηση (στιγμιότυπα μόνο αυτής της κλάσης) ή εκτεταμένη αναζήτηση (στιγμιότυπα αυτής ή κάποιας υποκλάσης της), ρωτώντας τον mapping manager συλλέγει ένα σύνολο ερωτήσεων που πρέπει να εκτελέσει στη βάση δεδομένων. Στη συνέχεια, εκτελεί αυτά τα ερωτήματα και από τα αποτελέσματα που λαμβάνει δημιουργεί μια Συλλογή Κειμένων πάνω στην οποία θα γίνει αυτόματη εξαγωγή όρων.

3.2.6 Automatic term extraction

Αυτό το υποσύστημα παίρνει την Συλλογή Κειμένων που έχει δημιουργηθεί από το Search Database for Instances-create Corpus και εφαρμόζει διάφορους αλγόριθμους αυτόματης εξαγωγής όρων. Αυτό το υποσύστημα έχει δημιουργηθεί χρησιμοποιώντας το εργαλείο jate[26] με ορισμένες τροποποιήσεις. Υποστηρίζει πολλούς διαφορετικούς αλγόριθμους όπως ο simple term frequency, average term frequency in corpus, tf-idf, Weirdness, C-value, GloosEx, TermEx. Από προεπιλογή εκτελεί τον αλγόριθμο simple term frequency. Κάνει αυτόματη εξαγωγή ονοματικών φράσεων (noun phrases) και επιθέτων.

3.2.7 Extracted Information Panel

Αυτό το υποσύστημα αναλαμβάνει να απεικονίσει τα αποτελέσματα που επιστρέφονται από την διαδικασία αυτόματης εξαγωγής όρων, καθώς και από τα επιπρόσθετα ερωτήματα (additional queries) που εκτελούνται στη βάση δεδομένων. Παρουσιάζει τα δεδομένα σε μορφή πίνακα και υποστηρίζει την εύκολη μεταφορά τους σε άλλα υποσυστήματα της εφαρμογής με τη χειρονομία drag and drop. Για να είναι η εφαρμογή πλήρως προσαρμόσιμη στις ανάγκες του χρήστη και στα σχήματα βάσης δεδομένων που χρησιμοποιούνται, προσφέρει ένα επιπλέον επίπεδο απομόνωσης μεταξύ της εξαγωγής των δεδομένων (είτε από τη βάση είτε από την αυτόματη εξαγωγή όρων) και της παρουσίασής τους. Δηλαδή περνά τα δεδομένα από ένα φίλτρο μετασχηματισμού. Ο χρήστης μπορεί να αλλάξει το φίλτρο που χρησιμοποιείται κατά την λειτουργία της εφαρμογής καθώς και να δημιουργήσει νέα δικά του φίλτρα χωρίς να τροποποιήσει καθόλου τον κώδικα της εφαρμογής, ούτε να γνωρίζει τον τρόπο υλοποίησής της. Το υποσύστημα υλοποιεί το σχεδιαστικό μόρφημα Observer και κάνει subscribe στο υποσύστημα User Preferences, ώστε να αλλάζει μορφή ανάλογα με τις προτιμήσεις χρήστη.

3.2.8 Classify instances to specific classes

Το υποσύστημα αυτό βοηθά τον χρήστη να ταξινομήσει τα στιγμιότυπα που βρήκαμε στη βάση δεδομένων σε νέες εξειδικευμένες κλάσεις της οντολογίας μας. Για παράδειγμα, να ταξινομήσει ένα αντικείμενο που είναι καταχωρημένο απλά ως μέλος της κλάσης pendant (κρεμαστό κόσμημα) σε μια νέα κλάση που θα αναδεικνύει σε μεγαλύτερο βαθμό τα χαρακτηριστικά του (πχ στην κλάση pendant ⊓

∃hasMaterial.gold ⊓ ∃hasTechnique.enamelling). Η νέα αυτή κλάση, αν δεν υπάρχει ήδη, ορίζεται μέσω του γραφικού περιβάλλοντος Η διαδικασία ταξινόμησης γίνεται ημιαυτόνομα. Δηλαδή ο χρήστης ορίζει ποια είναι τα νέα αυστηρά κριτήρια αναζήτησης στιγμιότυπων και επιβεβαιώνει τα αποτελέσματα που του επιστρέφονται ως προτεινόμενα από το σύστημα.

3.2.9 User Preferences

Το υποσύστημα αυτό διαχειρίζεται τις προτιμήσεις χρήστη. Συγκεκριμένα προσφέρει κατάλληλο γραφικό περιβάλλον ώστε να τροποποιεί τις προτιμήσεις του ο χρήστης και υποδομή για να γίνονται αντιληπτές οι αλλαγές αυτές σε όλους τους ενδιαφερόμενους άμεσα (observable object). Επίσης αναλαμβάνει την αποθήκευση και ανάκτηση των προτιμήσεων του χρήστη σε μόνιμο μέσο αποθήκευσης. Οι προτιμήσεις χρήστη αφορούν τον τρόπο λειτουργίας της εφαρμογής, αλλά και τον τρόπο εμφάνισης.

3.2.10 OntologyHierarchy

Το υποσύστημα αυτό αναλαμβάνει να απεικονίζει γραφικά την εξεταζόμενη οντολογία. Η απεικόνιση γίνεται με δενδρική δομή. Επικοινωνεί με τον OntologyManager για να αντλεί τις απαραίτητες πληροφορίες και να ενημερώνεται για τις αλλαγές που πραγματοποιούνται (δημιουργία καινούργιων κλάσεων, διαγραφή κλάσεων). Επικοινωνεί με τον Mappings Manager και τον Class Definition Graphs Manager ώστε να γνωρίζει ποιες κλάσεις έχουν ορισμένη δική τους αντιστοίχιση με τη βάση (mapping), και για ποιες κλάσεις έχουμε διαθέσιμο τον γράφο ορισμού τους· πληροφορία που γίνεται ορατή στο χρήστη. Δίνει την δυνατότητα μεταφοράς πληροφορίας (δηλαδή των κλάσεων που απεικονίζει) σε άλλα υποσυστήματα με τη χειρονομία drag and drop.

3.2.11 Properties Manager

Επικοινωνεί με τον Ontology Manager για να βρει όλες τις ιδιότητες (Object και Data Properties) που χρησιμοποιούνται στην οντολογία και τις απεικονίζει σε γραφικό περιβάλλον. Μέσω χειρονομίας drag and drop δίνει την δυνατότητα στο χρήστη να μεταφέρει την απεικονιζόμενη πληροφορία σε άλλα υποσυστήματα. Επίσης, δίνει την δυνατότητα ορισμού νέων ιδιοτήτων.

3.2.12 Individuals Manager

Αυτό το υποσύστημα επικοινωνεί με τον Ontology Manager και είναι σε θέση να δείχνει ποια άτομα είναι μέλη της κάθε κλάσης, καθώς και να εισάγει καινούργια άτομα ως μέλη, όλα με κατάλληλο γραφικό περιβάλλον.

3.2.13 Canvas

Αυτό το υποσύστημα αναλαμβάνει να προσφέρει στον χρήστη έναν καμβά, πάνω στον οποίο θα μπορεί να σχεδιάζει γράφους που θα αντιστοιχούν σε ορισμούς νέων κλάσεων. Είναι σε θέση να δέχεται δεδομένα (κλάσεις, ιδιότητες, άτομα, λεκτικά) μέσω λειτουργίας drag and drop. Προσφέρει διάφορες δυνατότητες απεικόνισης όπως εστίαση εικόνας, περιστροφή συστήματος συντεταγμένων, στρέβλωση, μεταφορά κόμβων ανεξάρτητα ή και σε ομάδες, δυνατότητα δημιουργίας σχολίων σε οποιοδήποτε σημείο, εξαγωγή του γράφου σε μορφή εικόνας jpeg. Μπορεί να δημιουργήσει νέες κλάσεις, Object και Data properties αλλά και individuals στην οντολογία με γραφικό τρόπο. Πριν αποθηκεύσει τη νέα κλάση επαληθεύει την ορθότητα του γράφου επικοινωνώντας με το υποσύστημα Graph Validation. Υλοποιεί το σχεδιαστικό μόρφημα Observer και κάνει subscribe στο υποσύστημα User Preferences. Έτσι ο χρήστης μπορεί να αλλάξει άμεσα διάφορους παραμέτρους που αφορούν την εμφάνιση του γράφου (πχ θέση των ετικετών των κόμβων και των ακμών) καθώς και τον τρόπο σχεδίασης στο γράφο (πλήκτρα για την δημιουργία νέων κόμβων κλπ).

3.2.14 Graph Validation

Αυτή η δομική μονάδα εξετάζει τον γράφο ορισμού για ορθότητα. Αν είναι σωστός, τότε ολοκληρώνεται κανονικά η διαδικασία ορισμού της νέας κλάσης. Διαφορετικά εμφανίζεται στον χρήστη κατάλληλο διαγνωστικό μήνυμα σφάλματος.

3.3 Περιπτώσεις χρήσης

Για να φανεί καλύτερα ο τρόπος αλληλεπίδρασης των διαφορετικών υποσυστημάτων και οι λειτουργίες τους, παρουσιάζονται αναλυτικά μερικές σημαντικές λειτουργίες της εφαρμογής.

3.3.1 Αναζήτηση στιγμιότυπων και αυτόματη εξαγωγή όρων

Περιγραφή:

Το σενάριο αυτό περιγράφει τις ενέργειες που εκτελούνται από τα διάφορα υποσυστήματα, όταν ο χρήστης επιθυμεί να εξειδικεύσει τον ορισμό μιας κλάσης, αξιοποιώντας τις εξαγόμενες πληροφορίες από την αυτόματη εξαγωγή όρων και από τα επιπρόσθετα ερωτήματα στη βάση δεδομένων.

Δράστες:

Χρήστης, Ontology Manager, Search Database for Instances-create Corpus, Mappings Manager, Database, Automatic Term Extraction, Extracted Information Panel

Προϋποθέσεις:

1. Ο mysql server να είναι ενεργοποιημένος.
2. Έχουμε δώσει σωστά στοιχεία σύνδεσης με τη βάση δεδομένων.
3. Έχουμε ορίσει αντιστοιχίσεις (mappings) με τη βάση για την κλάση που θέλουμε να επεκτείνουμε ή για κάποια υπερκλάση της.
4. Ο χρήστης έχει επιλέξει την κλάση που τον ενδιαφέρει από τη δενδρική απεικόνιση των κλάσεων της οντολογίας.

Σενάριο:

1. Ο χρήστης ξεκινά την αναζήτηση, επιλέγοντας αυστηρή ή εκτεταμένη αναζήτηση.
2. Το υποσύστημα Search Database for Instances-create Corpus επικοινωνεί με τον Ontology Manager και φτιάχνει ένα σύνολο με τις εξεταζόμενες κλάσεις. Αν πρόκειται για αυστηρή αναζήτηση το σύνολο αυτό περιέχει μόνο την επιλεγμένη κλάση, ενώ αν πρόκειται για εκτεταμένη αναζήτηση, τότε περιέχει αυτήν και όλες τις υποκλάσεις της.
3. Search Database for Instances-create Corpus: δημιουργεί ένα σύνολο με όλα τα ερωτήματα που θα εκτελεστούν στη βάση δεδομένων.
   3.1. Για κάθε κλάση του εξεταζόμενου συνόλου ρωτά τον Mappings Manager ποιο mapping της αντιστοιχεί και επομένως ποια ερωτήματα στη βάση.
   3.2. Ομαδοποιεί τα ερωτήματα σύμφωνα με τα mappings από τα οποία προέρχονται.
4. Search Database for Instances-create Corpus: Εκτέλεση των ερωτημάτων και δημιουργία της Συλλογής Κειμένων.

4.1. Εκτέλεση των ερωτημάτων στη βάση δεδομένων.

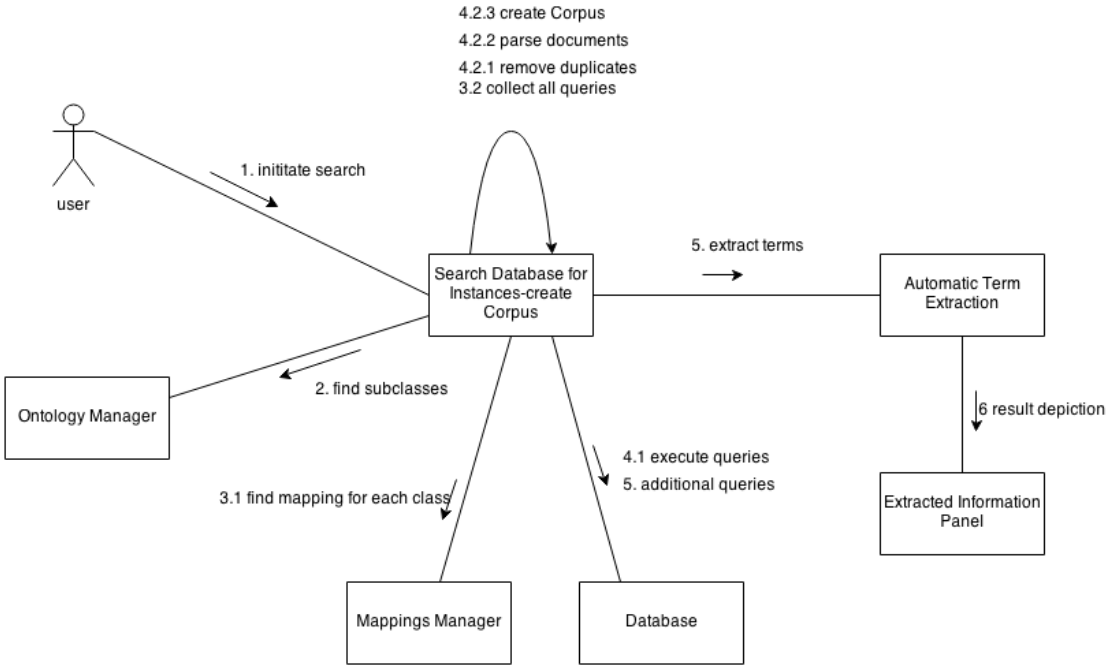
4.2. Συλλογή και επεξεργασία των αποτελεσμάτων.

4.2.1. Καταχωρήσεις που επιστρέφονται πολλαπλές φορές για το ίδιο mapping αφαιρούνται, διότι θα αντιστοιχούν σε ακριβώς ίδια γλωσσική περιγραφή.

4.2.2. Για κάθε καταχώρηση ενώνονται όλες οι στήλες που περιέχουν γλωσσικό περιεχόμενο και σχηματίζουν ένα έγγραφο.

4.2.3. Συλλογή όλων των εγγράφων για την δημιουργία της Συλλογής Κειμένων.

5. Αυτόματη εξαγωγή όρων από την Συλλογή Κειμένων και εκτέλεση των additional queries (γίνονται παράλληλα).
6. Απεικόνιση της εξαγόμενων όρων και των αποτελεσμάτων από τα additional queries σε γραφικό περιβάλλον.



Εικόνα 3.3-1 Συνεργατικό διάγραμμα για την αυτόματη εξαγωγή όρων

3.3.2 Ορισμός καινούργιας κλάσης

Περιγραφή:

Το σενάριο αυτό περιγράφει τις ενέργειες που εκτελούνται από τα διάφορα υποσυστήματα, όταν ο χρήστης έχει πλέον σχηματίσει τον γράφο ορισμού και επιλέξει την αποθήκευση της νέας κλάσης. Το στάδιο σχηματισμού του γράφου περιλαμβάνει επικοινωνία και μεταφορά πληροφορίας μεταξύ των διαφόρων υποσυστημάτων που όμως ενεργοποιείται και καθοδηγείται από τις ενέργειες του χρήστη, για αυτό δεν περιλαμβάνεται σε αυτό το σενάριο χρήσης.

Δράστες:

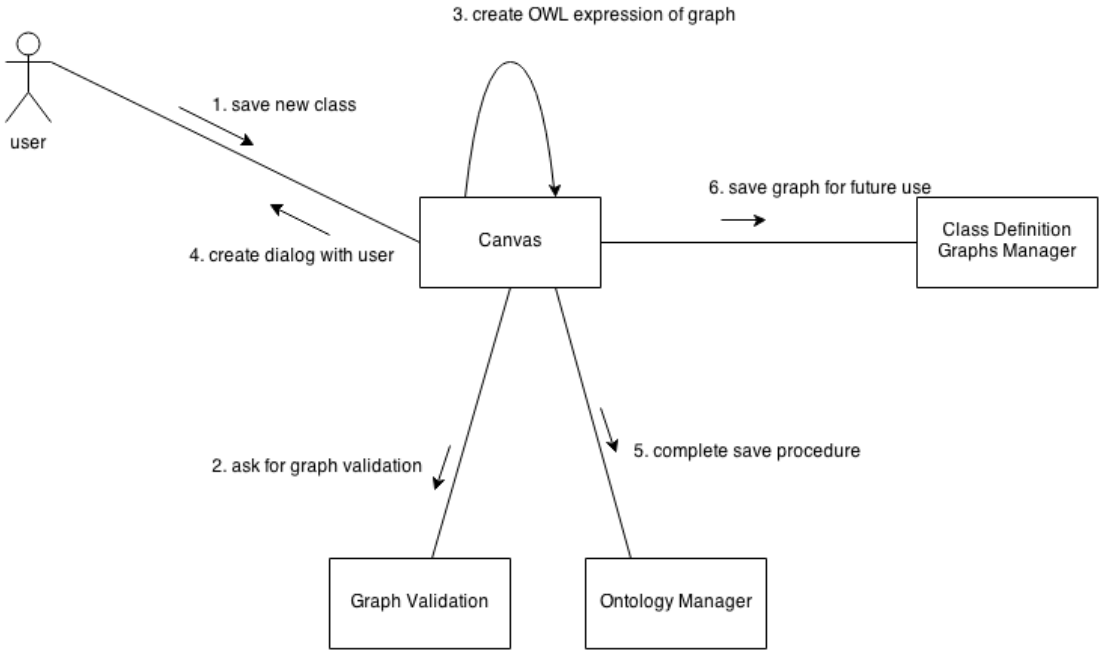
Χρήστης, Canvas, Graph Validation, Class Definition Graphs Manager, Ontology Manager,

Προϋποθέσεις:

1. Ο χρήστης έχει σχηματίσει τον γράφο ορισμού της νέας κλάσης.

Σενάριο:

1. Ο χρήστης επιλέγει την αποθήκευση της κλάσης.
2. Το υποσύστημα Canvas στέλνει τον γράφο στο υποσύστημα Graph Validation και ρωτά για την ορθότητά του.
   - Σε περίπτωση που υπάρχει κάποιο σφάλμα· κατάλληλο διαγνωστικό μήνυμα εμφανίζεται στον χρήστη
3. Canvas: διατρέχει τον γράφο bottom up (από τα φύλλα προς τη ρίζα) και σχηματίζει την owl έκφραση στην οποία αντιστοιχεί ο γράφος.
4. Αλληλεπίδραση με τον χρήστη. Δημιουργία διαλόγου για να προσδιορίσει το IRI της νέας κλάσης και αν θα είναι ισοδύναμη ή υποκλάση του γράφου ορισμού.
5. Επικοινωνία με τον Ontology Manager και αποθήκευση της νέας κλάσης.
6. Αποθήκευση του γράφου ορισμού στον Class Definition Graphs Manager.



Εικόνα 3.3-2 Συνεργατικό διάγραμμα για την αποθήκευση μιας νέας κλάσης

4

Έλεγχος εφαρμογής με πραγματικά δεδομένα

Προκειμένου να εξετάσουμε την λειτουργικότητα της εφαρμογής, την χρηστικότητά της για τη διαχείριση και την επέκταση οντολογιών, αλλά και την ικανότητά της να εξάγει αυτόματα χρήσιμους όρους από Συλλογές Κειμένων, χρησιμοποιήσαμε πραγματικά δεδομένα από μουσειακά εκθέματα στο χώρο της ενδυμασίας.

