2009

Αλεξάνδρειο Τ.Ε.Ι.
Θεσσαλονίκης
Τμήμα Πληροφορικής

Πτυχιακή Εργασία του
Ιωάννη Σωτηριάδη

Επιβλέπων Καθηγητής:
Δημοσθένης Σταμάτης

[ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΚΑΙ ΕΞΟΡΥΞΗ ΔΕΔΟΜΕΝΩΝ ΑΠΟ ΤΟΝ ΠΑΓΚΟΣΜΙΟ ΙΣΤΟ]

Λογικός Προγραμματισμός και Εξόρυξη Δεδομένων από τον Παγκόσμιο Ιστό

Πτυχιακή Εργασία του
Ιωάννη Σωτηριάδη

Επιβλέπων Καθηγητής: Δημοσθένης Σταμάτης

Αλεξάνδρειο Τ.Ε.Ι. Θεσσαλονίκης
Τμήμα Πληροφορικής
Οκτώβριος 09

Πρόλογος

Η εργασία αυτή ασχολείται με τη χρήση του λογικού προγραμματισμού για την εξόρυξη δεδομένων από το τον Παγκόσμιο Ιστό. Η εξόρυξη δεδομένων είναι ένας από τους πιο υποσχόμενους τομείς της πληροφορικής γιατί ο όγκος τον δεδομένων που υπάρχουν είναι μεγάλος και η επεξεργασία τους ιδιαίτερα δύσκολη. Η εργασία ασχολείται με τη δημιουργία ενός λεξικού όρων με τη χρήση ευφυών τρόπων εξόρυξης δεδομένων. Η δημιουργία αυτού του λεξικού γίνεται μέσω της αυτόματης ανάκτησης πληροφοριών από τη Wikipedia και σε μικρό χρονικό διάστημα, κάτι που είναι επίσης σημαντικό, γιατί ο χρόνος σήμερα είναι πολύτιμος. Πιστεύω πως αυτή η εργασία μπορεί να βοηθήσει άτομα τα οποία δεν έχουν χρόνο να ψάχνουν στο διαδίκτυο διάφορους ορισμούς και μέσω του προγράμματος μπορούν μόνο με την αναφορά του όρου να έχουν μια περίληψη και κάποιους σχετικούς όρους. Επίσης αυτή η εργασία μπορεί να τροποποιηθεί και να κάνει ανάκτηση και άλλων πληροφοριών από το διαδίκτυο ανάλογα με το τι ενδιαφέρει τον κάθε χρήστη. Όλη η εργασία έχει γίνει με τη γλώσσα λογικού προγραμματισμού Swi-Prolog και αυτό επειδή η γλώσσα αυτή έχει από μόνη της μια νοημοσύνη. Εκτός αυτού ήταν δική μου επιθυμία να διευρύνω τις προγραμματιστικές μου γνώσεις σε αυτή τη γλώσσα.

Επιθυμώ να ευχαριστήσω τη μητέρα μου και τον θείο μου το Φίλιππο για την υποστήριξη και τη βοήθεια που μου πρόσφεραν. Επιπλέον θέλω να ευχαριστήσω τους καθηγητές που είχα, και ιδιαίτερα τον επιβλέπων καθηγητή μου κ. Σταμάτη, για τις γνώσεις και την καθοδήγηση που μου παρείχαν ώστε να ολοκληρώσω αυτήν την εργασία με επιτυχία.

Περίληψη

Με τον όγκο των πληροφοριών που υπάρχουν σήμερα αποθηκευμένες με ψηφιακό τρόπο, η επεξεργασία τους και η εξαγωγή συμπερασμάτων παίζει έναν πάρα πολύ σημαντικό ρόλο. Αυτή η επεξεργασία μπορεί να γίνει μέσω της εξόρυξης δεδομένων. Η εξόρυξη δεδομένων είναι ο τρόπος με τον οποίο σήμερα οι περισσότεροι οργανισμοί και οι επιχειρήσεις, επεξεργάζονται τα δεδομένα τους έτσι ώστε να παραχθούν σχέσεις ανάμεσά τους τις οποίες εξ' αρχής δε θα μπορούσε κάποιος να καταλάβει. Κατά καιρούς έχουν αναπτυχθεί διάφοροι τρόποι για την εξόρυξη των δεδομένων. Στην παρούσα εργασία γίνεται ειδικότερη αναφορά στους ευφυείς τρόπους ανάκτησης/εξόρυξης των δεδομένων με τη χρήση τεχνικών του Λογικού Προγραμματισμού. Χρησιμοποιήθηκε η γλώσσα προγραμματισμού Swi-Prolog για να γίνει η εκμετάλλευση της “ευφυίας” που παρέχει σαν γλώσσα έτσι ώστε να γίνει πιο “έξυπνη” η ανάκτηση/εξόρυξη των δεδομένων. Σαν μία μελέτη περίπτωσης εξετάζεται η δυνατότητα δημιουργίας ενός λεξικού (γλωσσάρι όρων), από πληροφορίες που είναι αποθηκευμένες στο διαδίκτυο με βάση όρους που καθορίζει ο χρήστης. Ειδικότερα μέσω της Prolog γίνεται σύνδεση στον ιστότοπο της Wikipedia και γίνεται η αναζήτηση των όρων που έχει δώσει ο χρήστης. Αφού βρεθούν οι κατάλληλες σελίδες γίνεται η ανάκτηση κειμένων που αναφέρονται στους σχετικούς όρους και εν συνεχεία η επεξεργασία των κειμένων αυτών και ειδικών δεδομένων που εμπεριέχονται, που στη συγκεκριμένη περίπτωση είναι σύνδεσμοι σε άλλες σχετικές σελίδες.