4.1 Δημιουργία οντολογίας

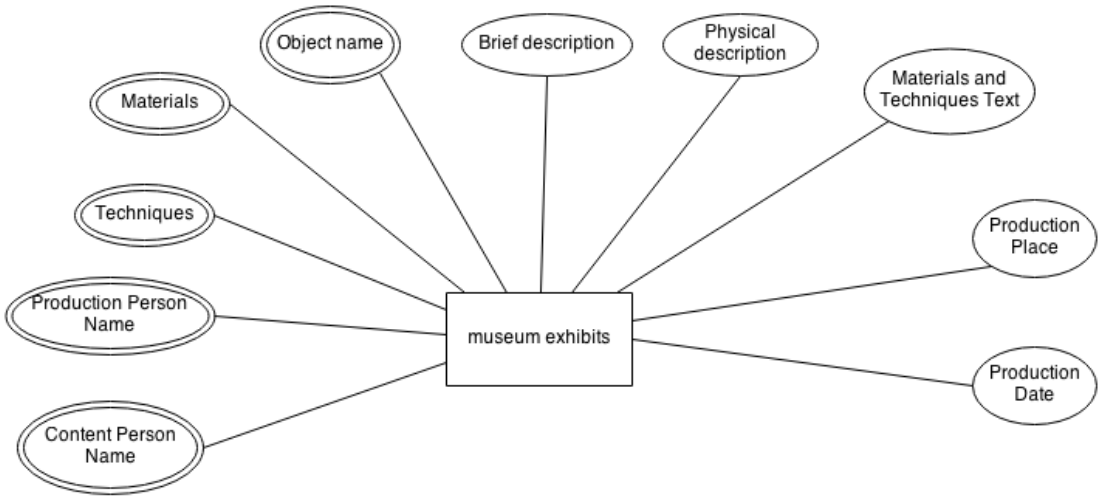
Η οντολογία μας δημιουργήθηκε με βάση το αρχείο testData.concepts_AAT_original.xls. Είναι ένα αρχείο excel που περιέχει σε δενδρική μορφή όλες τις έννοιες του τομέα που εξετάζουμε. Δηλαδή πρόκειται για μια ιεραρχική αναπαράσταση όλων των εννοιών που περιγράφουν το χώρο της ενδυμασίας. Η δομή αυτή είναι κατανοητή και χρήσιμη για τον άνθρωπο, όχι όμως και για τον υπολογιστή. Για αυτό έγινε μετατροπή αυτής της πληροφορίας σε owl οντολογία (ontologies.concepts_AAT_original.owl). Αυτή την οντολογία θα χρησιμοποιήσουμε στα επόμενα παραδείγματά μας και θα προσπαθήσουμε να την επεκτείνουμε με νέες εξειδικευμένες κλάσεις. Τα βήματα μετατροπής σε owl οντολογία ήταν τα εξής:

- Αφαίρεση της περιττής πληροφορίας που δυσχεραίνει την αυτόματη αναζήτηση (δηλαδή τις παρενθέσεις που υπάρχουν και τους μη αλφαριθμητικούς χαρακτήρες).
- Λημματοποίηση των όρων ώστε να είναι σε κανονική μορφή (βοηθά στην αυτόματη αναζήτηση όρων).
- Δημιουργία οντολογίας. Για κάθε κελί δημιουργείται μια κλάση με URI το URI της οντολογίας και προσαρτώντας σε αυτό ένα τμηματικό αναγνωριστικό (fragment identifier). Η ιεραρχική δομή διατηρείται, δημιουργώντας σχέσεις μεταξύ των κλάσεων με την ιδιότητα rdfs:subClassOf. Επίσης, για κάθε κλάση δημιουργείται ένα σχόλιο rdfs:label. Στο σχόλιο αυτό αποθηκεύεται η έννοια έτσι όπως είναι στο κελί, δηλαδή σε μορφή αναγνώσιμη από τον άνθρωπο. Αυτό το σχόλιο χρησιμοποιείται για την γραφική αναπαράσταση της έννοιας στην εφαρμογή, αλλά και την αναζήτησή στιγμιότυπών της στη βάση δεδομένων.

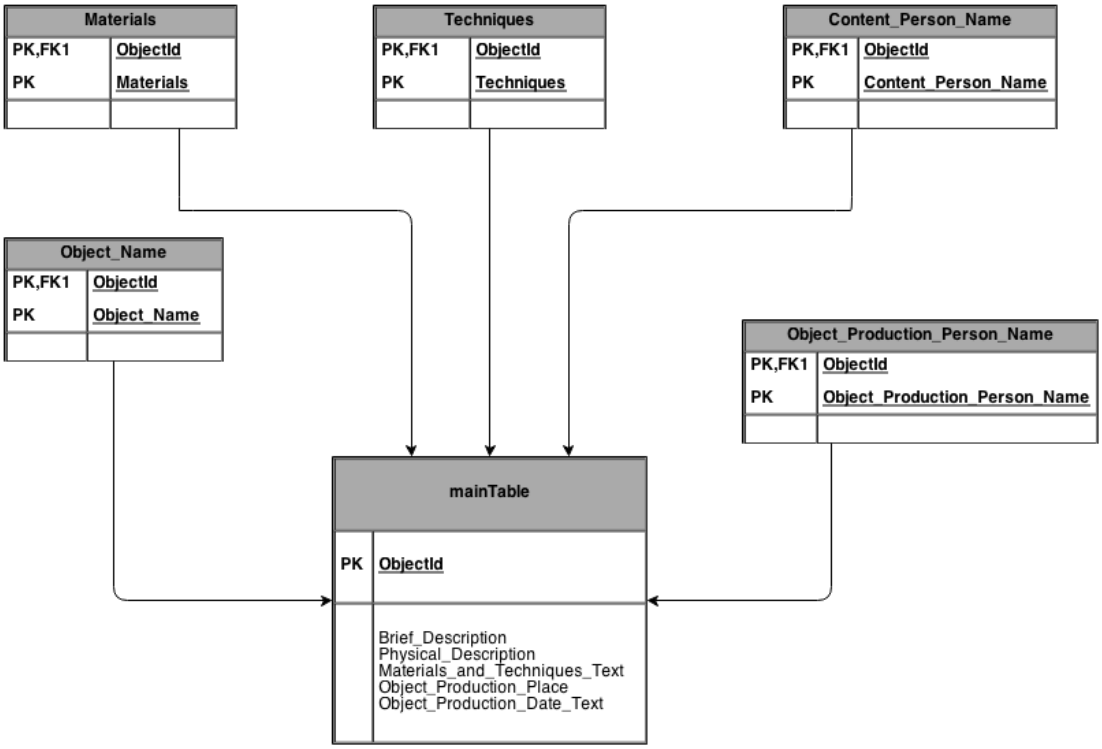
4.2 Δημιουργία Βάσης Δεδομένων

Η βάση δεδομένων με τα μουσειακά εκθέματα που δημιουργήθηκε, στηρίζεται στο αρχείο testData.Data_20_August_2013.xls. Το αρχείο αυτό περιλαμβάνει 3740 καταχωρίσεις εκθεμάτων. Για κάθε έκθεμα έχει ένα όνομα που δείχνει την κατηγορία στην οποία ανήκει (πχ dress, ring, watch), μια σύντομη περιγραφή του σε φυσική γλώσσα και μια αναλυτική περιγραφή του επίσης σε φυσική γλώσσα. Για ορισμένα εκθέματα έχουμε επιπλέον πληροφορία σε δομημένη μορφή, δηλαδή σε πίνακες και όχι σε κείμενα. Αυτή η πληροφορία αφορά τα υλικά από τα οποία είναι κατασκευασμένα, την τεχνική κατασκευής τους, τους δημιουργούς τους κ.α.. Η δημιουργία της βάσης δεδομένων αλλά και η καταχώριση των εκθεμάτων γίνεται αυτόματα από την εφαρμογή. Έτσι δημιουργούνται τα δύο αρχεία (create_mydb.txt και populate_mydb.txt) του φακέλου database. Η υπάρχουσα κατηγοριοποίηση των εκθεμάτων δεν είναι πλήρως συμβατή με την οντολογία. Δηλαδή ένα έκθεμα μπορεί να έχει ετικέτα «pair of shoes» ενώ στην οντολογία να έχουμε της κλάση «shoe». Χρησιμοποιώντας όμως τις κατάλληλες δομές και ερωτήματα στη βάση δεδομένων αυτό το πρόβλημα επιλύεται εύκολα.

Το σχήμα της βάσης είναι το εξής:



Εικόνα 4.2-1 Διάγραμμα οντοτήτων συσχετίσεων



Εικόνα 4.2-2 Σχεσιακό σχήμα

4.3 Παραδείγματα ορισμού εξειδικευμένων κλάσεων και ταξινόμησης στιγμιότυπων

Παρατηρώντας την οντολογία μας και τις καταχωρήσεις των αντικειμένων στη βάση δεδομένων, γρήγορα αναγνωρίζουμε δύο σημαντικά προβλήματα. Πρώτον, η

οντολογία μας έχει ορισμένες μόνο γενικές κλάσεις (πχ jewelry, ring, pendant, dress) και όχι ειδικές (πχ pendant ⊓ ∃hasMaterial.gold ⊓ ∃hasTechnique.enamelling). Δεύτερον, τα αντικείμενα (εκθέματα) που έχουμε καταχωρημένα στη βάση δεδομένων έχουν αποθηκευμένη την περιγραφή τους και τα χαρακτηριστικά σε φυσική γλώσσα και όχι με δομημένο τρόπο. Έτσι, αυτή η πληροφορία μένει αναξιοποίητη και δεν μπορεί να χρησιμοποιηθεί για την ταξινόμησή τους σε πιο ειδικές κλάσεις που αναδεικνύουν τα ιδιαίτερα χαρακτηριστικά τους.

Για παράδειγμα, έχουμε στη βάση καταχωρημένο το αντικείμενο με objectId=523.

Id=523

Object name= pendant

Brief Description: Enamelled pendant in gold and silver-gilt surround with pendant amethysts and an opal, designed and made by Nelson and Edith Dawson, London, 1900.

Physical Description: Enamel plaque with a floral motif in a surround of gold with three pendants. Gold and silver-gilt, set with an enamel plaque of a cluster of tradescantia and drops of opal and amethyst. The central pendant is an irregularly shaped opal, on each side are two faceted amethysts. The chain is of gold. Dated 1900. The enamel, probably by Edith Dawson, signed with the initial D enclosed by a leaf.

Εικόνα 4.3-1 Παράδειγμα καταχώρησης στη βάση δεδομένων

Αυτό το αντικείμενο έχει ετικέτα pendant. Όμως ακόμα δεν έχει εισαχθεί στην οντολογία, δηλαδή δεν υπάρχει ως άτομο, και επομένως ούτε σαν στιγμιότυπο της κλάσης pendant. Κοιτώντας όμως την περιγραφή του σε φυσική γλώσσα βρίσκουμε πολύ πληροφορία και κρίνουμε σκόπιμο να το εντάξουμε ως μέλος κάποιας ειδικής κλάσης και όχι της γενικής (pendant). Ιδανικά θα θέλαμε να το εισάγουμε σαν μέλος της κλάσης pendant ⊓ ∃hasMaterial.gold ⊓ ∃hasMaterial.opal ⊓ ∃hasMaterial.amethyst ⊓ ∃hasMaterial.silver-gilt ⊓ ∃hasTechnique.enamelling

Όμως αυτή η ειδική κλάση δεν υπάρχει στην οντολογία. Επίσης, η ταξινόμηση των αντικειμένων στις ειδικές αυτές κλάσεις πρέπει να μπορεί να γίνεται αυτόματα, ώστε να αποφεύγεται η χρονοβόρα διαδικασία ανάγνωσης κειμένων. Αυτά τα προβλήματα επιλύει το σύστημά μας.

Αρχικά, προτείνει περιεχόμενο στο χρήστη για να τον βοηθήσει να αποφασίσει ποιες είναι οι νέες εξειδικευμένες κλάσεις που πρέπει ορίσει. Αυτό γίνεται ως εξής: ψάχνει στη βάση δεδομένων και βρίσκει όλα τα αντικείμενα που φέρονται να είναι στιγμιότυπα της κλάσης που θέλουμε να επεκτείνουμε. Χρησιμοποιείται η λέξη φέρονται και όχι είναι, διότι αυτή η αναζήτηση μπορεί να γίνεται με ταίριασμα συμβολοσειρών (string matching) ή τιμών σε συγκεκριμένα πεδία πινάκων και όχι με βάση reasoning πάνω στην οντολογία (οπότε μπορεί να μην είμαστε σίγουροι αν όντως είναι μέλη της κλάσης). Στην συνέχεια, παίρνουμε τις γλωσσικές τους περιγραφές και δημιουργούμε μια Συλλογή Κειμένων. Πάνω σε αυτή τη Συλλογή Κειμένων κάνουμε αυτόματη εξαγωγή ονοματικών φράσεων και επιθέτων. Τους εξαγμένους όρους τους προτείνουμε στον χρήστη, ταξινομημένους με βάση διάφορα στατιστικά στοιχεία (πχ συχνότητα εμφάνισης του όρου στη Συλλογή Κειμένων κ.α.). Αυτή είναι η αυτόματη άντληση γνώσης από αδόμητες γλωσσικές περιγραφές.

Συγκεκριμένα, για να επεκτείνουμε την κλάση pendant βρίσκουμε στη βάση 260 καταχωρήσεις. Δηλαδή 260 περιγραφές σαν αυτή της προηγούμενης εικόνας (Εικόνα 4.3-1). Επομένως, η Συλλογή Κειμένων που δημιουργείται αποτελείται από 260 έγγραφα.

Όπως έχει αναφερθεί, ορισμένα αντικείμενα εκτός από τις γλωσσικές τους περιγραφές έχουν και δομημένη πληροφορία καταχωρημένη σε πίνακες της βάσης. Με την υποδομή που παρέχουμε αντλείται και αυτή μέσω των κατάλληλων ερωτημάτων και επεξεργασίας των αποτελεσμάτων.

Ενδεικτικά, παρουσιάζουμε ορισμένα αποτελέσματα που προτείνονται στον χρήστη για την κλάση pendant. Οι δύο πρώτες στήλες προέρχονται από την αυτόματη εξαγωγή όρων, ενώ οι υπόλοιπες από τη δομημένη πληροφορία. Σε κάθε κελί φαίνεται ο προτεινόμενος όρος και η συχνότητα εμφάνισής του στις περιγραφές.

noun phrases	**adjectives**	**materials**	**techniques**	**production person name**	**content person name**
pendant 184	enameled 139	gold 167	gilding 20	Giuliano Carlo 5	Jesus Christ 36
pearl 128	silver-gilt 41	enamel 92	casting 12	Vasters Reinhold 5	Mary (Virgin Mary) 32

gold 126	gold 37	pearl 52	engraving 12	Traquair Phoebe Anna 4	Cupid 6
silver 65	circular 19	diamond 35	filigree 8	Morris Mary 1	John (Saint John the Baptist) 5
enameled gold pendant 61	inscribed 11	ruby 26	lapidary 8	Novissimo Pasquale 1	Venus 2
glass 46	foiled 10	emerald 21	carving 6	Wyon Thomas 1	St. Catherine of Alexandria 2
rock crystal 39	opal 8	garnet 12	embossing 2	Bissinger Georges 1	Hercules 2
rose-cut diamond 28	engraved 7	ivory 4	painting 1	Rouw Peter 1	Charles I (King of England) 2

Πίνακας 3.3.2-1 Προτεινόμενο περιεχόμενο για την κλάση pendant

Με βάση αυτή την πληροφορία μπορούμε εύκολα να αποφασίσουμε ποιες νέες κλάσεις θα ορίσουμε. Για παράδειγμα, είναι χρήσιμο να ορίσουμε την κλάση pendant ⊓ ∃hasMaterial.gold καθώς πολλά εκθέματα χρησιμοποιούν αυτό το υλικό ή την κλάση pendant ⊓ ∃hasTechnique.casting διότι παρατηρούμε ότι είναι δημοφιλής τεχνική κατασκευής. Για λόγους παρουσίασης της λειτουργικότητας της εφαρμογής, ορίσαμε τις κλάσεις της εικόνας (Εικόνα 4.3-2). Οι νέες αυτές κλάσεις δημιουργήθηκαν μέσω του γραφικού περιβάλλοντος, κατασκευάζοντας σχήματα γράφων (Κεφάλαιο 5.9).

Εικόνα 4.3-2 Ορισμός νέων εξειδικευμένων κλάσεων και ταξινόμηση των εκθεμάτων

Στη συνέχεια ταξινομήσαμε αντικείμενα στις διάφορες κλάσεις με τον αυτόματο μηχανισμό που παρέχει η εφαρμογή. Αυτό το κάναμε ενδεικτικά μόνο για ορισμένες κλάσεις. Το νούμερο που αναγράφεται κάτω από τις κλάσεις είναι το πλήθος των

στιγμιότυπων που βρήκαμε. Η ταξινόμηση αυτή χρειάζεται να γίνει μόνο για τις κλάσεις του πρώτου επιπέδου (πχ ∃hasMaterial.gold ή ∃hasMaterial.pearl). Τα επόμενα επίπεδα αποκτούν στιγμιότυπα μέσα από τις υπηρεσίες εξαγωγής συμπερασμάτων του reasoner.

Στο παράδειγμά μας βλέπουμε ότι η κλάση pendant ⊓ ∃hasMaterial.gold ⊓ ∃hasMaterial.pearl ⊓ ∃hasContent.Jesus_Christ έχει αποκτήσει δύο στιγμιότυπα (Εικόνα 4.3-2) . Οι γλωσσικές περιγραφές αυτών των αντικειμένων είναι οι ακόλουθες

Id=1244

Object name= pendant

Physical Description: Enamelled gold pendant set with pearls and garnets, with a cast figure of the Infant Christ standing in a niche, the pendant made in Spain, the figure made in India, Goa 1680-1700.

Id=1034

Object name= pendant

Brief Description: Pendant, enamelled gold, decorated with Behold the Man! (*Ecce Homo*) and scenes from Christ's Passion, made in France about 1640-1650

Physical Description: Pendant, gold decorated with grisaille enamel, in the centre, Behold the Man! (*Ecce Homo*), surrounded by scenes from the Passion of Christ; the reverse with painted enamel flowers. The pendant is hung with a pearl.

Με αυτή την ημιαυτόματη διαδικασία έχουμε καταφέρει να επεκτείνουμε την οντολογία μας με πολλές νέες εξειδικευμένες κλάσεις και να ταξινομήσουμε τα αντικείμενά μας σε αυτές, χωρίς να χρειαστεί να διαβάσουμε τις περιγραφές τους (το αναλαμβάνει ο υπολογιστής).

5

Περιγραφή Λειτουργικότητας Εφαρμογής

Σε αυτό το κεφάλαιο περιγράφουμε αναλυτικά τον τρόπο λειτουργίας της εφαρμογής, τα διάφορα χαρακτηριστικά και δυνατότητες που έχει. Επομένως, μπορεί να χρησιμοποιηθεί σαν σύντομο εγχειρίδιο χρήσης. Προκειμένου να μπορεί να διαβαστεί ανεξάρτητα από το υπόλοιπο κείμενο, ορισμένες σημαντικές πληροφορίες επαναλαμβάνονται.

5.1 Εκκίνηση της εφαρμογής για πρώτη φορά

Έχει δοθεί ιδιαίτερη μέριμνα, ώστε να είναι πολύ εύκολη και γρήγορη η χρήση της εφαρμογής. Συγκεκριμένα, στο αρχείο υλοποίησης υπάρχει ένας φάκελος executable. Μέσα σε αυτόν βρίσκεται ένα αρχείο jar (diplwmatikh.jar) που είναι και η εφαρμογή μας. Για να ξεκινήσει η εφαρμογή, πρέπει να πατήσουμε διπλό κλικ στο jar αρχείο ή από ένα terminal να δώσουμε την παρακάτω εντολή:

java –jar diplwmatikh.jar

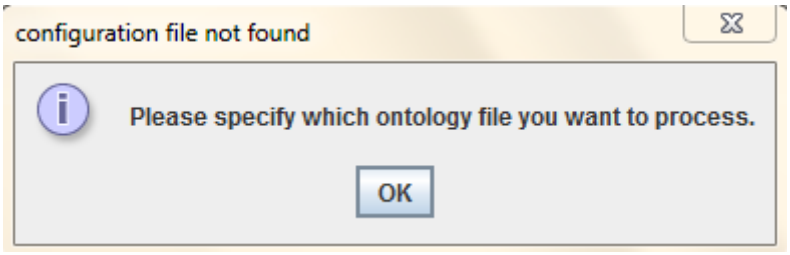
Προτείνεται ο δεύτερος τρόπος, καθώς η εφαρμογή τυπώνει κατά διαστήματα χρήσιμες πληροφορίες στο system out που αλλιώς δεν θα μπορούμε να τις δούμε.

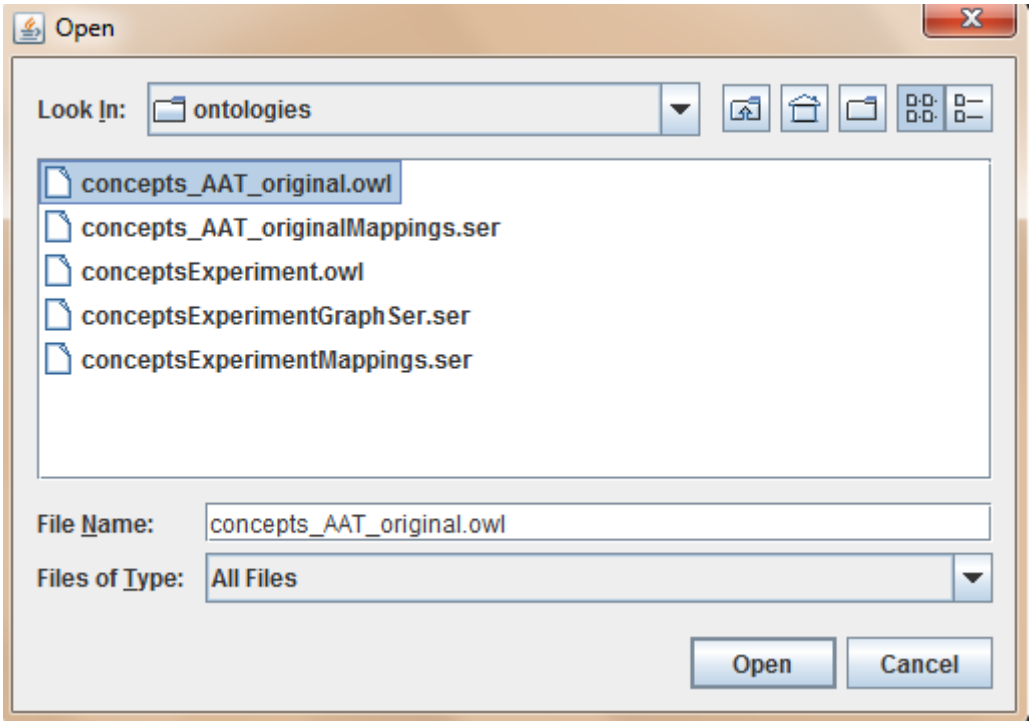
Κατά την πρώτη εκκίνηση θα χρειαστεί να δώσουμε ορισμένες πληροφορίες για να λειτουργήσει σωστά η εφαρμογή. Για παράδειγμα, σε ποια βάση δεδομένων να

συνδεθεί (όνομα χρήστη, κωδικό πρόσβασης, όνομα βάσης) και ποιο αρχείο οντολογίας να ανοίξει για επεξεργασία από το σύστημα αρχείων του υπολογιστή. Αυτές οι πληροφορίες όπως και πολλές άλλες που αφορούν τις προτιμήσεις χρήστη (εμφάνιση ενδιάμεσων αποτελεσμάτων αναζήτησης, τρόπος απεικόνισης των ακμών και των κόμβων του γράφου, τρόπος απεικόνισης των αποτελεσμάτων της αυτόματης διαδικασίας εξαγωγής όρων κ.α.) αποθηκεύονται σε ένα configuration file στον ίδιο φάκελο με το εκτελέσιμο αρχείο (config.properties). Την πρώτη φορά που ο χρήστης θα τρέξει την εφαρμογή, αυτό το αρχείο δεν υπάρχει. Οπότε δημιουργούνται κατάλληλα πάνελ, για να εισάγει ο χρήστης τις απαραίτητες πληροφορίες. Για τις υπόλοιπες παραμέτρους χρησιμοποιούνται κάποιες default τιμές.

5.1.1 Επιλογή αρχείου για επεξεργασία

Την πρώτη φορά που χρησιμοποιούμε την εφαρμογή πρέπει να επιλέξουμε το αρχείο που θέλουμε να επεξεργαστούμε. Τις επόμενες φορές θα φορτώνεται αυτόματα το τελευταίο αρχείο που είχαμε χρησιμοποιήσει. Το αρχείο πρέπει να είναι οντολογία. Χρησιμοποιείται η διεπαφή προγραμματισμού εφαρμογών OWL API[27] για τον χειρισμό και την επεξεργασία των οντολογιών, επομένως το αρχείο μπορεί να είναι σε κάποια από τις ακόλουθες μορφές Manchester syntax, Turtle, OWL/XML, RDF/XML, OWL functional syntax.

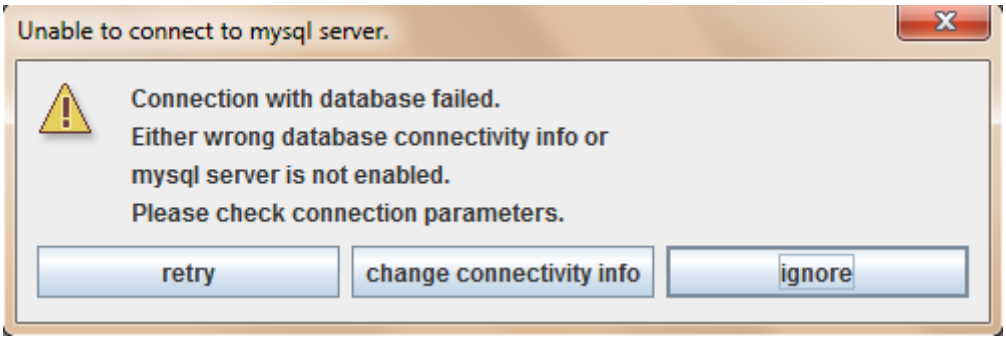




5.2 *Σύνδεση με τη Βάση Δεδομένων*

Την αδόμητη και ημιδομημένη πληροφορία, που αργότερα θα χρησιμοποιήσουμε για την αυτόματη εξαγωγή γνώσης, την βρίσκουμε από κάποια βάση δεδομένων. Η εφαρμογή έχει σχεδιαστεί με πολύ γενικό τρόπο, ώστε να μπορεί να προσαρμόζεται εύκολα σε όλα τα σχήματα βάσεων δεδομένων. Αυτή τη στιγμή υπάρχει μόνο ένας περιορισμός: η βάση δεδομένων πρέπει να είναι mysql.

Έχει ληφθεί μέριμνα, ώστε να επιλύονται διάφορα προβλήματα επικοινωνίας με τον server της βάσης γραφικά και με κατάλληλα μηνύματα προς τον χρήστη, χωρίς να τερματίζεται η λειτουργία της εφαρμογής.



Αυτό το μήνυμα μπορεί να εμφανιστεί, όταν ξεκινά η εφαρμογή ή εκτελείται κάποιο ερώτημα στη βάση (strict ή extended για την εξαγωγή όρων) και συντρέχει ένας από τους παρακάτω λόγους:

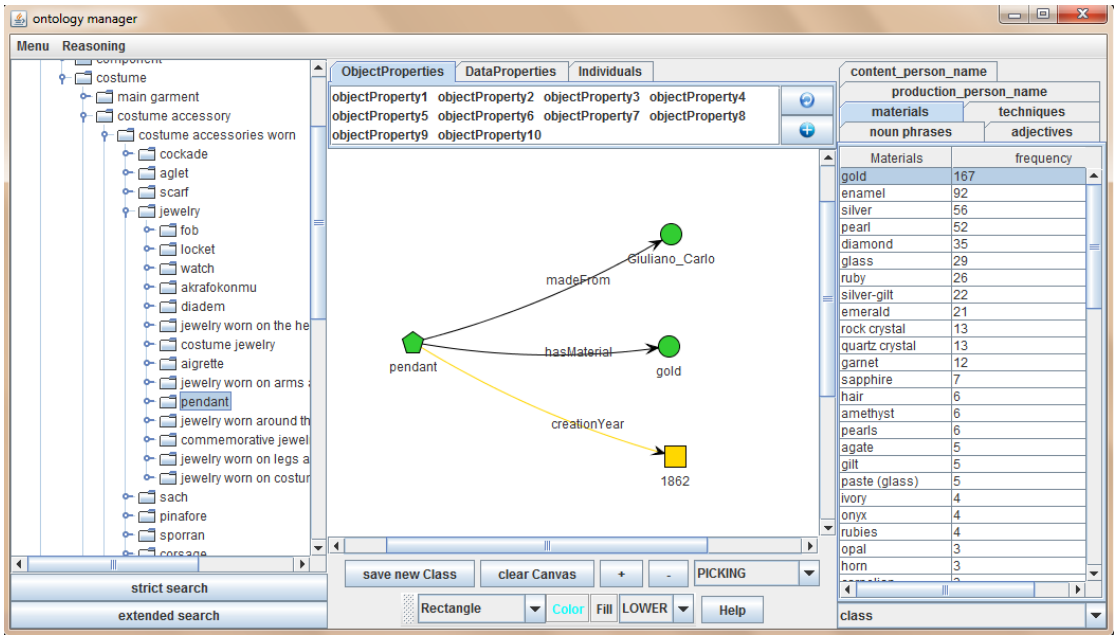
- Δεν είναι ενεργοποιημένος ο mysql server
- Τα στοιχεία σύνδεσης είναι λανθασμένα (πχ username, password, port κλπ)

Πιθανές επιλογές και λειτουργίες τους:

- retry: Με τα ίδια στοιχεία σύνδεσης, επιχειρείται ξανά η σύνδεση με τον server.
- change connectivity info: Δημιουργείται διάλογος (Εικόνα 5.2-1) για την αλλαγή των στοιχείων σύνδεσης.
- ignore: Παραβλέπεται η αδυναμία σύνδεσης και προχωρά κανονικά η λειτουργία της εφαρμογής.

Εικόνα 5.2-1 Διάλογος αλλαγής στοιχείων σύνδεσης στη βάση δεδομένων

5.3 Συνολική μορφή εφαρμογής



Αυτή είναι η συνολική εικόνα της εφαρμογής. Στο αριστερό μέρος γίνεται απεικόνιση της οντολογίας σε δενδρική δομή. Με τα κουμπιά strict και extended search γίνεται η

αναζήτηση στιγμιότυπων στη βάση δεδομένων και η αυτόματη εξαγωγή όρων από τις Συλλογές Κειμένων που δημιουργούνται. Στο πάνω μέρος απεικονίζονται τα Object Properties, τα Data Properties και τα Individuals της οντολογίας. Στο δεξί μέρος παρουσιάζονται τα αποτελέσματα από την αυτόματη εξαγωγή όρων, ενώ στο κέντρο είναι ο καμβάς όπου μπορούμε να ορίσουμε νέες κλάσεις σχηματίζοντας γράφους.

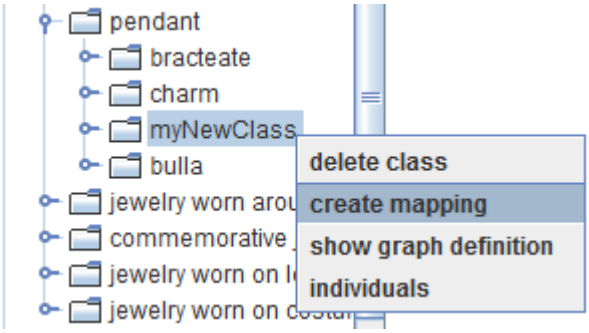
5.4 Απεικόνιση οντολογίας

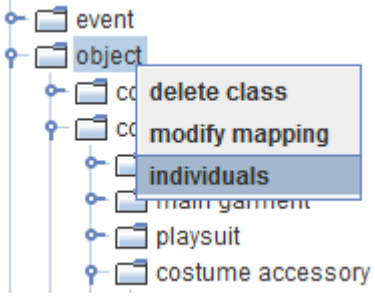
Στην αριστερή μεριά της εφαρμογής φαίνεται σε δενδρική δομή η οντολογία που έχουμε φορτώσει. Κάθε κόμβος αντιπροσωπεύει μία κλάση και πρέπει να έχει ένα όνομα, ώστε να μπορεί να την διακρίνει ο χρήστης. Το αναγνωριστικό μιας κλάσης είναι το IRI (Internationalized Resource Identifier) της. Όμως αυτό είναι συνήθως μεγάλο και δυσανάγνωστο και δεν είναι ενδεικτικό της σημασίας της κλάσης. Παραδείγματος χάριν, η κλάση “statue” μπορεί να έχει IRI "http://thesaurus.europeanaart.eu/thesaurus/statue" ή ακόμα και "http://thesaurus.europeanaart.eu/thesaurus/10142". Ως εκ τούτου, αποφασίστηκε τα ονόματα των κλάσεων που εμφανίζονται να προκύπτουν ως εξής: αν υπάρχει κάποιο annotation για την κλάση αυτή (πχ rdfs:label), τότε αυτό να χρησιμοποιείται ως όνομα. Διαφορετικά, θα χρησιμοποιείται το IRI της.

Πατώντας πάνω σε κάποιο κόμβο ρωτάμε τον reasoner (hermit) ποιες είναι οι άμεσες υποκλάσεις αυτής της κλάσης και δημιουργούμε τους αντίστοιχους κόμβους. Αυτή η λειτουργία γίνεται δυναμικά, διότι η οντολογία μπορεί να αλλάζει κατά την διάρκεια της εφαρμογής (ορισμός καινούργιων κλάσεων ή διαγραφή ήδη υπαρχόντων). Έτσι, γίνονται άμεσα αντιληπτές οι αλλαγές της οντολογίας.

Επίσης, επειδή μια κλάση μπορεί να βρίσκεται σε περισσότερα από ένα σημεία στο δέντρο της οντολογίας (υποκλάση διαφορετικών κλάσεων), έχει προστεθεί η εξής λειτουργία: όταν προσπαθούμε να επεκτείνουμε μια κλάση (προβάλλουμε τις υποκλάσεις τις στο δέντρο της οντολογίας) που έχει ήδη επεκταθεί σε άλλο σημείο του δέντρου, να μην επιτρέπεται και να γίνεται αυτόματη μεταφορά σε εκείνο το σημείο. Έτσι, δεν γίνεται χαοτική η απεικόνιση της οντολογίας με επαναλαμβανόμενα τμήματα υποδέντρων.

Όταν επιλέξουμε μια κλάση και στην συνέχεια πατήσουμε δεξί κλικ τότε εμφανίζονται οι ακόλουθες επιλογές:





Όπως φαίνεται από τις παραπάνω εικόνες, οι διαθέσιμες επιλογές μπορεί να διαφέρουν.

- delete class: Με αυτή την επιλογή διαγράφεται η συγκεκριμένη κλάση. Όπως έχει αναφερθεί και προηγουμένως, για την διαχείριση της οντολογίας χρησιμοποιείται το OWL API. Σε αυτό το σημείο είναι σημαντικό να θυμηθούμε πως λειτουργεί η διεπαφή αυτή. Δεν μπορούμε απευθείας να διαγράψουμε μια κλάση, μια ιδιότητα ή ένα άτομο, καθώς η οντολογία δεν περιέχει άμεσα τις οντότητες αυτές, αλλά περιέχει αξιώματα που αναφέρονται σε αυτές. Επομένως, αν θέλουμε να διαγράψουμε μια οντότητα, πρέπει να διαγράψουμε όλα τα αξιώματα που αναφέρονται σε αυτή. Έτσι όμως μπορεί να χαθεί παραπάνω πληροφορία απ' ότι επιθυμούμε, γι' αυτό συνίσταται προσοχή. Για παράδειγμα, έστω ότι έχουμε ορίσει την κλάση C με το αξίωμα EquivalentClasses(C ObjectIntersectionOf(A ObjectSomeValuesFrom(R B))) (σε ΠΛ είναι C≡A ⊓ ∃ R.B) και ότι θέλουμε να διαγράψουμε την κλάση B. Σύμφωνα με τα προηγούμενα, το παραπάνω αξίωμα θα διαγραφεί, καθώς περιέχει αναφορά στην κλάση B. Επομένως, θα χάσουμε και τον ορισμό της κλάσης C και πλέον δεν θα γνωρίζουμε ότι είναι υποσύνολο της A [28].
- create/modify mapping: Αν έχει οριστεί αντιστοίχιση για την συγκεκριμένη κλάση, τότε εμφανίζεται η επιλογή modify, ενώ αν δεν έχει οριστεί και κληρονομεί την αντιστοίχιση κάποιου προγόνου της, τότε εμφανίζεται η επιλογή create.
- show graph definition: Αυτή η επιλογή εμφανίζεται μόνο για τις κλάσεις που έχουν οριστεί μέσα από αυτήν την εφαρμογή. Συγκεκριμένα ανακατασκευάζει τον γράφο ορισμού αυτής της κλάσης, έτσι ακριβώς όπως τον είχε φτιάξει ο χρήστης. Ο γράφος ορισμού όπως και οι αντιστοιχίσεις είναι μόνιμη πληροφορία που μένει και μετά τον τερματισμό της εφαρμογής.

- Individuals: Με αυτή την επιλογή μπορούμε να δούμε ποια άτομα είναι μέλη της συγκεκριμένης κλάσης, καθώς και να κάνουμε καινούργιες εισαγωγές.

5.5 *Εκκίνηση αναζήτησης στιγμιότυπων*

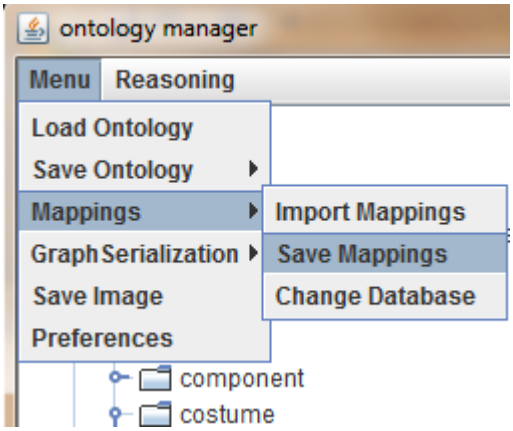
Για να ξεκινήσουμε την αναζήτηση στη βάση για εύρεση στιγμιότυπων και αυτόματη εξαγωγή όρων, πρέπει πρώτα να έχουμε επιλέξει μια κλάση από το δέντρο της οντολογίας μας και μετά να πατήσουμε ένα από τα δύο κουμπιά:

- strict search: Ψάχνει για στιγμιότυπα μόνο αυτής της κλάσης.
- extended search: Ψάχνει για στιγμιότυπα αυτής της κλάσης ή οποιασδήποτε υποκλάσης της.

5.6 *Επεξήγηση αντιστοίχισης με την βάση δεδομένων.*

Με την αντιστοίχιση (mapping) έχουμε την δυνατότητα να εξηγήσουμε στην εφαρμογή πώς ταιριάζει η οντολογία με την βάση δεδομένων στην οποία θα κάνουμε αναζητήσεις. Συγκεκριμένα, θέλουμε να έχουμε έναν ευέλικτο τρόπο που να εξηγεί πώς θα γίνεται η αναζήτηση για στιγμιότυπα της κάθε κλάσης και η εξαγωγή των γλωσσικών τους περιγραφών.