Abstract

In today's society the amount of digital data is enormous so the processes and the conclusions that we receive are pretty important, especially for companies. This processes can be made by the field of data mining. Data mining is a way of processing large amounts of data, in a short amount of time, to produce “unexpected” relations between data. By the word unexpected is meant that those relations are not seen with first sight. There are different types of data mining but in this text we are concern most with intelligent data mining with the use of Logic Programming. The programming language which is used for data mining, in this subject, is Swi-Prolog. Swi-Prolog is mostly used as an artificial intelligent language due to its cleverness. This cleverness is used here for achieving a higher level of intelligent data mining. The case study that we used in this thesis is the possibility to create a glossary of terms, from information that are stored in Web, based on terms that are defined by the user. Especially through Prolog we establish an internet connection to Wikipedia's website, where the search of elements is done. When those pages are found data extraction is taking place. The extracted data are links and the preview from Wikipedia.

Πίνακας Περιεχομένων

Κατάλογος εικόνων

Εισαγωγή

Η πτυχιακή εργασία που ακολουθεί έχει σαν θέμα **"Λογικός Προγραμματισμός και Εξόρυξη Δεδομένων από τον Παγκόσμιο Ιστό"** και όπως καταμαρτυρεί ο τίτλος, ασχολείται με την εξόρυξη δεδομένων. Με τον όγκο των πληροφοριών που υπάρχουν σήμερα αποθηκευμένες με ψηφιακό τρόπο, η επεξεργασία τους και η εξαγωγή συμπερασμάτων παίζει έναν πάρα πολύ σημαντικό ρόλο. Αυτή η επεξεργασία μπορεί να γίνει μέσω της εξόρυξης δεδομένων. Η εξόρυξη δεδομένων είναι ο τρόπος με τον οποίο σήμερα οι περισσότερες των επιχειρήσεων, καθώς και όλοι οι υπόλοιποι οργανισμοί, επεξεργάζονται τα δεδομένα τους έτσι ώστε να παράγουν σχέσεις ανάμεσά τους τις οποίες εξ' αρχής δε θα μπορούσε κάποιος να καταλάβει. Κατά καιρούς έχουν αναπτυχθεί διάφοροι τρόποι για την εξόρυξη των δεδομένων. Στην παρούσα εργασία θα γίνει αναφορά στους ευφυείς τρόπους ανάκτησης δεδομένων, με τη χρήση τεχνικών του Λογικού Προγραμματισμού που χρησιμοποιούνται για εφαρμογές τεχνητής νοημοσύνης. Στα πλαίσια μιας ευφυούς τεχνικής ανάκτησης/εξόρυξης δεδομένων χρησιμοποιούνται διάφοροι τρόποι όπως είναι:

- ο *περιορισμός του χώρου αναζήτησης των δεδομένων (data dimensionality reduction)*, δηλαδή μέσω μιας πρώτης επεξεργασίας μειώνεται ο χώρος στον οποίο αναζητούνται οι σχέσεις μεταξύ των πληροφοριών για να γίνει πιο γρήγορα η τελική επεξεργασία.

- η *ταξινόμηση και ομαδοποίηση*, κατά την οποία στοιχεία των δεδομένων με ίδια ή παρόμοια χαρακτηριστικά ομαδοποιούνται και αναζητούνται μέσα σε αυτές τις ομάδες οι σχέσεις που ενώνουν τα διάφορα στοιχεία,

- τα *νευρωνικά δίκτυα*, μέσω των οποίων δίνεται σε ένα σύστημα ανάκτησης δεδομένων η ικανότητα να θυμάται καθώς επίσης και να μπορεί να μαθαίνει μόνο του.