Αρχικά, πρέπει να ορίσουμε την βάση δεδομένων που θα χρησιμοποιήσουμε για την οντολογία αυτή. Από το Menu -> Mappings -> Change Database συμπληρώνουμε τα στοιχεία της βάση με την οποία θα δουλέψουμε. Ο ορισμός των στοιχείων σύνδεσης με τη βάση όπως και οι αντιστοιχίσεις των κλάσεων αποθηκεύονται και έτσι ξαναχρησιμοποιούνται αυτόματα χωρίς να επαναλαμβάνεται η διαδικασία. Αν θέλουμε, βέβαια, μπορούμε να δημιουργήσουμε καινούργιες αντιστοιχίσεις ή να έχουμε πολλές παράλληλα και να επιλέγουμε κάθε φορά ποια θα χρησιμοποιήσουμε.

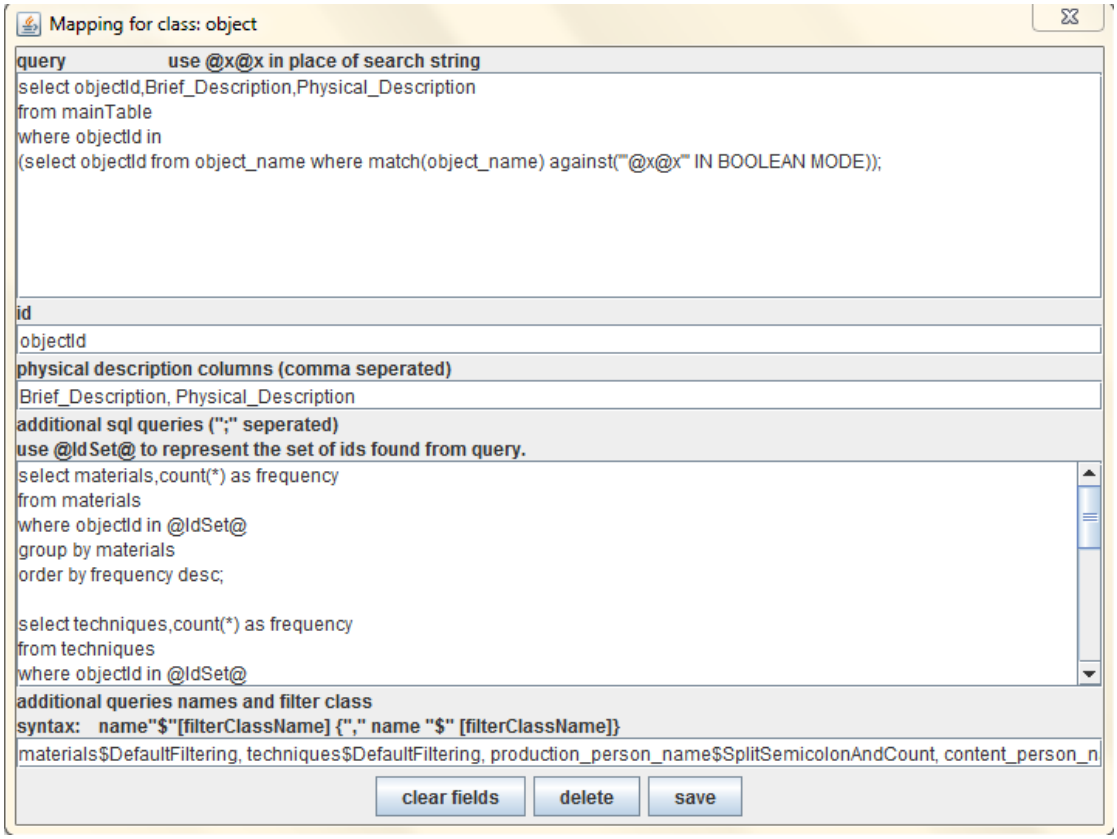


Στη γραμμή μενού επιλέγοντας Menu->Mappings εμφανίζονται οι παρακάτω επιλογές:

- Save Mappings: με αυτή την επιλογή αποθηκεύονται οι αντιστοιχίσεις όλων των κλάσεων, καθώς και τα στοιχεία σύνδεσης με τη βάση, σε ένα αρχείο με σειριοποίηση των αντίστοιχων δομών. Είναι πολύ πιθανό να μην χρειαστεί η συγκεκριμένη λειτουργία, καθώς εκτός από χειροκίνητα γίνεται και αυτόματα, όταν αποθηκεύουμε την οντολογία. Αν δηλαδή αποθηκεύσουμε την οντολογία μας με όνομα: c:\myFolder\myOntology.owl, τότε αυτόματα αποθηκεύονται και οι αντιστοιχίσεις που χρησιμοποιούνται με όνομα αρχείου c:\myFolder\myOntologyMapppings.ser
- Import Mappings: διαβάζεται το αρχείο που επιλέγουμε και γίνεται αποσειριοποίηση. Και αυτή η διαδικασία γίνεται αυτόματα, όταν επιλέξουμε να φορτώσουμε ένα αρχείο οντολογίας (ψάχνει στον ίδιο φάκελο να βρει κάποιο αρχείο με όνομα που να ακολουθεί την παραπάνω σύμβαση).
- Change Database: αλλάζει τα στοιχεία σύνδεσης με τη βάση δεδομένων.

5.6.1 Δημιουργία-Τροποποίηση αντιστοιχίσεων

Επιλέγοντας μια κλάση και πατώντας δεξί κλικ-> (create|modify) Mapping εμφανίζεται το παρακάτω παράθυρο.



Εικόνα 5.6-1 Ορισμός ή τροποποίηση mapping

Στο πεδίο query γράφουμε τον τρόπο με τον οποίο αναζητούμε στιγμιότυπα της κλάση αυτής στη βάση δεδομένων. Εδώ γράφουμε ένα οποιοδήποτε sql ερώτημα που να ταιριάζει στη βάση δεδομένων που χρησιμοποιούμε.

Η συμβολοσειρά @x@x είναι μια ειδική λέξη που μπορεί να φανεί πολύ χρήσιμη για να ορίσουμε μαζικά τις αντιστοιχίσεις όλων των κλάσεων της οντολογίας. Η λέξη αυτή, όπου συναντάται στο ερώτημα query, αντικαθιστάται από το όνομα της κλάσης και στη συνέχεια εκτελείται το ερώτημα στη βάση δεδομένων. Επομένως, είναι σημαντικό να χρησιμοποιείται αυτή η συγκεκριμένη λέξη και όχι απευθείας το όνομα της κλάσης, ώστε να μπορεί να κληρονομηθεί σωστά η αντιστοίχιση και στους απογόνους της.

Στο πεδίο id ορίζουμε ποια στήλη του αποτελέσματος είναι το αναγνωριστικό του αντικειμένου.

Στο πεδίο physical description columns ορίζουμε ποιες στήλες του αποτελέσματος θα χρησιμοποιηθούν για την δημιουργία του εγγράφου που αργότερα θα χρησιμοποιήσουμε για την αυτόματη εξαγωγή όρων. Δηλαδή ενώνει το περιεχόμενο των αντίστοιχων στηλών σε ένα ενιαίο έγγραφο για κάθε αντικείμενο.

Στο πεδίο additional queries μπορούμε να γράψουμε διάφορα επιπλέον sql ερωτήματα που θέλουμε να εκτελεστούν, ώστε να εξάγουμε επιπρόσθετη πληροφορία από τη βάση δεδομένων. Εδώ, δεν στοχεύουμε σε αδόμητες γλωσσικές περιγραφές, αλλά σε δομημένη ή ημιδομημένη πληροφορία της βάσης, που αφού περάσει από κάποιο φίλτρο μετασχηματισμού και τροποποίησης, παρουσιάζεται στον χρήστη (δηλαδή χωρίς να υποστεί επεξεργασία φυσικής γλώσσας και αυτόματη εξαγωγή όρων). Με τη λέξη κλειδί @IdSet@ δίνεται η δυνατότητα να περιοριστεί η αναζήτηση μόνο στα αντικείμενα που επιστράφηκαν από το προηγούμενο ερώτημα (query).

Το παράδειγμα που ακολουθεί εξηγεί αναλυτικά την όλη διαδικασία. Έστω ότι κάνουμε αυστηρή αναζήτηση (strict search) στην κλάση pendant, τότε το ερώτημα που θα εκτελεστεί είναι το εξής:

```
select objectId,Brief_Description,Physical_Description
 from mainTable
 where objectId in
 (select objectId from object_name where match(object_name)
against('"pendant"' IN BOOLEAN MODE));
```

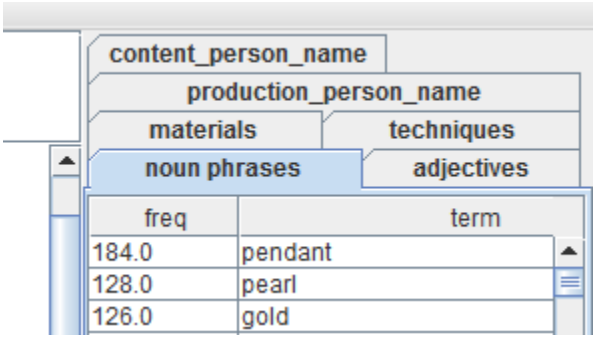
Έστω ότι τα αποτελέσματα του ερωτήματος είναι τρία, δηλαδή το result set περιλαμβάνει τα objectId: 3, 4 και 8. Το πρώτο additional query της παραπάνω εικόνας θα γίνει:

```
select materials,count(*) as frequency
from materials
where objectId in ('3','4','8')
group by materials
order by frequency desc
```

Είναι σαν να αντικαθιστούμε την λέξη @IdSet@ με το query σαν εμφωλευμένο ερώτημα και να επιλέγουμε μόνο τη στήλη με τα αναγνωριστικά. Όμως τώρα είναι πιο ευανάγνωστο και πιο αποδοτικό, καθώς δεν επαναλαμβάνει το ερώτημα στη βάση.

Το τελευταίο πεδίο είναι το additional queries names and filter class.

Εδώ, γράφουμε τα ονόματα που θέλουμε να έχουν τα additional queries του προηγούμενου πεδίου. Αυτά εμφανίζονται μαζί με τα αποτελέσματα στη δεξιά μεριά της εφαρμογής ως επικεφαλίδες των αντίστοιχων καρτελών.



Τα ονόματα είναι πριν το σύμβολο του δολαρίου, ενώ μετά το δολάριο είναι το όνομα της κλάσης που θα χρησιμοποιηθεί για μετασχηματισμό των αποτελεσμάτων (filtering). Εδώ, filtering σημαίνει μετασχηματισμός του ResultSet που επιστρέφεται από το ερώτημα στη βάση δεδομένων σε ένα TableModel. Μπορούμε να αλλάξουμε τον αλγόριθμο filtering που χρησιμοποιείται για κάθε additional query δυναμικά, χωρίς να κάνουμε καμία αλλαγή στον κώδικα, ούτε επανεκκίνηση της εφαρμογής. Η κλάση που θα χρησιμοποιηθεί πρέπει φυσικά να υλοποιεί την κατάλληλη διαπροσωπεία για αυτή την δουλεία (λεπτομέρειες στο κεφάλαιο υλοποίησης).

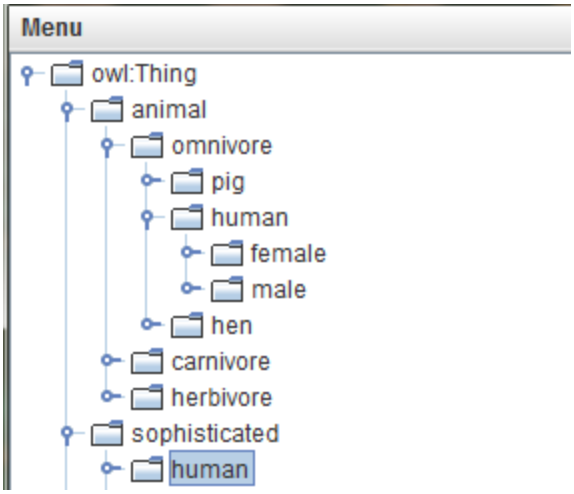
Το πλήθος των καρτελών που εμφανίζονται στη δεξιά μεριά είναι μεταβαλλόμενο και εξαρτάται από τις αντιστοιχίσεις που έχουμε ορίσει και τις αναζητήσεις που κάνουμε.

5.6.2 Κληρονομικότητα των αντιστοιχίσεων

Επειδή το να ορίσουμε αντιστοίχιση για κάθε κλάση είναι κοπιαστικό και μη αποδοτικό, χρησιμοποιείται ο παρακάτω αλγόριθμος:

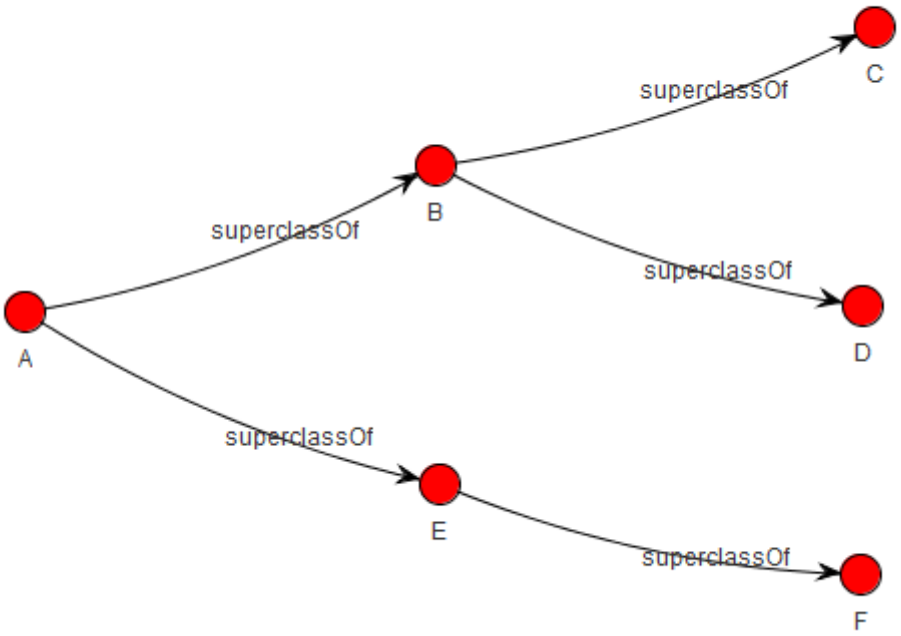
Αν έχει οριστεί αντιστοίχιση για την συγκεκριμένη κλάση τότε χρησιμοποίησέ την.

Διαφορετικά, χρησιμοποίησε την αντιστοίχιση της άμεσης υπερκλάσης της (από το δέντρο της οντολογίας στο αριστερό πλαίσιο).



Εδώ χρησιμοποιείται το mapping του sophisticated και όχι του omnivore.

Αναλυτικό παράδειγμα:



Έστω ότι έχουμε την παραπάνω ιεραρχία κλάσεων. Για την κλάση A ορίζουμε την αντιστοίχιση (firstMapping), ενώ για την κλάση E ορίζουμε την αντιστοίχιση (secondMapping). Αν κάνουμε εκτεταμένη αναζήτηση στην κλάση A, τότε θα εκτελεστεί το firstMapping για τις κλάσεις A,B,C,D και το secondMapping για τις κλάσεις E,F.

Έστω ότι η αναζήτηση στιγμιότυπων για το A επιστρέφει το σύνολο αναγνωριστικών Id={1,3,4} ,για το B το Id={2,5}, για το C το Id={10}, και για το D το Id={1,2}.

Όπως εξηγήθηκε προηγουμένως, για κάθε αντικείμενο που ταιριάζει στην αναζήτηση δημιουργούμε ένα έγγραφο και το προσθέτουμε στη Συλλογή Κειμένων. Παρατηρούμε όμως ότι μερικά αναγνωριστικά μπορεί να επιστραφούν περισσότερες από μία φορές (πχ Id=1 και Id=2). Για καθένα από αυτά θα δημιουργήσουμε μόνο ένα έγγραφο και όχι παραπάνω. Αυτό διότι αφού πρόκειται για το ίδιο αναγνωριστικό και για την ίδια αντιστοίχιση, τα κείμενα που θα προκύψουν θα είναι ακριβώς τα ίδια. Έστω επίσης ότι το secondMapping για το E επιστρέφει το Id={20} και για το F το Id={1,21}. Το Id=1 το έχουμε ξανασυναντήσει και στο firstMapping. Παρόλα αυτά θα δημιουργήσουμε καινούργιο έγγραφο, αφού πρόκειται για διαφορετική αντιστοίχιση, επομένως και το έγγραφο που θα δημιουργηθεί θα είναι διαφορετικό (πχ διαφορετικές στήλες ή διαφορετικός πίνακας από τη βάση δεδομένων).

Στα αντίστοιχα additional queries που θα εκτελεστούν για το firstMapping, το @IdSet@ θα αντικατασταθεί από το εξής string: "('1','2','3','4','5','10')" δηλαδή από την ένωση όλων των ανακτημένων αναγνωριστικών.

Η τελική Συλλογή Κειμένων που θα δημιουργηθεί, θα είναι η ένωση από όλα τα έγγραφα του firstMapping και όλα τα έγγραφα του secondMapping.

5.7 Επεξεργασία της Συλλογής Κειμένων και άντληση πληροφορίας

Αφού πλέον έχει δημιουργηθεί η Συλλογή Κειμένων με όλα τα έγγραφα, είμαστε έτοιμοι να τα επεξεργαστούμε και να κάνουμε αυτόματη εξαγωγή σημαντικών όρων που πιθανόν να μας βοηθήσουν να δημιουργήσουμε καινούργιες κλάσεις στην οντολογία μας.

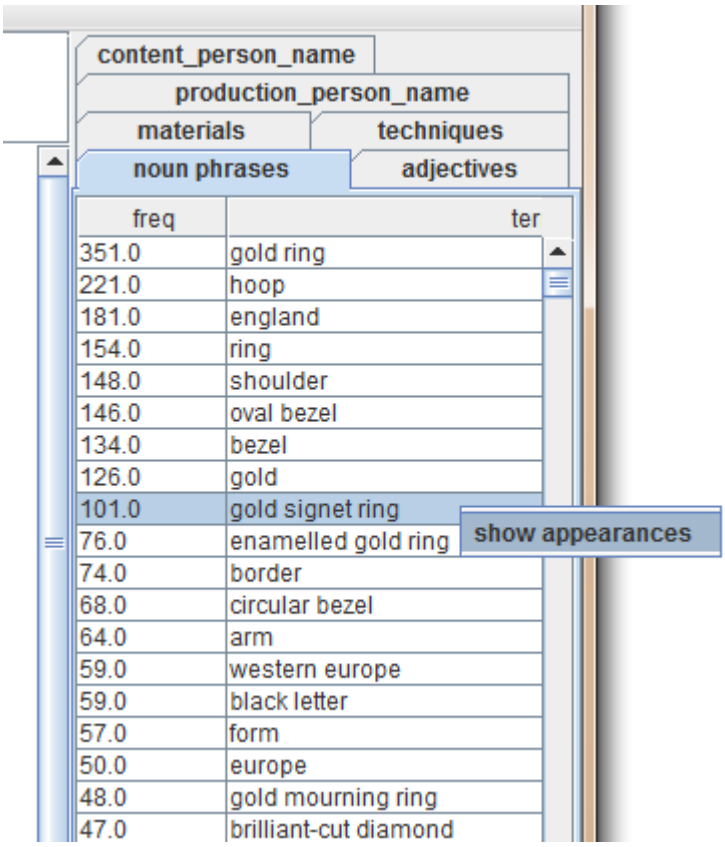
Για την επεξεργασία των κειμένων έχει χρησιμοποιηθεί μια τροποποιημένη έκδοση του εργαλείου Jate (Java Automatic Term Extraction). Μερικές τροποποιήσεις που έχουν γίνει στο εργαλείο αυτό είναι οι εξής:

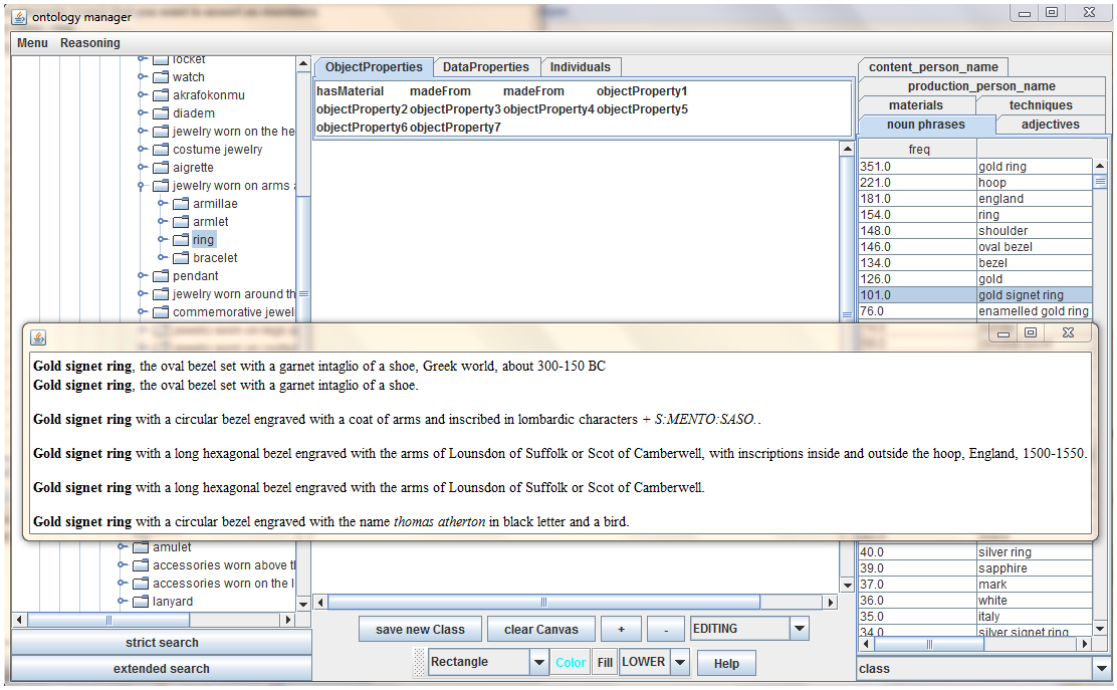
- Εκτός από εξαγωγή ονοματικών φράσεων του έχουμε προσθέσει την δυνατότητα να κάνει και εξαγωγή επιθέτων.
- Εκτός από απλή καταμέτρηση των εμφανίσεων των όρων, του έχουμε προσθέσει την λειτουργία να αποθηκεύει και τις θέσεις εντοπισμού του κάθε

όρου. Έτσι μπορούμε να ανακτήσουμε γρήγορα, ανά πάσα στιγμή τις προτάσεις μέσα στις οποίες εμφανίζεται ο συγκεκριμένος όρος.

- Το εργαλείο αυτό λειτουργεί με απλά αρχεία κειμένου. Δηλαδή παίρνει σαν είσοδο το μονοπάτι ενός φακέλου και αρχίζει και αναλύει όλα τα αρχεία που εντοπίζονται μέσα σε αυτόν. Το έχουμε τροποποιήσει ώστε να λειτουργεί με συμβολοσειρές, που τις βλέπει σαν εικονικά αρχεία, και τις παίρνει από την βάση δεδομένων.
- Τροποποιήσεις σε ορισμένα αρχεία ώστε να μπορεί να υποστηρίζει πολυνηματισμό (η επεξεργασία της Συλλογής Κειμένων για ονοματικές φράσεις και επίθετα γίνεται παράλληλα).

Επιλέγοντας έναν οποιοδήποτε όρο από την καρτέλα των ονοματικών φράσεων ή από την καρτέλα των επιθέτων και πατώντας δεξί κλικ -> show appearances, εμφανίζονται οι προτάσεις στις οποίες εντοπίζεται ο συγκεκριμένος όρος.





Όπως έχει εξηγηθεί και στο κεφάλαιο 2.2.2.2 υπάρχουν διάφοροι αλγόριθμοι που μπορούν να χρησιμοποιηθούν για την εξαγωγή όρων. Το εργαλείο jate έχει υλοποιημένους πολλούς από αυτούς. Έγιναν πειράματα, δοκιμές και αξιολόγηση των αποτελεσμάτων με όλους. Αυτή τη στιγμή όμως η εφαρμογή χρησιμοποιεί από προεπιλογή τον simple term frequency αλγόριθμο.

Όπως στα additional queries, έτσι και εδώ, τα αποτελέσματα που επιστρέφονται από την εξαγωγή όρων περνούν από ένα επίπεδο μετασχηματισμού. Σε αυτό το επίπεδο μπορούμε να ελέγξουμε ποιους όρους θα προτείνουμε στο χρήστη, με ποια σειρά ή να κάνουμε κάποια επεξεργασία των αποτελεσμάτων. Ο αλγόριθμος που εφαρμόζεται μπορεί να αλλάζει δυναμικά, κατά την διάρκεια λειτουργίας από το menu -> preferences -> nounFilteringClass ή adjectiveFilteringClass. Όπως αναφέρθηκε και στα additional queries, οι νέοι αλγόριθμοι του χρήστη ενσωματώνονται στην εφαρμογή χωρίς να απαιτείται καμία αλλαγή στον κώδικά της.

5.8 Ιδιότητες αντικειμένων, ιδιότητες τύπων δεδομένων, άτομα

Στο πλαίσιο πάνω από τον καμβά εμφανίζονται τρεις καρτέλες: μια για τις ιδιότητες αντικειμένων (Object Properties), μια για τις ιδιότητες τύπων δεδομένων (Data

Properties) και μια για τα άτομα (Individuals) της οντολογίας. Τα περιεχόμενα αυτών των καρτελών μπορούν να χρησιμοποιηθούν για την κατασκευή του γράφου. Στην δεξιά μεριά έχουμε δύο επιλογές:

- refresh: ψάχνει την οντολογία και βρίσκει όλες τις ιδιότητες (ή όλα τα άτομα) που χρησιμοποιούνται και τις προβάλλει στη καρτέλα. Έτσι ενημερώνεται για πιθανές αλλαγές όπως νέες εισαγωγές ή διαγραφές.
- add: με αυτή την επιλογή εμφανίζεται κατάλληλος διάλογος για την δημιουργία καινούργιων ιδιοτήτων (ή ατόμων) από τον χρήστη.

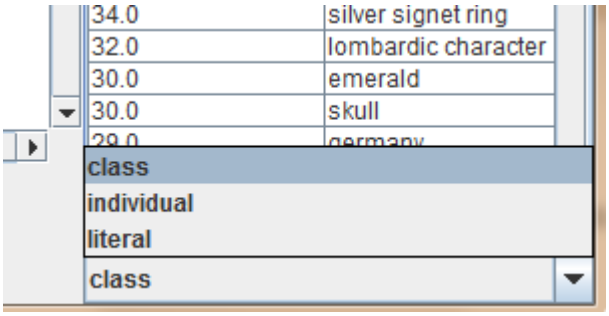
5.9 Ορισμός καινούργιων κλάσεων με γραφικό τρόπο.

Ένας από τους πρωταρχικούς λόγους δημιουργίας αυτής της εφαρμογής είναι ο γρήγορος, εύκολος, γραφικός τρόπος ορισμού νέων κλάσεων της οντολογίας, υποβοηθούμενος από την επεξεργασία φυσικής γλώσσας. Για αυτό το λόγο δόθηκε ιδιαίτερο βάρος στο γραφικό κομμάτι. Για την δημιουργία και απεικόνιση των γράφων έχει χρησιμοποιηθεί το **Jung** (Java Universal Network/Graph Framework)[29].

5.9.1 Σύρσιμο και απόθεση

Ουσιαστικά, μπορούμε να σύρουμε (drag) οτιδήποτε φαίνεται στην οθόνη και να το αποθέσουμε (drop) πάνω στον καμβά. Έτσι, μπορούμε να δημιουργήσουμε έναν γράφο που αντιπροσωπεύει τον ορισμό της νέας κλάσης.

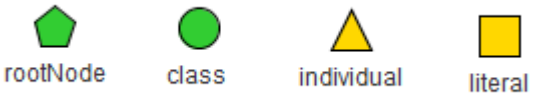
Από το αριστερό πλαίσιο (δενδρική απεικόνιση οντολογίας) μπορούμε να τραβήξουμε τις απεικονιζόμενες κλάσεις. Από το δεξί πλαίσιο μπορούμε να τραβήξουμε το προτεινόμενο περιεχόμενο σε τρεις διαφορετικές μορφές: ως κλάση, άτομο ή λεκτικό.



Τέλος, μπορούμε να τραβήξουμε οτιδήποτε υπάρχει στο πάνω πλαίσιο (ιδιότητες αντικειμένων, ιδιότητες τύπων δεδομένων, άτομα).

5.9.2 Στοιχεία γράφου

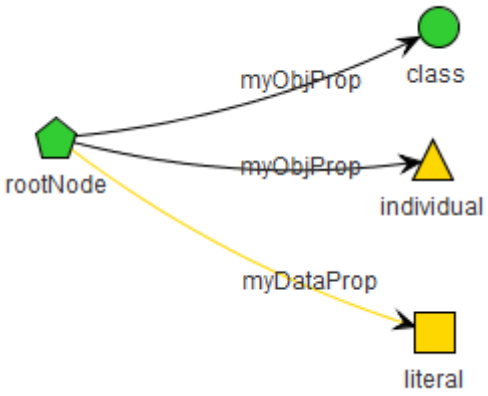
Για να ξεχωρίζουν εύκολα οι διαφορετικοί τύποι κόμβων και ακμών, χρησιμοποιούνται διαφορετικά σχήματα και χρώματα. Τύποι κόμβων:



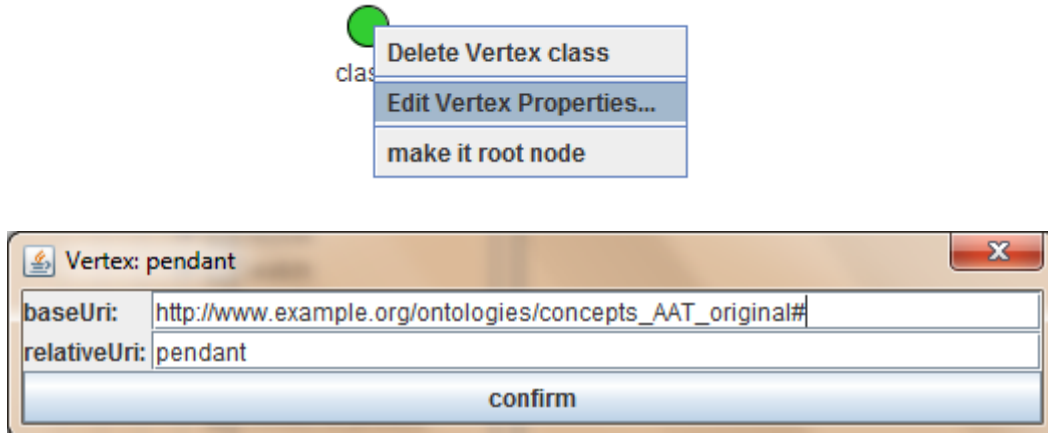
- rootNode: Σε κάθε γράφο πρέπει να υπάρχει ακριβώς ένας rootNode. Αυτός ο κόμβος δείχνει από πού ξεκινάει η μετάφραση του γράφου σε owl έκφραση. RootNode μπορεί να γίνει μια κλάση ή μια τομή κλάσεων πατώντας πάνω της δεξί κλικ -> make it rootNode.
- class: συμβολίζεται με πράσινο κύκλο και μπορεί να αντιπροσωπεύει μια κλάση ή τομή κλάσεων.
- individual: τα άτομα της οντολογίας μας
- literal: αυτά είναι απλά λεκτικά που έχουν όμως κάποιο τύπο που δείχνει πώς να ερμηνεύονται από τις εφαρμογές (σαν integer, double, float, string κοκ)

Τύποι ακμών:

- Ιδιότητες Αντικειμένων: έχουν μαύρο χρώμα
- Ιδιότητες Τύπων Δεδομένων: κίτρινο χρώμα



Πατώντας δεξί κλικ πάνω σε κάποιο κόμβο-κλάση του γράφου εμφανίζονται διάφορες επιλογές: delete vertex για διαγραφή του, edit vertex properties για αλλαγή του IRI και επομένως της κλάσης που αντιπροσωπεύει, make it root node για να θεωρείται αρχή του γράφου αυτός ο κόμβος.



Εικόνα 5.9-1 Αλλαγή ιδιοτήτων κόμβου

Στο πεδίο relativeUri στην πραγματικότητα γράφουμε την ετικέτα που θέλουμε να έχει ο κόμβος στον γράφο. Επομένως μπορεί να περιέχει και μη αλφαβητικούς χαρακτήρες όπως κενά, παρενθέσεις κλπ. Βέβαια, αυτοί οι χαρακτήρες δεν επιτρέπονται να είναι στο IRI μιας οντότητας. Οπότε το fragment identifier που θα αποδώσουμε στην οντότητα που αντιπροσωπεύει θα είναι το relativeUri αφού αντικατασταθούν όλοι οι μη αποδεκτοί χαρακτήρες από κάτω παύλες '_'. Αν για παράδειγμα στο relativeUri συμπληρώσουμε το "pendant (gold and silver)" τότε το IRI του κόμβου θα είναι

http://www.example.org/ontologies/concepts_AAT_original/canvasElements#pendant_gold_and_silver, ενώ στον γράφο θα εμφανίζεται έτσι:

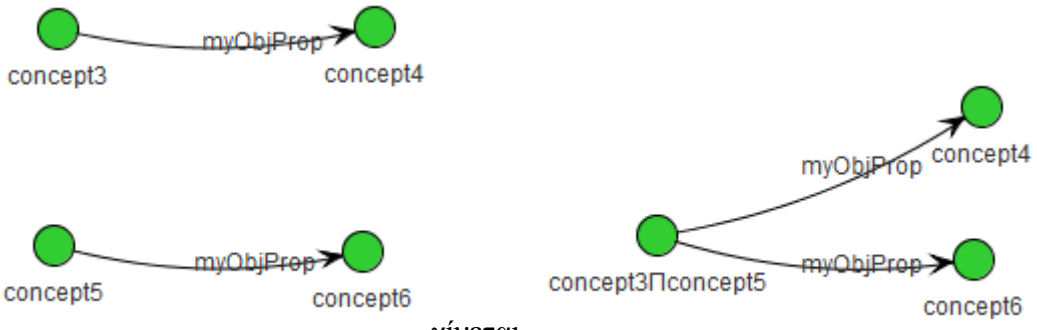
Για τα άτομα, τις ιδιότητες, τα λεκτικά καθώς και τις σύνθετες κλάσεις (πχ τομές κλάσεων) πατώντας δεξί κλικ εμφανίζονται παρόμοιες επιλογές με μικρές διαφοροποιήσεις οπότε δεν θα αναλυθούν.

5.9.3 Δημιουργία τομής κλάσεων

Για να φτιάξουμε τομή κλάσεων το μόνο που έχουμε να κάνουμε είναι να αφήσουμε (drop) την μεταφερόμενη κλάση πάνω σε κάποιον ήδη υπάρχον κόμβο. Αν έχουμε δυο κόμβους πάνω στον καμβά και θέλουμε να δημιουργήσουμε την τομή τους, τότε τραβάμε (σε picking mode) τον έναν κόμβο πάνω στον άλλο και γίνεται συγχώνευση. Αν αυτοί οι κόμβοι πριν την συγχώνευσή τους είχαν ακμές, τότε αυτές μεταφέρονται στον καινούργιο κόμβο της ένωσης.

γίνεται

Παράδειγμα με κόμβους που έχουν ήδη ακμές



γίνεται

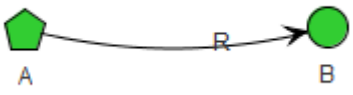
5.9.4 Αντιστοίχιση γράφου σε έκφραση owl

Η εκφραστικότητα που παρέχουμε είναι αυτή της OWL EL, με την διαφοροποίηση όμως ότι υποστηρίζουμε και αντίστροφους ρόλους.

Κόμβοι που αντιστοιχούν σε άτομα ή λεκτικά έχουν προφανή ερμηνεία. Οι ακμές ερμηνεύονται σαν πλήρεις υπαρξιακοί περιορισμοί. Επομένως, οι κόμβοι που αντιπροσωπεύουν κλάσεις και έχουν ακμές έχουν την ακόλουθη ερμηνεία: τομή της κλάσης που αντιπροσωπεύουν (ή της τομής κλάσεων που αντιπροσωπεύουν) με τους υπαρξιακούς περιορισμούς που εισάγουν οι ακμές. Η ερμηνεία γίνεται από τα φύλλα προς την ρίζα.

Ακολουθούν μερικά παραδείγματα μαζί με την ερμηνεία τους σε Περιγραφική Λογική και σε Functional-Style Syntax.

1) newClass ≡ A ⊓ ∃ R.B

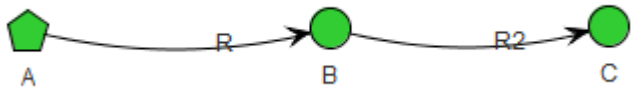



```
EquivalentClasses(<http://www.example.org/ontologies/concepts_
AAT_original#newClass>
ObjectIntersectionOf(<http://www.example.org/ontologies/concep
```

```
ts_AAT_original/canvasElements#A>
ObjectSomeValuesFrom(<http://www.example.org/ontologies/concep
ts_AAT_original#R>
<http://www.example.org/ontologies/concepts_AAT_original/canva
sElements#B>)))
```

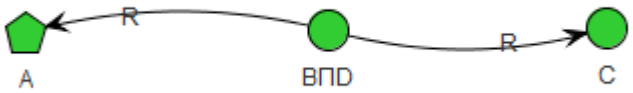
Στα επόμενα παραδείγματα δεν θα αναγράφεται ολόκληρο το IRI των οντοτήτων αλλά μόνο το fragment.

2) newClass ≡ A ⊓ ∃ R.(B ⊓ ∃ R2.C)




```
EquivalentClasses(newClass   ObjectIntersectionOf(A
ObjectSomeValuesFrom(R   ObjectIntersectionOf(B
ObjectSomeValuesFrom(R2   C)))) )
```

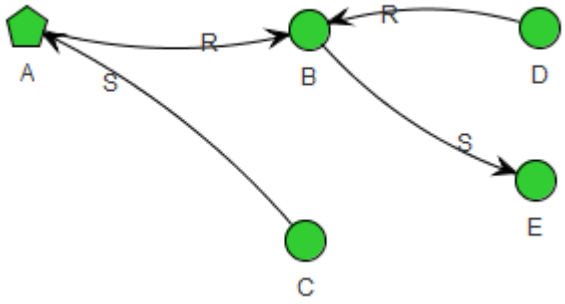
3) newClass ⊑ A ⊓ ∃ R^{-1}.(B ⊓ D ⊓ ∃ R.C)




```
SubClassOf(newClass   ObjectIntersectionOf(A
ObjectSomeValuesFrom(InverseOf(R)
ObjectIntersectionOf(ObjectIntersectionOf(B   D)
ObjectSomeValuesFrom(R   C)))))
```

Παρατήρηση: όταν μία ακμή έχει φορά προς τη ρίζα και όχι προς τα φύλλα τότε παίρνουμε την αντίστροφη ιδιότητα από αυτή που αντιπροσωπεύει.