Άλλες τεχνικές είναι επίσης η εξαγωγή κανόνων, τα *συστήματα fuzzy*, τα οποία μοιάζουν με τα νευρωνικά δίκτυα αλλά προσομοιάζουν πιο πολύ τον τρόπο

σκέψης και αντίληψης του ανθρώπου, τα *Radial Base Functions* και οι *γενετικοί αλγόριθμοι*, που η λειτουργία τους βασίζεται στο τρόπο λειτουργίας των οργανισμών κατά την εξέλιξη τους και στην προσπάθεια προσομοίωσης αυτής της λειτουργίας μέσω υπολογιστικών διαδικασιών και τέλος τα *Support Vector Machines* που χρησιμοποιούν για τη λειτουργία τους διανύσματα υποστήριξης από τα οποία πήραν και το όνομα τους.

Για την ανάπτυξη του συστήματος χρησιμοποιήθηκε η γλώσσα προγραμματισμού Swi-Prolog, για να γίνει η εκμετάλλευση της “ευφυίας” που παρέχει σαν γλώσσα έτσι ώστε να γίνει πιο “έξυπνη” η ανάκτηση/εξόρυξη των δεδομένων. Σαν μία μελέτη περίπτωσης εξετάζεται η δυνατότητα δημιουργίας ενός λεξικού (γλωσσάρι όρων), από πληροφορίες που είναι αποθηκευμένες στο διαδίκτυο με βάση όρους που καθορίζει ο χρήστης. Ειδικότερα, μέσω της Prolog γίνεται σύνδεση στον ιστότοπο της Wikipedia και γίνεται η αναζήτηση των όρων που έχει δώσει ο χρήστης. Αφού βρεθούν οι κατάλληλες σελίδες γίνεται η ανάκτηση κειμένου που αναφέρονται στους σχετικούς όρους και εν συνεχεία η επεξεργασία των κειμένων αυτών και ειδικών δεδομένων που εμπεριέχονται, που στη συγκεκριμένη περίπτωση είναι σύνδεσμοι σε άλλες σχετικές σελίδες. Θεωρείται σημαντικό να αναφερθεί πως πολύ ώρα δαπανήθηκε στη παρατήρηση και εξέταση του ιστότοπου της Wikipedia για να γίνει αντιληπτός ο τρόπος με τον οποίο έχει γίνει η μορφοποίηση του και να βρεθούν όσο το δυνατόν περισσότερες περιπτώσεις. Σκοπός μας ήταν να καλυφθεί το μεγαλύτερο εύρος των περιπτώσεων και να βρεθούν οι διάφορες ιδιαιτερότητες που μπορεί να προκαλέσουν σφάλματα.

Το σύστημα χρησιμοποιεί 14 modules. Οι λόγοι για τη χρήση αυτών των modules είναι οι εξής:

- Για να γίνουν όλες αυτές οι λειτουργίες,
- για την καλύτερη δόμηση του συστήματος,
- για την ευκολότερη κατανόηση,
- τη δυνατότητα επαναχρησιμοποίησης ή τροποποίησης του και

- για να μπορεί ο χρήστης να το προσαρμόσει στις απαιτήσεις του κατασκευάστηκαν δεκατέσσερα modules.

Τα modules είναι ο τρόπος της Prolog να “κρύβει” τα κατηγορήματα της και να φαίνονται μόνο αυτά που θέλει ο προγραμματιστής. Τα περισσότερα από αυτά τα modules κατασκευάζονται για να επιτελέσουν συγκεκριμένες λειτουργίες. Σε αυτήν την εργασία κατασκευάστηκαν modules:

- για γίνεται η *ανάγνωση των όρων* από το πληκτρολόγιο,
- για να *επεξεργάζονται οι όροι* που δέχεται το πρόγραμμα από το πληκτρολόγιο έτσι ώστε να είναι δυνατόν να δοθούν, οι όροι, με μια μορφοποίηση και να μπορεί να γίνει αντιληπτή από το πρόγραμμα αυτή η μορφοποίηση,
- για τις *λειτουργίες ανάκτησης*,
- για τη *μορφοποίηση* των ανακτηθέντων κειμένων,
- για τις *λειτουργίες εγγραφής,*
- για να γίνονται κάποιες πρωταρχικές λειτουργίες, οι οποίες θα μπορούσε να επιτελεστούν από το κυρίως πρόγραμμα, αλλά για καλύτερο καταμερισμό των εργασιών προτιμήθηκε η δημιουργία και ενός καινούργιο module.