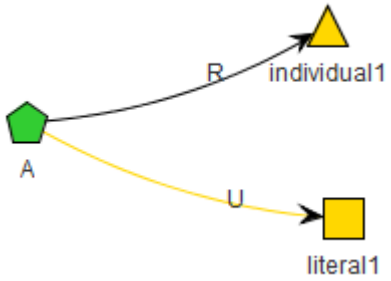
4) newClass ⊑ A ⊓ ∃ S^{-1}.C ⊓ ∃ R.(B ⊓ ∃ S.E ⊓ ∃ R^{-1}.D)




```
SubClassOf(newClass   ObjectIntersectionOf(A
ObjectSomeValuesFrom(R   ObjectIntersectionOf(B
ObjectSomeValuesFrom(S   E)
ObjectSomeValuesFrom(InverseOf(R)   D)))
ObjectSomeValuesFrom(InverseOf(S)   C)))
```

Παράδειγμα με άτομα και λεκτικά:

5) newClass ≡ A ⊓ ∃ R.{individual1} ⊓ ∃ U.”literal1”




```
EquivalentClasses(newClass   ObjectIntersectionOf(A
ObjectHasValue(R   individual1)
DataHasValue(U   "literal1"^^xsd:string)) )
```

5.9.5 Αποθήκευση της νέας κλάσης

Αφού σχηματιστεί ο επιθυμητός γράφος που αντιπροσωπεύει την έκφραση owl που θέλουμε, επιλέγουμε save new class από το κάτω πλαίσιο. Αρχικά, γίνονται ορισμένοι έλεγχοι ορθότητας του γράφου και αν είναι σωστός τότε δημιουργείται ο παρακάτω διάλογος:

Εδώ, ορίζουμε το IRI της νέας κλάσης και αποφασίζουμε αν θα είναι ισοδύναμη ή υποκλάση της έκφρασης owl που αντιπροσωπεύει ο γράφος.

Η νέα κλάση είναι πλέον ορατή και στο δέντρο οντολογίας μας στο αριστερό πλαίσιο. Επίσης, αποθηκεύεται και ο γράφος ορισμού αυτής της κλάσης, ώστε σε επόμενες χρήσεις της εφαρμογής να μπορούμε να τον ανακτήσουμε και πιθανόν τροποποιήσουμε. Οι γράφοι ορισμού όλων των νέων κλάσεων αποθηκεύονται με σειριοποίηση των κατάλληλων δομών. Η αποθήκευση γίνεται επιλέγοντας Menu->Graph Serialization ->Save Graph Serialization ή αυτόματα όταν απλά αποθηκεύσουμε την οντολογία μας. Κατά την αυτόματη αποθήκευση, δημιουργείται στον ίδιο φάκελο με την οντολογία ένα επιπλέον αρχείο με κατάληξη GraphSer.ser.

5.9.6 Έλεγχοι ορθότητας του γράφου ορισμού

Για να είναι σωστός ένας γράφος ορισμού νέας κλάσης, πρέπει να τηρούνται ορισμένες προϋποθέσεις:

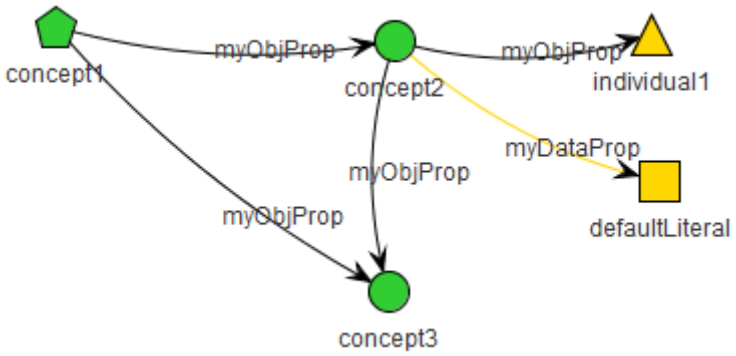
- Πρέπει να υπάρχει ένας κόμβος ρίζα (rootNode). Δηλαδή να έχει οριστεί από ποιο σημείο αρχίζει η ερμηνεία του γράφου σε έκφραση owl.
- Ο γράφος πρέπει να είναι ασθενά συνεκτικός (όλος ο γράφος μια συνιστώσα).
- Ο γράφος των κλάσεων (δηλαδή ο γράφος χωρίς τα άτομα και τα λεκτικά και χωρίς να λαμβάνουμε υπ' όψιν την φορά των ακμών) πρέπει να είναι δέντρο. Δηλαδή δεν μπορεί να περιέχει κύκλους· ανεξάρτητα της φοράς των ακμών.
- Επίσης πρέπει τα άτομα και τα λεκτικά να είναι υποχρεωτικά μόνο φύλλα του δέντρου (όχι ενδιάμεσοι κόμβοι).
- Οι ακμές μεταξύ κόμβων κλάσεων ή κλάσεων και ατόμων πρέπει να είναι τύπου Object Property, ενώ μεταξύ κλάσεων και λεκτικών να είναι τύπου Data Property.

Αν κάποια από αυτές τις προϋποθέσεις δεν πληρείται, τότε επιλέγοντας save new class, τυπώνεται κατάλληλο μήνυμα σφάλματος που εντοπίζει το πρόβλημα.

5.9.6.1 *Παραδείγματα λανθασμένων ορισμών και διαγνωστικά μηνύματα.*

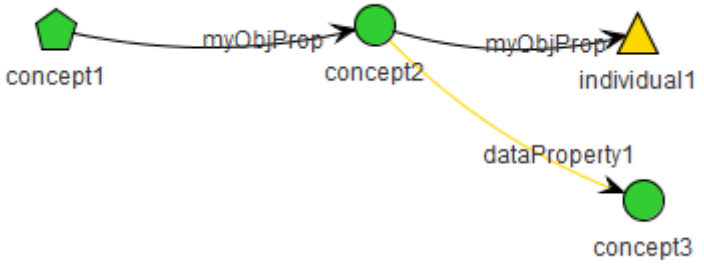
Σε κάθε πιθανό σφάλμα στο σχεδιασμό του γράφου, τυπώνονται κατάλληλα διαγνωστικά μηνύματα. Ενδεικτικά παρουσιάζονται μερικά:

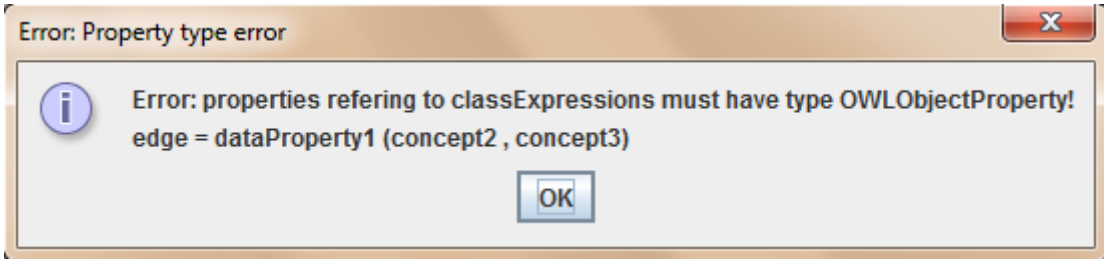
Ο παρακάτω γράφος περιέχει κύκλο.



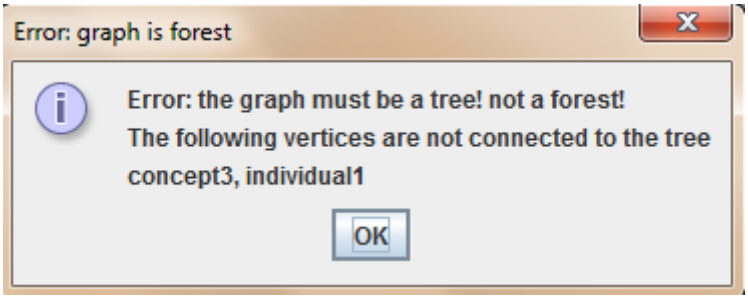
Μήνυμα σφάλματος

Λάθος τύπος ακμής μεταξύ κόμβων:



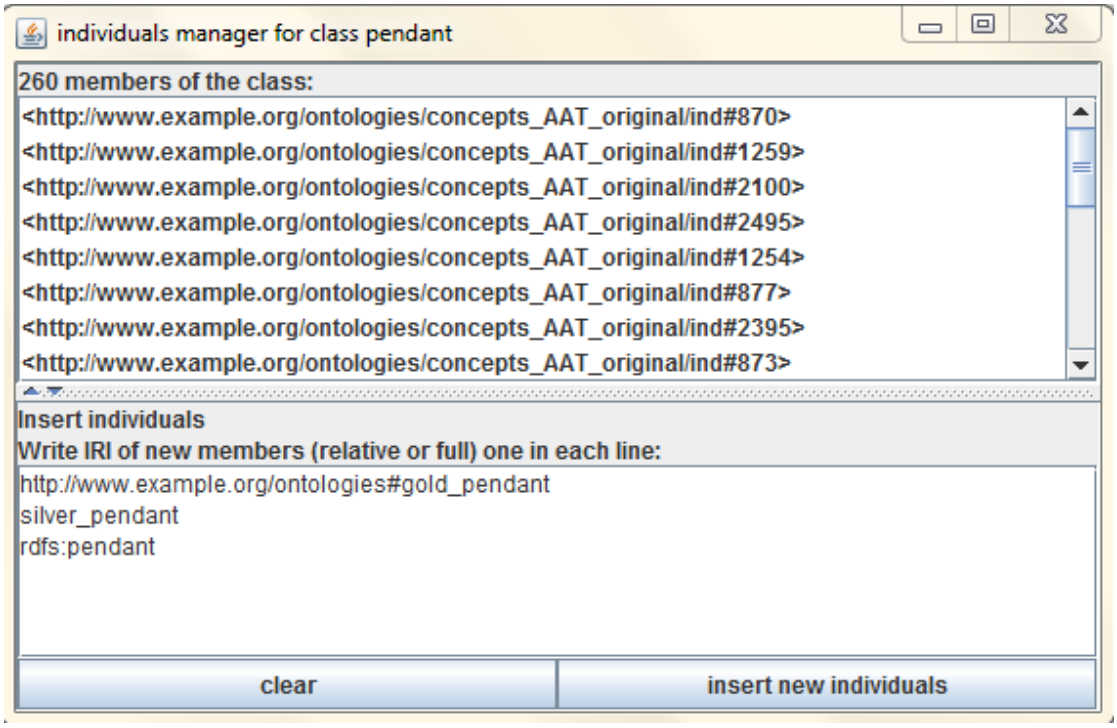


Δεν είναι ασθενά συνεκτικός γράφος:



5.10 Διαχείριση ατόμων οντολογίας

Όταν εργαζόμαστε πάνω σε μία οντολογία, είναι ιδιαίτερα χρήσιμο να μπορούμε να βρούμε ποια είναι τα μέλη της κάθε κλάσης, αλλά και να μπορούμε να εισάγουμε και καινούργια στιγμιότυπα. Επιλέγοντας μια κλάση από το δέντρο οντολογίας μας και πατώντας δεξί κλικ-> individuals εμφανίζεται το παρακάτω παράθυρο διαχείρισης ατόμων.



Στο πάνω μέρος εμφανίζονται όλα τα στιγμιότυπα αυτής της κλάσης. Στο κάτω μέρος μπορούμε να γράψουμε τα άτομα που θέλουμε να εισάγουμε ως νέα μέλη. Κάθε άτομο που εισάγεται πρέπει να μπαίνει σε ξεχωριστή γραμμή. Υπάρχουν τρείς επιλογές:

- Να δοθεί ολόκληρο το IRI του ατόμου όπως γίνεται με το gold_pendant (δηλαδή να αρχίζει με http://)
- Να δοθεί μόνο το fragment identifier (δηλαδή να μην αρχίζει με http και να μην περιέχει τον χαρακτήρα ":"), όπως γίνεται με το silver_pendant. Τότε το IRI που χρησιμοποιείται για το νέο άτομο είναι ontologyIRI + "/ind" + fragment. Οπότε το silver_pendant αντιστοιχεί στο http://www.example.org/ontologies/concepts_AAT_original/ind#silver_pendant.
- Να δοθεί το IRI με χρήση προθέματος (πχ rdfs:pendant)

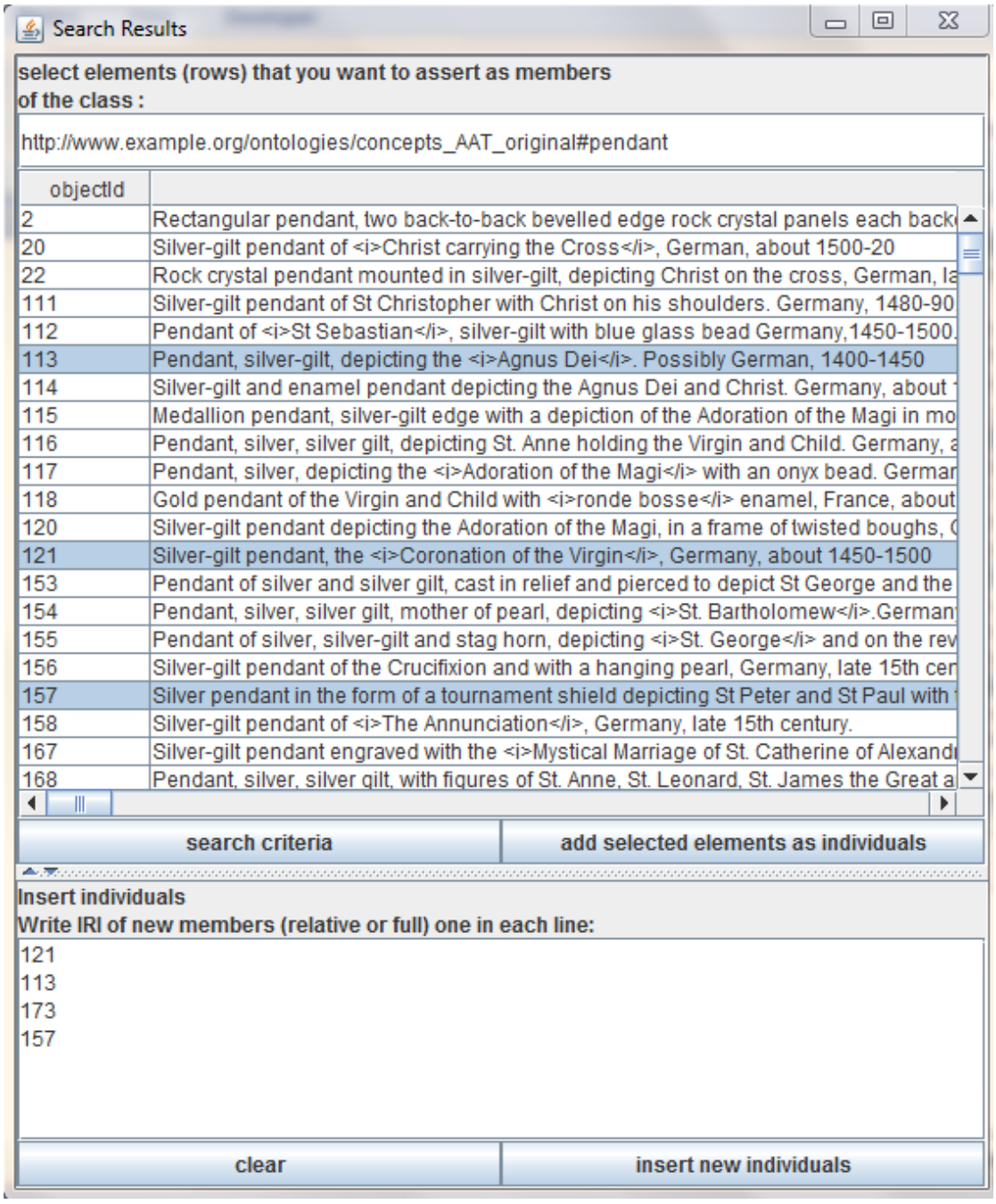
5.10.1 Ταξινόμηση αντικειμένων σε εξειδικευμένες κλάσεις με ημιαυτόματο τρόπο

Όταν επιλέξουμε αυστηρή ή εκτεταμένη αναζήτηση, τότε εξάγουμε όλες τις καταχωρήσεις που μας ενδιαφέρουν από τη βάση δεδομένων μαζί με τις γλωσσικές τους περιγραφές. Αυτές οι περιγραφές δεν χρησιμοποιούνται μόνο για την αυτόματη εξαγωγή όρων, αλλά προβάλλονται και σε ξεχωριστό παράθυρο για να μπορεί να τις διαβάσει ο χρήστης (Εικόνα 5.10-1). Στο πάνω μέρος του παραθύρου φαίνονται τα αναγνωριστικά των καταχωρήσεων και οι γλωσσικές τους περιγραφές. Μπορούμε να επιλέξουμε όποια καταχώρηση θέλουμε και με το κουμπί "add" να την εισάγουμε ως νέο μέλος της κλάσης.

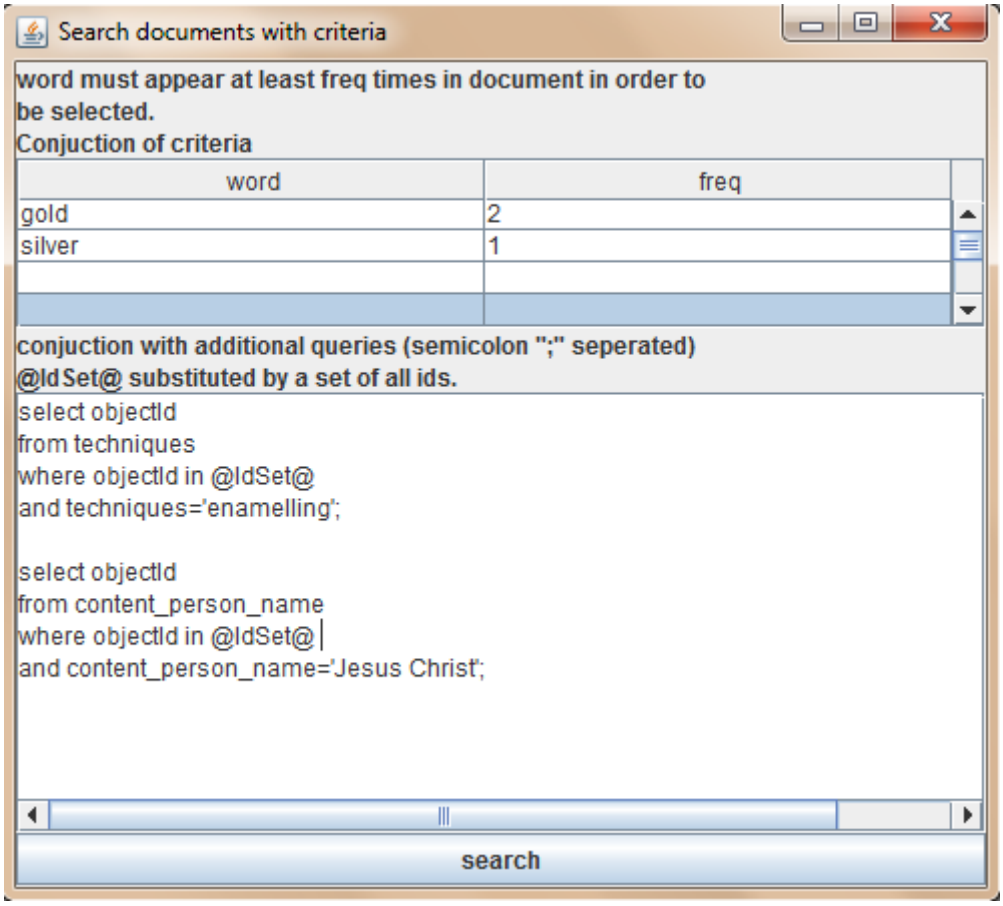
Η αναζήτηση αυτή μπορεί να γίνει και αυτόματα επιλέγοντας το κουμπί "search criteria". Με αυτή την επιλογή εμφανίζεται διάλογος (Εικόνα 5.10-2) όπου μπορούμε να ορίσουμε ποιες προϋποθέσεις πρέπει να πληρούνται για να επιλεγεί ένα αντικείμενο ως νέο μέλος της κλάσης. Στο πάνω μέρος γράφουμε τους όρους που θέλουμε να περιέχουν οι γλωσσικές τους περιγραφές και σε ποια ελάχιστη συχνότητα. Στο κάτω μέρος γράφουμε τα ερωτήματα sql που θέλουμε να ικανοποιούν τα υποψήφια αντικείμενα (δομημένη πληροφορία).

Το παράδειγμα της εικόνας επιλέγει από το σύνολο των αντικειμένων που βρήκαμε στη βάση εκείνα που περιέχουν στις γλωσσικές τους περιγραφές τη λέξη gold με

συχνότητα τουλάχιστον 2, την λέξη silver με συχνότητα τουλάχιστον 1 και που το αναγνωριστικό τους υπάρχει στο ResultSet και των δύο sql ερωτημάτων.