Τέλος μέσα στο σύστημα χρησιμοποιήθηκαν μερικά modules από τη βιβλιοθήκη της Swi-Prolog, στα οποία έγιναν μερικές τροποποιήσεις σε κάποια σημεία ώστε να γίνει πιο λειτουργικό το σύστημα.

Η δομή που ακολουθεί το κείμενο της πτυχιακής εργασίας είναι η ακόλουθη:

Στο **πρώτο κεφάλαιο** γίνεται μια ανάλυση του όρου εξόρυξη δεδομένων (data mining), της εξόρυξης δεδομένων με ευφυΐα και διαφόρων τεχνικών της.

Στο **δεύτερο κεφάλαιο** γίνεται αναφορά για το λογικό προγραμματισμό, καταγράφονται τα προτερήματα που παρουσιάζει και πως χρησιμοποιήθηκε στην εργασία όπως επίσης και πως είναι δομημένη και πως λειτουργεί η Swi-Prolog.

Στα επόμενα κεφάλαια γίνεται μια τεκμηρίωση του τι κάνει η εργασία και πως το υλοποιεί. Πιο αναλυτικά:

Στο **τρίτο κεφάλαιο** επεξηγείται η σύνδεση στο Παγκόσμιο Ιστό και οι τρόποι με τους οποίους γίνεται η αναζήτηση και η εύρεση των σελίδων τον οποίων θέλουμε να χρησιμοποιήσουμε και τέλος η ανάκτηση του κατάλληλου κειμένου.

Στο **τέταρτο κεφάλαιο** γίνεται η επεξήγηση μιας επιπλέον ανάκτησης που γίνεται για να παρουσιάζεται πιο πλήρης η περιγραφή των όρων.

Στο **πέμπτο κεφάλαιο** γίνεται η επεξήγηση της επεξεργασίας των κειμένων και των δεδομένων που έχουν ανακτηθεί όπως επίσης και πως γίνεται η διαχείριση των εγγραφών σε αρχεία.

Τέλος στο **τελευταίο κεφάλαιο** δίνεται ένας επίλογος που περιέχει μια ανασκόπηση, τα συμπεράσματα που βγήκαν καθώς και που μπορεί να προσφέρει η συγκεκριμένη εργασία και που μπορεί να χρησιμοποιηθεί.

Κεφάλαιο 1

Εξόρυξη Δεδομένων

Εξόρυξη Δεδομένων έχει περιγραφεί *ως "η ασυνήθιστη εξαγωγή υπονοούμενης, ως τότε άγνωστης, και πιθανώς χρήσιμης πληροφορίας από τα δεδομένα"* (W. Frawley et al.,1992) και ως *"η επιστήμη εξαγωγής χρήσιμων πληροφοριών από σημαντικές ποσότητες δεδομένων ή από βάσεις δεδομένων"* (D. Hand et al.,2001). Ο λόγος που χρησιμοποιούμε την εξόρυξη δεδομένων σε τέτοια σύνολα -από δεδομένα- είναι η παραγωγή κανόνων ώστε να οδηγηθούμε σε αποφάσεις σχετικά με μελλοντικές ενέργειες. Χρησιμοποιείται επίσης, από εταιρίες σε συνδυασμό με το σύστημα ERP των εταιριών, ως σύμβουλος για διοικητικές αποφάσεις βασιζόμενη σε πρότυπα και προβλέψεις που μπορεί να γίνουν από τα δεδομένα που υπάρχουν.

Η εξόρυξη δεδομένων είναι όμοια με τον κλάδο της *διερευνητικής ανάλυσης* της στατιστικής, η οποία έχει τους ίδιους στόχους και βασίζεται σε στατιστικά μέτρα. Ένας άλλος τομέας με τον οποίο είναι συνδεδεμένη, είναι αυτός της Τεχνητής Νοημοσύνης. Είναι πολύ σημαντικό τμήμα της **ανακάλυψης γνώσης** και της **μηχανικής μάθησης**. (Ramakrishnan R., Gehrke J., 2002)

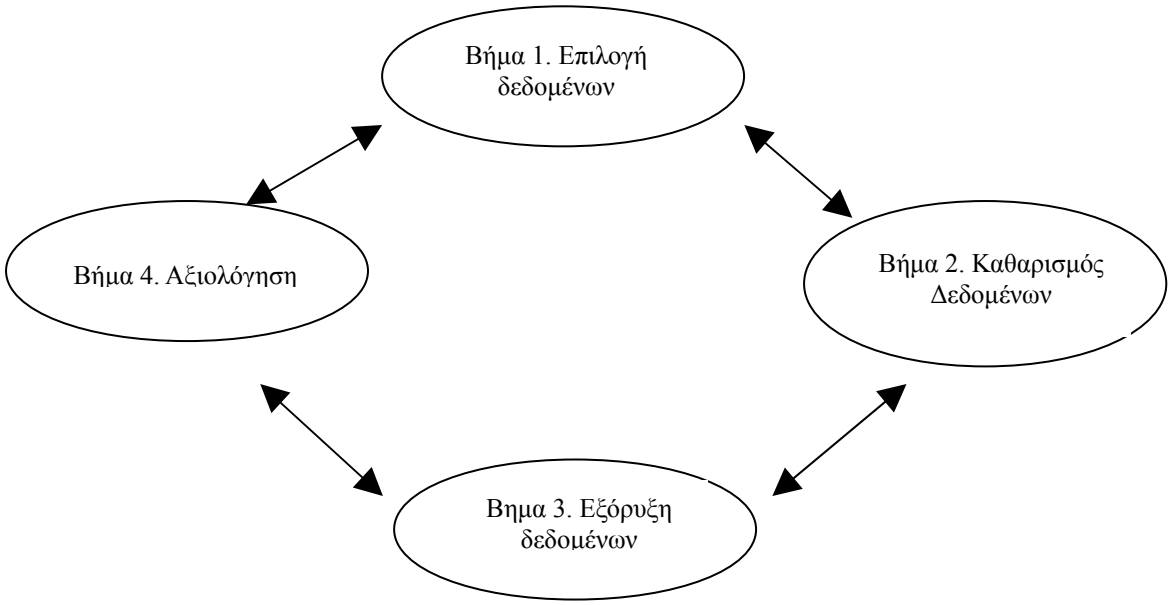
Αν και η εξόρυξη δεδομένων είναι κάτι σχετικά καινούργιο στον τομέα της Πληροφορικής, η τεχνολογία για εξόρυξη δεδομένων έχει αναπτυχθεί εδώ και χρόνια λόγω των αναγκών των διαφόρων επιχειρήσεων. Οι επιχειρήσεις αυτές χρησιμοποιούσαν ισχυρούς υπολογιστές για να καταφέρουν να προσπελάσουν τους μεγάλους όγκους των δεδομένων (Wikipedia Data mining, 2009). Στη σημερινή κοινωνία βρίσκει αρκετές εφαρμογές σε διάφορους τομείς όπως είναι τα παιχνίδια, η καταπολέμηση της τρομοκρατίας, οι επιχειρήσεις, η επιστήμη κτλ.

1.1 Ανακάλυψη γνώσης σε βάσεις δεδομένων(KDD)

Η *ανακάλυψη γνώσης σε βάσεις δεδομένων* είναι ένας τομέας της επιστήμης των υπολογιστών που βρίσκει σημαντική εφαρμογή στην **εξόρυξη δεδομένων** όπως και στη **μηχανική μάθηση**. Κατά την ανακάλυψη γνώσης γίνονται σύνθετες διαδικασίες μέσω των οποίων μπορούμε να ανακαλύψουμε χρήσιμα και κατανοητά **πρότυπα** τα οποία προέρχονται από τα δεδομένα των βάσεων. Αυτές είναι ολοκληρωμένες διαδικασίες που περιλαμβάνουν την *επεξεργασία των δεδομένων*, την *εφαρμογή των αλγορίθμων ανακάλυψης γνώσης* και την *ερμηνεία των αποτελεσμάτων*. Τέλος η ανακάλυψη γνώσης σε βάσεις δεδομένων απαιτεί και τη συνδρομή ενός **ειδικού**, του τομέα όπου θα εφαρμοστεί η διαδικασία, ο οποίος θα πρέπει να παίρνει συγκεκριμένες αποφάσεις σε κάποια επιμέρους στάδια της διαδικασίας (Βλαχάβας Ι. κ.α., 2006).