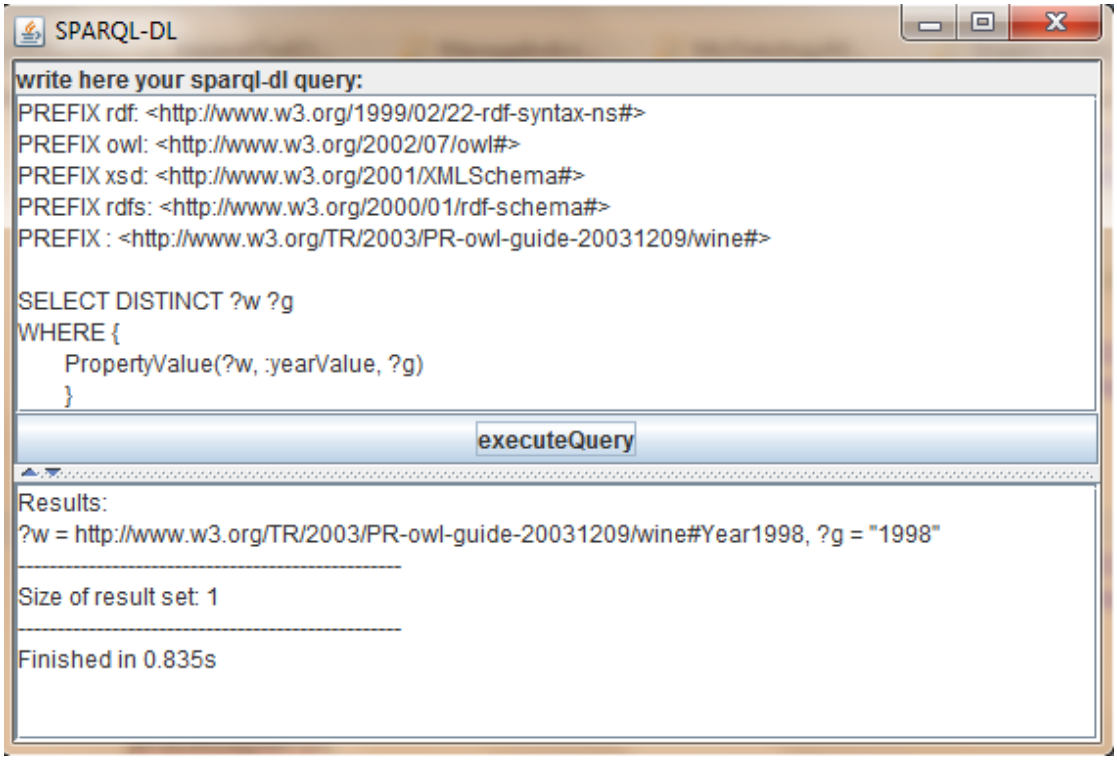
Εικόνα 5.10-1 Ταξινόμηση στιγμιότυπων



Εικόνα 5.10-2 Αναζήτηση στιγμιότυπων με κριτήρια

5.11 Έλεγχος συνέπειας, SPARQL-DL ερωτήματα

Στη γραμμή μενού->Reasoning έχουμε δύο επιλογές: check consistency και sparql-dl. Με την επιλογή check consistency ρωτάμε τον reasoner αν η οντολογία μας είναι συνεπής. Ανάλογα με το αποτέλεσμα κατάλληλο μήνυμα τυπώνεται στην οθόνη. Η επιλογή sparql-dl δημιουργεί το διάλογο της παρακάτω εικόνας. Στο πάνω μέρος ο χρήστης γράφει το ερώτημα που θέλει να εκτελέσει, και στο κάτω μέρος εμφανίζονται τα αποτελέσματα του ερωτήματος.



Η SPARQL-DL[30][31] είναι μια γλώσσα εκτέλεσης ερωτημάτων (query language) πάνω σε οντολογίες. Γενικά οι γλώσσες εκτέλεσης ερωτημάτων χωρίζονται σε δύο κατηγορίες: αυτές που βασίζονται στην RDF (RDF-based QLs όπως είναι οι: RDQL, SeRQL, SPARQL) και αυτές που βασίζονται στις Περιγραφικές Λογικές (DL-based). Η SPARQL-DL είναι DL-based και τα κύρια χαρακτηριστικά της είναι τα εξής:

- Πολύ καλά ορισμένη σημασιολογία που βασίζεται στο μοντέλο των Περιγραφικών Λογικών.
- Δυνατή και εκφραστική γλώσσα που μπορεί να συνδυάσει ταυτόχρονα Tbox, Rbox και Abox ερωτήματα.
- Συμβατή με την SPARQL για να βοηθά την διαλειτουργικότητα των εφαρμογών του Σημασιολογικού Ιστού.
- Είναι υποσύνολο της SPARQL.
- Λειτουργεί πάνω από υπάρχοντες OWL-DL reasoners όπως ο Pellet και προσφέρει την επιπλέον λειτουργικότητα.

6

Λεπτομέρειες Υλοποίησης

Σε αυτό το κεφάλαιο θα αναλυθούν ορισμένα ζητήματα υλοποίησης της εφαρμογής. Προγραμματιστικά εργαλεία και βιβλιοθήκες που χρησιμοποιήθηκαν καθώς και μέθοδοι και τεχνικές για επίλυση επιμέρους προβλημάτων. Θα γίνει μια αναφορά σε επιλεγμένες μόνο κλάσεις και τις λειτουργίες τους.

6.1 Εργαλεία

Τα σημαντικότερα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής είναι τα εξής:

- Γλώσσα προγραμματισμού: java
- Ολοκληρωμένο περιβάλλον ανάπτυξης: eclipse
- Διαχείριση οντολογιών: OWL API
- Απεικόνιση γράφων: JUNG (Java Universal Network/Graph Framework)
- Αυτόματη εξαγωγή όρων: Jate toolkit (Java Automatic Term Extraction)
- Βάση Δεδομένων: MySQL

6.2 Υλοποίηση

6.2.1 Αλγόριθμος μετασχηματισμού των αποτελεσμάτων των επιπρόσθετων ερωτημάτων (additional queries)

Το συγκεκριμένο κεφάλαιο αναφέρεται πρώτο, καθώς είναι πολύ σημαντικό, όχι μόνο για όποιον θέλει να κατανοήσει τον τρόπο λειτουργίας της εφαρμογής, αλλά και για τον απλό χρήστη που θέλει να επεκτείνει την λειτουργία της προσθέτοντας δικούς του αλγόριθμους φιλτραρίσματος.

Όπως έχει εξηγηθεί, για να είναι ευέλικτη η εφαρμογή και προσαρμόσιμη σε οποιοδήποτε σχήμα βάσης δεδομένων· έχει προστεθεί ένα επίπεδο απομόνωσης μεταξύ άντλησης των δεδομένων και απεικόνισής τους. Για να βάλει ο χρήστης τον δικό του αλγόριθμο σε αυτό το σημείο και όχι κάποιον έτοιμο· πρέπει να φτιάξει μια κλάση που να υλοποιεί τη διαπροσωπεία rightPanelsFiltering.Filtering. Τη νέα αυτή κλάση μπορεί να την τοποθετήσει οπουδήποτε, αν και υπάρχει ειδική συσκευασία: rightPanelsFiltering για αυτή τη δουλειά. Δεν χρειάζεται να κάνει τίποτα άλλο, ούτε να τροποποιήσει σε κάποιο σημείο τον κώδικα. Στη συνέχεια απλά εκκινεί την εφαρμογή και τροποποιεί τις αντιστοιχίσεις (Εικόνα 5.6-1) (μετά το σήμα του δολαρίου βάζει το όνομα της νέας του κλάσης). Η διαπροσωπεία είναι όσο πιο γενική γίνεται ώστε να είναι ευέλικτη:

```
public interface Filtering {
      TableModel filtering(ResultSet rs);
      int getColumnNumWithContent();
}
```

Η μέθοδος filtering παίρνει σαν παράμετρο το ResultSet που επιστρέφει η MySQL από την εκτέλεση του ερωτήματος και το μετασχηματίζει σε ένα TableModel, δηλαδή σε πίνακα. Έτσι, ο πίνακάς μας μπορεί να έχει μεταβλητό πλήθος στηλών καθώς και να επεξεργαστεί τα αποτελέσματα πριν τα προβάλλει.

Η δεύτερη μέθοδος καθορίζει ποια στήλη του πίνακα που δημιουργήσαμε έχει το περιεχόμενο που θέλουμε να μεταφέρεται με την χειρονομία drag and drop, καθώς τραβάμε κατά γραμμές.

Δεν χρειάζεται καμία τροποποίηση ο κώδικα της εφαρμογής, διότι το περιβάλλον χρόνου εκτέλεσης της Java (Java Runtime Environment) βρίσκει την νέα κλάση του

χρήστη με βάση το όνομα. Δηλαδή από την συμβολοσειρά του ονόματος που γράφει στις αντιστοιχίσεις [32]:

```
Class<?> clazz = Class.forName(className);
Constructor<?> ctor = clazz.getConstructor(String.class);
Object object = ctor.newInstance(new Object[] { ctorArgument
});
```

Σχετικά αρχεία: package rightPanelsFiltering

6.2.2 Αλγόριθμος μετασχηματισμού των εξαγμένων όρων

Αυτά που αναφέρονται εδώ αφορούν τις καρτέλες των ονοματικών φράσεων και των επιθέτων του δεξιού πλαισίου. Και εδώ ισχύουν τα ίδια με παραπάνω. Δηλαδή μπορούμε να αλλάξουμε τον αλγόριθμο που χρησιμοποιείται και να ορίσουμε καινούργιο. Εδώ όμως είναι λίγο διαφορετική η διαπροσωπεία που πρέπει να υλοποιήσουμε (textProcessingFiltering.Filtering):

```
public interface Filtering {
      TableModel filtering(Term[] terms);
      int getColumnNumWithContent();
}
```

Η μέθοδος filtering παίρνει σαν παράμετρο έναν πίνακα από όρους και επιστρέφει ένα TableModel. Το αντικείμενο Term είναι αυτό που επιστρέφει το εργαλείο Jate μετά την ανάλυση. Κάθε αντικείμενο Term αποτελείται από δύο πράγματα: τον εξαγμένο όρο που είναι μια συμβολοσειρά (ονοματική φράση ή επίθετο) και το confidence του όρου αυτού που είναι ένας πραγματικός αριθμός και δείχνει την εμπιστοσύνη μας ότι πράγματι αποτελεί σημαντικό όρο της Συλλογής Κειμένων.

Η δεύτερη μέθοδος έχει την ίδια λειτουργία με πριν.

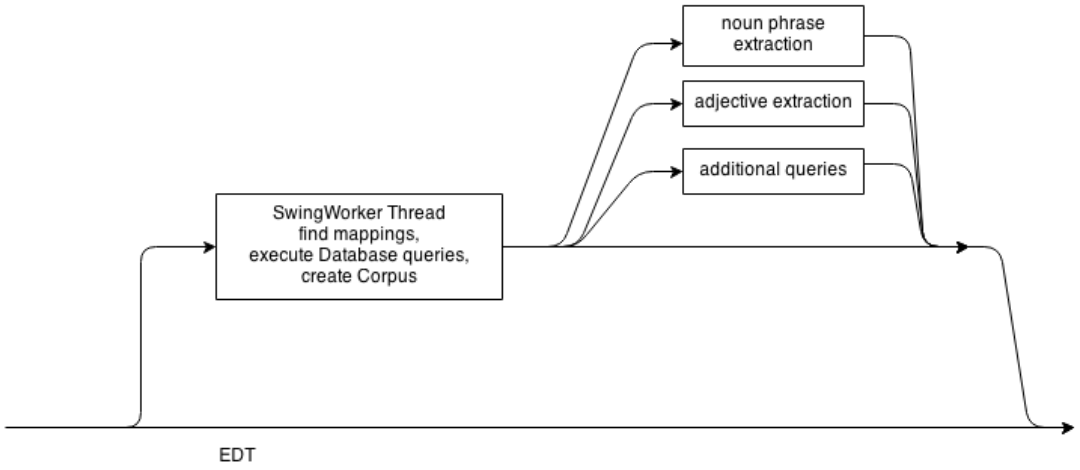
Σχετικά αρχεία: package textProcessingFiltering

6.2.3 Εξαγωγή όρων με παραλληλισμό

Η διαδικασία αναζήτησης στιγμιότυπων και εξαγωγής όρων που περιγράφεται και στην παράγραφο 3.3.1 είναι μια πολύ πολύπλοκη και χρονοβόρα διαδικασία. Συνδυάζει επικοινωνία και συνεργασία πολλών υποσυστημάτων μεταξύ τους, εκτέλεση πολλών ερωτημάτων στη βάση δεδομένων, και τέλος αυτόματη εξαγωγή

όρων από μεγάλες Συλλογές Κειμένων. Αυτό καθιστά αναγκαία την εύρεση λύσεων που θα επιταχύνουν την όλη διαδικασία και θα κάνουν την εφαρμογή συνεχώς αποκρίσιμη στις ενέργειες του χρήστη.

Η λύση που δόθηκε ήταν ο πολυνηματισμός. Μια βασική αρχή του σωστού προγραμματισμού με γραφικές εφαρμογές είναι το Event Dispatch Thread (EDT) να μην εκτελεί ποτέ υπολογιστικά βαριές διεργασίες, ώστε να είναι πάντα διαθέσιμο και να αποκρίνεται στις ενέργειες του χρήστη.



Εικόνα 6.2-1 Πολυνηματισμός κατά την εξαγωγή όρων

Όπως φαίνεται και στο παραπάνω σχήμα, εφαρμόστηκε πολυνηματισμός σε πολλά σημεία. Όταν ο χρήστης ξεκινά την αναζήτηση (strict ή extended search) το EDT δημιουργεί ένα καινούργιο νήμα που βρίσκει τις αντιστοιχίσεις, εκτελεί τα ερωτήματα στη βάση δεδομένων και κατασκευάζει την Συλλογή Κειμένων. Μετά, το νήμα αυτό δημιουργεί τρία νέα νήματα και τους αναθέτει τις επιμέρους εργασίες: εξαγωγή ονοματικών φράσεων, εξαγωγή επιθέτων και εκτέλεση των επιπρόσθετων ερωτημάτων στη βάση δεδομένων.

Για την δημιουργία του πρώτου νήματος χρησιμοποιείται η κλάση SwingWorker [33][34]. Είναι μια ειδική κλάση για πολυνηματισμό σε γραφικές εφαρμογές. Συγκεκριμένα, διευκολύνει την διαδικασία παράλληλου προγραμματισμού σε σχέση με τα απλά νήματα της java, καθώς απλοποιεί την επικοινωνία με το EDT και την αποστολή ενδιάμεσων και τελικών αποτελεσμάτων.

Σχετικά αρχεία:

Package databaseQuery:

- ExpandOwlClassNew.java: Ουσιαστικά αυτό το αρχείο υλοποιεί το υποσύστημα Search Database for Instances-create Corpus και είναι το SwingWorker thread που περιγράφεται παραπάνω.
- ExpandOwlClassAux.java: Είναι ένα βοηθητικό αρχείο που αναλαμβάνει να βρει όλες τις αντιστοιχίσεις των κλάσεων που μας ενδιαφέρουν για την αναζήτηση και τη δημιουργία της Συλλογής Κειμένων.
- AdditionalQueriesThread.java: Είναι το παράλληλο νήμα που εκτελεί τα επιπρόσθετα ερωτήματα στη βάση δεδομένων. Επίσης, βρίσκει τον αλγόριθμο φιλτραρίσματος που θα χρησιμοποιηθεί για κάθε ερώτημα και ειδοποιεί το υποσύστημα απεικόνισης για τα νέα δεδομένα.
- DatabaseManager.java: Αυτό το αντικείμενο αναλαμβάνει όλη την επικοινωνία με τη βάση δεδομένων. Όλα τα ερωτήματα στη βάση εκτελούνται μέσα από αυτό. Επίσης, έχει όλα τα στοιχεία σύνδεσης με τη βάση δεδομένων.

6.2.4 Υλοποίηση drag and drop

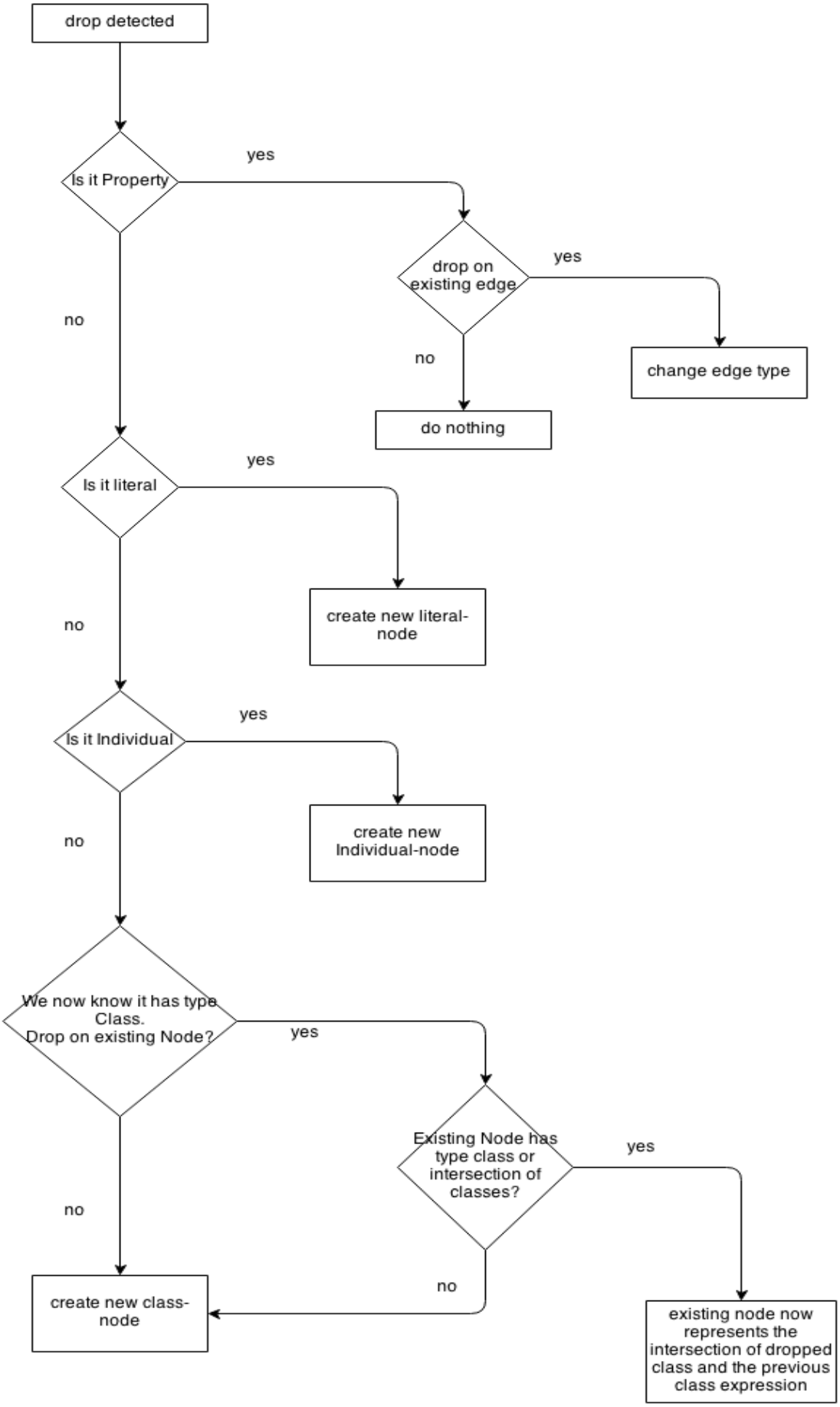
Η λειτουργία drag and drop είναι ένας εύκολος και διαισθητικός τρόπος για μεταφορά πληροφορίας μεταξύ διαφορετικών υποσυστημάτων μιας εφαρμογής. Η java έχει υλοποιημένες διάφορες κλάσεις που διευκολύνουν την υλοποίησή του, παρόλα αυτά πρέπει να τις προσαρμόσουμε στις ανάγκες μας.[35][36]. Οι βασικές ενέργειες που πρέπει να κάνουμε είναι οι εξής:

- Για όλα τα αντικείμενα απ' όπου σκοπεύουμε να κάνουμε σύρσιμο (drag), δηλαδή να εξάγουμε πληροφορία, πρέπει να ενεργοποιήσουμε την αντίστοιχη λειτουργία. Επίσης, πρέπει να προσδιορίσουμε τις ενέργειες που θα υποστηρίζει το συγκεκριμένο αντικείμενο (πχ αντιγραφή, μετακίνηση). Τέλος, πρέπει να εξηγήσουμε με ποιο τρόπο αντλείται η πληροφορία από το αντικείμενο και πακετάρεται σε εξαγόμενη μορφή.
- Για όλα τα αντικείμενα που πρόκειται να δεχτούν μεταφέρσιμη πληροφορία (drop), πρέπει να ενεργοποιήσουμε την αντίστοιχη λειτουργία και να προσδιορίσουμε πώς συμπεριφέρεται στους διαφόρους τύπους δεδομένων που του αποστέλλονται.