Τα επιμέρους στάδια τις διαδικασίας της ανακάλυψης γνώσης, σύμφωνα με τους **Ramahkrishnan** και **Gehrke**, είναι τέσσερα. Το πρώτο στάδιο είναι η *επιλογή δεδομένων***,** στο οποίο έρχονται τα ακατέργαστα δεδομένα για να γίνει ο προσδιορισμός των συνόλων δεδομένων και των γνωρισμάτων εκείνων που μπορούν να προκαλέσουν ενδιαφέρον. Το δεύτερο στάδιο είναι ο *καθαρισμός των δεδομένων*. Ο θόρυβος και η απομάκρυνσή του είναι ένα από τα πιο σημαντικά βήματα, γιατί εάν δεν αφαιρεθεί τότε υπάρχει η πιθανότητα να χάσουμε ενδιαφέροντα πρότυπα σχήματα, ενώ η αξιοπιστία των εντοπισμένων προτύπων θα είναι χαμηλή. Ο καθαρισμός των δεδομένων, έχει σαν στόχο την απομάκρυνση του θορύβου και των τιμών προς εξαίρεσης. Εν συνεχεία γίνεται ο μετασχηματισμός των τιμών, που περιέχουν τα πεδία, σε κοινές μονάδες μέτρησης, δημιουργούνται έτσι νέα πεδία, συνδυάζοντας τα ήδη υπάρχοντα, και γίνεται η τοποθέτηση των δεδομένων στο σχεσιακό σχήμα, το οποίο θα χρησιμοποιηθεί στην είσοδο της επεξεργασίας της εξόρυξης δεδομένων. Τρίτο στάδιο είναι η *εξόρυξη δεδομένων* στο οποίο γίνεται η εξαγωγή των πραγματικών πρότυπων σχημάτων. Τελικό στάδιο είναι η *αξιολόγηση*. Κατά την αξιολόγηση γίνεται η παρουσίαση των αποτελεσμάτων σε κατανοητή, για το χρήστη, μορφή.

Υπάρχει η δυνατότητα από κάθε βήμα να γυρίσουμε πίσω σε οποιοδήποτε στάδιο ώστε να είναι δυνατόν να γίνει η επανάληψη της όλης διαδικασίας με την επιπρόσθετη γνώση που έχουμε αποκτήσει. Τα τέσσερα στάδια αυτά φαίνονται παρακάτω στην Εικόνα 1.



Εικόνα 1. Στάδια ανακάλυψης γνώσης

1.2 Ευφυής Εξόρυξη Δεδομένων

Επειδή όπως αναφέρθηκε ο όγκος των δεδομένων σήμερα είναι υπερβολικά μεγάλος αναζητούνται τρόποι για να γίνεται πιο εύκολα η επεξεργασία τους. Έχουν αναπτυχθεί πλέον τεχνικές οι οποίες χρησιμοποιούν “ευφυΐα” για να μπορεί να γίνει αυτή η επεξεργασία και να παραχθούν τα σωστά συμπεράσματα. Αυτές οι τεχνικές χρησιμοποιούν κυρίως **νευρωνικά δίκτυα** -τα οποία έχουν την ικανότητα εκμάθησης- όπως τα *δίκτυα fuzzy*, το *multi layer perceptron*, το *Radial Base Function*. Επιπλέον χρησιμοποιούν *γενετικούς αλγορίθμους* και *support vector machines*. Θα γίνει αναφορά σε τρεις κύριες τεχνικές εξόρυξης δεδομένων τον

Περιορισμό του χώρου αναζήτησης των δεδομένων (Data Dimensionality Reduction), την **ταξινόμηση και ομαδοποίηση** και την **εξαγωγή κανόνων**. (Wang L., Fu X., 2005)

1.3 Περιορισμός του χώρου αναζήτησης των δεδομένων (Data Dimensionallity Reduction)

Ο *περιορισμός του χώρου αναζήτησης των δεδομένων (DDR)* μπορεί:

- να μειώσει τη διάσταση του διαστήματος αναζήτησης της υπόθεσης,
- να μειώσει τις δαπάνες συλλογής δεδομένων και αποθήκευσης,
- να ενισχύσει την απόδοση εξόρυξης δεδομένων,
- και να απλοποιήσει τα αποτελέσματα εξόρυξης δεδομένων.

Μια κατηγορία του DDR είναι η **εξαγωγή χαρακτηριστικών.** Οι ιδιότητες ή τα χαρακτηριστικά γνωρίσματα είναι μεταβλητές των στοιχείων που πάρθηκαν σαν δείγμα. Κατά την εξαγωγή αυτή, τα νέα χαρακτηριστικά προέρχονται από τα αρχικά χαρακτηριστικά προκειμένου να αυξηθεί η υπολογιστική αποτελεσματικότητα και η ακρίβεια της ταξινόμησης. Οι τεχνικές εξαγωγής χαρακτηριστικών γνωρισμάτων περιλαμβάνουν συχνά μη γραμμικό μετασχηματισμό. Η *Γραμμική διακρίνουσα ανάλυση* και η *ανάλυση κύριων τμημάτων* είναι δύο δημοφιλείς τεχνικές για την εξαγωγή χαρακτηριστικών.

Οι μη γραμμικές μέθοδοι μετασχηματισμού είναι καλές στην προσέγγιση και την εξέταση πρακτικού μη γραμμικού προβλήματος. Αυτές οι μέθοδοι μετασχηματισμού έχουν και προβλήματα. Μπορεί να να βγάλουν απροσδόκητα και ανεπιθύμητα αποτελέσματα για τα δεδομένα. Επιπλέον δεν είναι συχνά ικανές να αναστρέψουν τα αποτελέσματα, και η γνώση που μαθαίνεται, με την εφαρμογή μιας μη γραμμικής μεθόδου μετασχηματισμού σε ένα διάστημα χαρακτηριστικών γνωρισμάτων, δεν μπορεί να μεταβιβαστεί στο επόμενο διάστημα

χαρακτηριστικών γνωρισμάτων.

Η άλλη κατηγορία της DDR είναι **επιλογή χαρακτηριστικών**. Λαμβάνοντας υπόψη ένα σύνολο αρχικών χαρακτηριστικών, οι τεχνικές επιλογής επιλέγουν ένα υποσύνολο χαρακτηριστικών που εκτελεί το καλύτερο για τα συστήματα επαγωγής, όπως ένα σύστημα ομαδοποίησης. Η έρευνα για το βέλτιστο υποσύνολο των χαρακτηριστικών γνωρισμάτων είναι συνήθως δύσκολη. Έχει πολλά προβλήματα επιλογής χαρακτηριστικών που έχουν αποδειχθεί ότι να είναι *NP-hard*. Η τεχνολογία επιλογής χαρακτηριστικών, εξερευνάται ευρέος λόγω της ευκολίας της μετάφρασης των χαρακτηριστικών που επιλέγονται από το αρχικό χαρακτηριστικό. (Wang L., Fu X., 2005)

1.4 Ταξινόμηση και Ομαδοποίηση

Η *ταξινόμηση και η ομαδοποίηση* είναι δύο τεχνικές εξόρυξης δεδομένων με στενές σχέσεις. Μια **κατηγορία** είναι ένα σύνολο δειγμάτων στοιχείων με κάποια ομοιότητα ή σχέση. Σε όλα τα δείγματα μιας κατηγορίας ορίζεται η ίδια ετικέτα για να τις διακρίνουν από τα δείγματα άλλων κατηγοριών. Μια **συστάδα** είναι μια συλλογή αντικειμένων που είναι παρόμοια σε συγκεκριμένα στοιχεία. Οι συστάδες παράγονται συνήθως προκειμένου να ταξινομηθούν περαιτέρω τα αντικείμενα, στις σχετικά μεγαλύτερες και με καλύτερο νόημα κατηγορίες.

Λαμβάνοντας υπόψη ένα σύνολο στοιχείων με τις ετικέτες κατηγορίας, οι αναλυτές στοιχείων χτίζουν τις ομάδες για τα μελλοντικά άγνωστα αντικείμενα. Ένα πρότυπο ταξινόμησης είναι διαμορφωμένο ώστε να βασίζεται στα διαθέσιμα στοιχεία. Οι μελλοντικές τάσεις προβλέπονται χρησιμοποιώντας ένα **μαθησιακό πρότυπο**.

Γενικά, η συγκέντρωση μπορεί να υιοθετηθεί για την εξέταση των στοιχείων χωρίς ετικέτες κατηγορίας. Μερικές μέθοδοι ταξινόμησης μαζεύουν τα στοιχεία σε μικρές ομάδες πριν προχωρήσουν στη ταξινόμηση, π.χ. στο νευρωνικό δίκτυο RBF. (Wang L., Fu X., 2005)

1.5 Εξαγωγή Κανόνων

Η *εξαγωγή κανόνων* γίνεται με σκοπό να παρουσιαστούν τα δεδομένα κατά τέτοιο τρόπο ώστε οι ερμηνείες να είναι κατανοητές και οι αποφάσεις να μπορούν να ληφθούν βασισμένες στη γνώση που προέρχεται από τα στοιχεία. Ο λόγος που γίνεται αυτή η εξαγωγή είναι γιατί οι “πελάτες” της εξόρυξης δεδομένων αναμένουν μια απλή εξήγηση γιατί υπάρχουν ορισμένα αποτελέσματα ομαδοποίησης, τι συμβαίνει σε μια βάση δεδομένων με πολλά χαρακτηριστικά, ποιο χαρακτηριστικό επηρεάζει τα αποτελέσματα της εξόρυξης δεδομένων κ.λπ.