Σχετικά αρχεία:

Package dragAndDropAux:

- JTreeTransferHandler.java: Χρησιμοποιείται για τη μεταφορά κλάσεων από τη δενδρική αναπαράσταση της οντολογίας στον καμβά. Προσδιορίζει πώς βρίσκουμε την κλάση που πρέπει να μεταφερθεί και την πακετάρει κατάλληλα, καθώς και τις ενέργειες που υποστηρίζονται που είναι μόνο αντιγραφή.
- JTableTransferHandler.java: Χρησιμοποιείται από το Extracted Information Panel για μεταφορά πληροφορίας. Εδώ, η μεταφερόμενη πληροφορία μπορεί να είναι κλάση, άτομο ή λεκτικό. Η συμβολοσειρά που επιλέγεται για μεταφορά μετατρέπεται μέσω του Ontology Manager στον επιλεγμένο τύπο και μετά πακετάρεται σε κατάλληλη μορφή.
- JListTransferHandler.java: κάνει παρόμοια δουλειά για τα components JList, που είναι τα πλαίσια που απεικονίζουν τα Object Properties και τα Data Properties.
- SupportedFlavors.java: Ορίζει ποιοι είναι οι υποστηριζόμενοι τύποι δεδομένων για μεταφορά, μεταφέρει τα δεδομένα, και διαπραγματεύεται με τον παραλήπτη των δεδομένων σε τι μορφή θα του τα στείλει.
- VVDropTargetListener.java: Χρησιμοποιείται από τον καμβά για να δέχεται τα απεσταλμένα δεδομένα. Ανάλογα με τα δεδομένα που παραλαμβάνει, δημιουργεί καινούργιους κόμβους στο γράφο, κατασκευάζει τομές κλάσεων ή αλλάζει τις υπάρχουσες ακμές (Εικόνα 6.2-2).


drop detected
Is it Property
yes
drop on existing edge
yes
change edge type
no
do nothing
no
Is it literal
yes
create new literal-node
no
Is it Individual
yes
create new Individual-node
no
We now know it has type Class. Drop on existing Node?
yes
Existing Node has type class or intersection of classes?
yes
existing node now represents the intersection of dropped class and the previous class expression
no
no
create new class-node

Εικόνα 6.2-2 Αλγόριθμος επιλογής ενέργειας καμβά σε drop events.

6.2.5 Class Definition Graphs Manager

Όπως έχει ειπωθεί, αυτό το υποσύστημα αναλαμβάνει την αποθήκευση των γράφων ορισμού κλάσεων, την ανάκτησή τους, και τον χειρισμό τους. Η αποθήκευση των γράφων σε μόνιμο μέσο αποφασίστηκε να γίνει με σειριοποίηση (serialization).

Σχετικά αρχεία:

Package graphSerialization:

- GraphSerializationNode.java: Αυτή η κλάση αναλαμβάνει να σειριοποιήσει ολόκληρο τον γράφο. Για να γίνει όμως αυτό πρέπει πρώτα να σειριοποιήσουμε τα δομικά του στοιχεία, δηλαδή τους κόμβους και τις ακμές. Ο κάθε κόμβος σειριοποιείται από την inner class VertexSerialization. Για κάθε κόμβο κρατάμε την εξής πληροφορία:
    - Ένα Point2D που δείχνει τις συντεταγμένες του στον καμβά απεικόνισης.
    - Μια λίστα από URI. Όλοι οι κόμβοι έχουν ένα URI είτε αντιπροσωπεύουν κλάση, είτε άτομο, είτε λεκτικό. Οι κόμβοι όμως που αντιπροσωπεύουν τομές κλάσεων χρησιμοποιούν ένα URI για την κάθε κλάση της τομής.
    - Ένα label που είναι το όνομα του κόμβου στον γράφο.
    - Μια μεταβλητή type που δείχνει τον τύπο του κόμβου (κλάση, άτομο, λεκτικό).

  Η κάθε ακμή σειριοποιείται από την inner class EdgeSerialization. Για κάθε ακμή κρατάμε την εξής πληροφορία:
    - Το URI της ακμής σε μορφή συμβολοσειράς.
    - Το τύπο της ακμής (Object ή Data Property)
    - Τα δύο άκρα της ακμής, δηλαδή η πηγή και ο προορισμός.

  Επομένως για να γίνει η σειριοποίηση ολόκληρου του γράφου, κρατάμε την εξής πληροφορία:
    - Ένα πίνακα με όλους τους σειριοποιημένους κόμβους.
    - Ένα πίνακα με όλες τις σειριοποιημένες ακμές.
    - Ποιος κόμβος είναι η ρίζα του δέντρου (rootNode).

- GraphSerializationManager.java: Αυτή η κλάση φορτώνει, αλλά και αποθηκεύει τα σειριοποιημένα αρχεία στο σύστημα αρχείων του υπολογιστή. Επίσης, κατά την εκτέλεση της εφαρμογής, κρατά σε μια εσωτερική δομή όλους τους γράφους ορισμού. Έτσι, είναι σε θέση να απαντά στα διάφορα υποσυστήματα ποιές κλάσεις έχουν γράφο ορισμού, και αν έχουν, ποιος είναι αυτός. Η δομή που χρησιμοποιείται είναι ένα HashMap με κλειδί το URI της κλάσης.
- GraphSerializationListener.java: Αυτή η κλάση ενεργοποιείται, όταν ο χρήστης επιλέξει να απεικονίσει τον ορισμό μιας κλάσης. Αρχικά, καθαρίζει τον καμβά από υπάρχοντες κόμβους και ακμές, μετά επαναφέρει το σύστημα συντεταγμένων στο φυσιολογικό (χωρίς περιστροφές, στρεβλώσεις και μετακινήσεις) και μετά ανακατασκευάζει τον σειριοποιημένο γράφο ορισμού στον καμβά.

6.2.6 Mappings Manager

Η υλοποίηση αυτού του υποσυστήματος γίνεται στο package mappingToDatabase και έχει δομή παρόμοια με το Class Definition Graphs Manager. Συγκεκριμένα αποτελείται από τις κλάσεις:

- AdditionalQuery.java: Κάθε τέτοιο αντικείμενο αντιπροσωπεύει ένα επιπρόσθετο ερώτημα. Αποθηκεύει το ερώτημα που εκτελείται στη βάση δεδομένων, το όνομα του ερωτήματος αυτού (που είναι και το όνομα που εμφανίζεται στην αντίστοιχη καρτέλα απεικόνισης), και το όνομα της κλάσης που χρησιμοποιείται για filtering των αποτελεσμάτων.
- MappingNode.java: Αποθηκεύει όλες τις πληροφορίες για μια αντιστοίχιση. Δηλαδή ποιο είναι το βασικό ερώτημα εύρεσης στιγμιότυπων στη βάση δεδομένων, ποια στήλη του αποτελέσματος είναι το αναγνωριστικό id, ποιες στήλες περιέχουν τη γλωσσική περιγραφή που θα αναλύσουμε, και φυσικά μια λίστα με τα additionalQueries που θα εκτελεστούν.
- MappingManager.java: Φορτώνει αλλά και αποθηκεύει τα αρχεία σειριοποίησης με τις αντίστοιχες δομές. Κρατά στη μνήμη ένα HashMap με κλειδί το URI της κλάσης και τιμή την αντιστοίχιση που της έχει οριστεί.

- MappingListener.java: Ενεργοποιείται όταν ο χρήστης θέλει να ορίσει ή να τροποποιήσει την αντιστοίχιση μιας κλάσης. Δημιουργεί το παράθυρο διαλόγου Εικόνα 5.6-1.

6.2.7 Υλοποίηση του καμβά σχεδίασης γράφων

Η υλοποίηση του υποσυστήματος γίνεται στο package canvasCode. Σε αυτό το πακέτο υπάρχουν πολλές κλάσεις που ρυθμίζουν τη συμπεριφορά του καμβά σε όλες τις ενέργειες του χρήστη και πολλές που αλλάζουν τον προεπιλεγμένο τρόπο λειτουργίας της βιβλιοθήκης jung. Θα περιγραφούν μόνο οι βασικές:

- MyCanvas.java: Είναι η βασική κλάση του καμβά καθώς αυτή απεικονίζει τον γράφο. Είναι subscriber στο υποσύστημα User Preferences και αλλάζει εμφάνιση με τις προτιμήσεις χρήστη.
- GraphElements.java: Εδώ ορίζονται τι είναι κόμβος και τι ακμή. Δηλαδή τι στοιχεία της οντολογίας μπορούν να αντιπροσωπεύουν.
- SaveNewClassDialog.java: Διάλογος αποθήκευσης νέας κλάσης, επαλήθευση ορθότητας γράφου και παραγωγή κατάλληλων διαγνωστικών μηνυμάτων σφάλματος, εύρεση της OWL έκφρασης που αντιστοιχεί στον γράφο ορισμού.
- UriPropertyDialog: Δημιουργεί τους διαλόγους αλλαγής στοιχείων για τους κόμβους και τις ακμές.

6.2.8 Γραφικά

Όλες οι λειτουργίες της εφαρμογής είναι προσβάσιμες από το γραφικό περιβάλλον. Οι κυριότερες κλάσεις υλοποίησής του είναι στο package **graphics** και **graphicsAux**.

7

Επίλογος

7.1 Σύνοψη και συμπεράσματα

Αυτή η διπλωματική εργασία είχε σκοπό την κατασκευή μιας εφαρμογής γενικής χρήσης που θα απλοποιεί και θα αυτοματοποιεί ορισμένα στάδια της ανάπτυξης οντολογίας. Εξετάστηκαν τα εξής θέματα:

- Πώς μπορούμε να βρίσκουμε και να προτείνουμε περιεχόμενο στο χρήστη που θα του είναι χρήσιμο για τον ορισμό νέων εξειδικευμένων κλάσεων της οντολογίας.
- Πώς να αυτοματοποιείται η διαδικασία ταξινόμησης αντικειμένων σε εξειδικευμένες κλάσεις με βάση τις δομημένες και αδόμητες περιγραφές τους.
- Πώς μπορεί να γίνει ο ορισμός και η διαχείριση της οντολογίας με εύκολο γραφικό τρόπο.

Για την αυτόματη άντληση πληροφορίας χρησιμοποιήθηκαν τεχνολογίες επεξεργασίας φυσικής γλώσσας και αυτόματης εξαγωγής όρων. Έγινε θεωρητική ανάλυση και περιγραφή των σταδίων που ακολουθούνται από την διαδικασία αυτή, και των διαφορετικών αλγορίθμων που μπορούν να χρησιμοποιηθούν. Δημιουργήθηκε μια υποδομή που επιτρέπει στο χρήστη να αντλεί περιεχόμενο από οποιαδήποτε βάση δεδομένων και να την απεικονίζει γραφικά. Η υποδομή αυτή έχει

φτιαχτεί με τρόπο που να είναι εύκολα επεκτάσιμη από τον χρήστη (με δικούς του νέους αλγόριθμους επεξεργασίας των εξαγόμενων δεδομένων) και εύκολα προσαρμόσιμη στις διαφορετικές μορφές των δεδομένων.

Για την διαχείριση των οντολογιών κατασκευάσαμε γραφικό περιβάλλον, μέσω του οποίου ο χρήστης έχει πλήρη έλεγχο της οντολογίας και μπορεί να την εξετάσει και να την τροποποιήσει με ποικίλους τρόπους. Μελετήσαμε πώς μπορεί να γίνει η αντιστοίχιση owl εκφράσεων σε σχήματα γράφων, και υλοποιήσαμε σύστημα που επιτρέπει τον ορισμό νέων κλάσεων μέσω γράφων. Τέλος, κάναμε θεωρητική ανάλυση των Περιγραφικών Λογικών και των γλωσσών αναπαράστασης γνώσης στο διαδίκτυο, τις υπηρεσίες που μας προσφέρουν και την χρησιμότητά τους.

7.2 Μελλοντικές επεκτάσεις

Οι σημαντικότερες επεκτάσεις που μπορούν να γίνουν στην εφαρμογή είναι οι εξής:

- Στην παρούσα έκδοση της εφαρμογής, η αυτόματη εξαγωγή όρων γίνεται εφαρμόζοντας από προεπιλογή τον αλγόριθμο simple term frequency. Για να χρησιμοποιηθεί κάποιος άλλος, απαιτείται μικρή τροποποίηση του κώδικα. Μία επιθυμητή επέκταση θα ήταν να μπορεί ο χρήστης να επιλέγει τον αλγόριθμο μέσω του γραφικού περιβάλλοντος.
- Αυτή τη στιγμή η εκφραστικότητα που παρέχουμε για τον ορισμό νέων κλάσεων είναι αυτός της OWL EL με μικρές τροποποιήσεις. Μια επιθυμητή επέκταση θα ήταν να υποστηρίζουμε μεγαλύτερη εκφραστικότητα, όπως της OWL DL.

8

Βιβλιογραφία

[1] «Semantic Web - W3C,» [Ηλεκτρονικό]. Available: http://www.w3.org/standards/semanticweb/.

[2] T. Berners-Lee, J. Hendler και O. Lassila, «The Semantic Web,» *Scientific American,* αρ. http://www.cs.umd.edu/~golbeck/LBSC690/SemanticWeb.html, May 17, 2001.

[3] F. Baader, D. L. McGuinness, D. Nardi και P. F. Patel-Schneider, THE DESCRIPTION LOGIC HANDBOOK: Theory, implementation, and applications.

[4] «World Wide Web Consortium,» [Ηλεκτρονικό]. Available: http://www.w3.org/.

[5] J. Z. Pan, *Description Logics: reasoning support for the Semantic Web,* phd thesis - University of Manchester , 2004.

[6] B. Glimm, *Querying Description Logic knowledge bases,* phd thesis - University of Manchester , 2007.

[7] F. M. Donini, M. Lenzerini, D. Nardi και A. Schaerf, «Deduction in Concept Languages: From Subsumption to Instance Checking,» *Journal of Logic and Computation,* τόμ. 4, αρ. 4, pp. 423-452, 1994.

[8] D. Beckett, «RDF/XML Syntax Specification,» W3C, 10 February 2004. [Ηλεκτρονικό].

Available: http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

[9] F. Manola και E. Miller, «RDF Primer,» W3C, 10 February 2004. [Ηλεκτρονικό]. Available: http://www.w3.org/TR/rdf-primer/#rdfschema.

[10] P. Hitzler, M. Krotzsch, B. Parsia, P. Patel-Schneider και S. Rudolph, «OWL 2 Web Ontology Language Primer (Second Edition),» W3C, 11 December 2012. [Ηλεκτρονικό]. Available: http://www.w3.org/TR/owl2-primer/.

[11] B. Motik, P. Patel-Schneider και B. Parsia, «OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition),» W3C, 11 December 2012. [Ηλεκτρονικό]. Available: http://www.w3.org/TR/owl2-syntax/.

[12] «W3C EL,» 18 January 2008. [Ηλεκτρονικό]. Available: http://www.w3.org/2007/OWL/wiki/EL.

[13] F. Baader, S. Brandt και C. Lutz, «Pushing the EL Envelope,» σε *Artificial Intelligence*, 2005.

[14] «International health terminology standards development organisation,» [Ηλεκτρονικό]. Available: http://www.ihtsdo.org/.

[15] «National Cancer Institute,» [Ηλεκτρονικό]. Available: http://www.cancer.gov/.

[16] «Gene Ontology website,» [Ηλεκτρονικό]. Available: http://www.geneontology.org/.

[17] «The Stanford Natural Language Processing Group,» Stanford university, [Ηλεκτρονικό]. Available: http://nlp.stanford.edu/index.shtml.

[18] «Apache OpenNLP,» Apache Software Foundation, [Ηλεκτρονικό]. Available: http://opennlp.apache.org/.

[19] «The Dragon Toolkit,» Drexel university, [Ηλεκτρονικό]. Available: http://dragon.ischool.drexel.edu/features.asp.

[20] M. Porter, «An algorithm for suffix stripping,» *Program,* τόμ. 14, αρ. 3, pp. 130-137, 1980.

[21] G. Salton και C. Buckley, «Term-weighting approaches in automatic text retrieval.,» *Information Processing & Management,* τόμ. 24, αρ. 5, pp. 513-523, 1988.

[22] K. Ahmad, L. Gillam και L. Tostevin, «Weirdness indexing for logical document extrapolation and retrieval (wilder).,» σε *The Eighth Text REtrieval Conference (TREC-8).*, 1999.

[23] «British National Corpus,» [Ηλεκτρονικό]. Available: http://www.natcorp.ox.ac.uk/.

[24] «Text REtrieval Conference (TREC),» [Ηλεκτρονικό]. Available: http://trec.nist.gov/.

[25] K. FRANTZI, S. ANANIADOU και H. MIMA, «Automatic Recognition of Multi-Word Terms: the C-value/NC-value Method,» 2000.

[26] «JATE (Java Automatic Term Extraction) toolkit,» [Ηλεκτρονικό]. Available: https://code.google.com/p/jatetoolkit/.

[27] «The OWL API,» [Ηλεκτρονικό]. Available: http://owlapi.sourceforge.net/.

[28] M. Horridge, «owl api, javadoc, entity remover,» The University Of Manchester, Bio-Health Informatics Group, 11 Dec 2006. [Ηλεκτρονικό]. Available: http://owlapi.sourceforge.net/javadoc/org/semanticweb/owlapi/util/OWLEntityRemover.html. [Πρόσβαση 28 12 2013].

[29] «JUNG (Java Universal Network/Graph Framework),» [Ηλεκτρονικό]. Available: http://jung.sourceforge.net/.

[30] «Semantic Processing, SPARQL-DL API,» derivo Semantic Systems, [Ηλεκτρονικό]. Available: http://www.derivo.de/en/resources/sparql-dl-api.html.

[31] E. Sirin και B. Parsia, «SPARQL-DL: SPARQL Query for OWL-DL,» [Ηλεκτρονικό]. Available: http://webont.org/owled/2007/PapersPDF/submission_23.pdf.

[32] Oracle, «java documentation,» [Ηλεκτρονικό]. Available: http://docs.oracle.com/javase/7/docs/api/java/lang/Class.html. [Πρόσβαση 4 1 2014].

[33] «SwingWorker documentation,» Oracle, [Ηλεκτρονικό]. Available: http://docs.oracle.com/javase/6/docs/api/javax/swing/SwingWorker.html.

[34] «swing worker Oracle tutorial,» Oracle, [Ηλεκτρονικό]. Available: http://docs.oracle.com/javase/tutorial/uiswing/concurrency/worker.html.

[35] «Drag and Drop and Data Transfer,» Oracle, [Ηλεκτρονικό]. Available: http://docs.oracle.com/javase/tutorial/uiswing/dnd/index.html.

[36] «Top-Level Drop,» Oracle, [Ηλεκτρονικό]. Available: http://docs.oracle.com/javase/tutorial/uiswing/dnd/toplevel.html.