Η εξαγωγή κανόνων μπορεί να ταξινομηθεί σε δύο σημαντικούς τύπους. Ο πρώτος ενδιαφέρεται για τη **σχέση μεταξύ των ιδιοτήτων εισαγωγής και των ετικετών**. Ο δεύτερος είναι η **εξόρυξη κανόνων σχέσεων** που εξάγει σχέσεις μεταξύ των ιδιοτήτων, στα σύνολα στοιχείων που μπορούν να μην έχουν τις ετικέτες κατηγορίας. Οι τεχνικές εξαγωγής κανόνα-ένωσης χρησιμοποιούνται συνήθως για να ανακαλύψουν σχέσεις μεταξύ των στοιχείων κάποιας συναλλαγής.

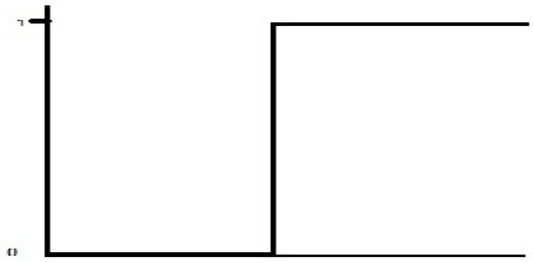
1.6 Νευρωνικά δίκτυα

Τα *νευρωνικά δίκτυα* όπως αναφέρθηκε χρησιμοποιούνται ευρέως στην εξόρυξη δεδομένων και κυρίως στη δημιουργία ταξινομητών. Η λειτουργία τους προσομοιάζει τις νευρικές διασυνδέσεις στον εγκέφαλο, μέσω ψηφιακών συστημάτων που θεωρούνται νευρικά πρότυπα δικτύων. Οι νέες εφαρμογές και οι νέες αρχιτεκτονικές των νευρικών δικτύων (NNs) χρησιμοποιούνται και ερευνώνται περαιτέρω στις επιχειρήσεις και τα ερευνητικά ιδρύματα για τον έλεγχο των δαπανών και τη παραγωγή του εισοδήματος. Η αναβίωση ενδιαφέροντος για τα νευρωνικά δίκτυα έχει τροφοδοτηθεί από την επιτυχία στη θεωρία και τις εφαρμογές.

1.6.1 Multilayer Perceptron

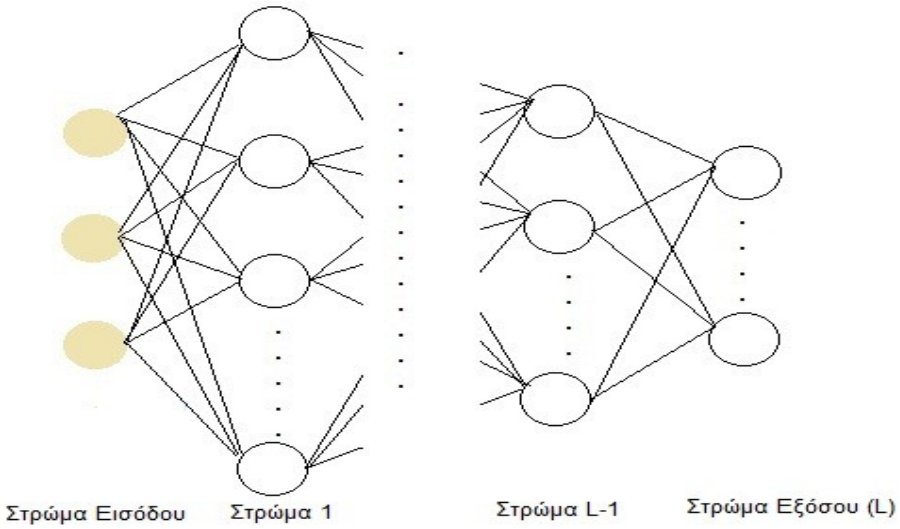
Θεωρούμαι ένα δίκτυο τριών νευρώνων οι οποίοι χρησιμοποιούν τη **βηματική συνάρτηση**(Εικόνα 32:

$$f(u) = \begin{cases} 0 \text{ αν } u < 0, \\ 1 \text{ αν } u \geq 0. \end{cases}$$



Εικόνα 2. Βηματική συνάρτηση

Ένα τέτοιο δίκτυο αποτελείται από δύο στρώματα: το κρυφό στρώμα και το στρώμα εξόδου (Εικόνα 3). Στην ουσία οι κρυφοί νευρώνες υλοποιούν 2 ευθείες έτσι ώστε να κατηγοριοποιήσουν τα στοιχεία. Εάν χρησιμοποιήσουμε παραπάνω νευρώνες μπορούμε να υλοποιήσουμε περισσότερες ευθείες. Υπάρχουν πολλοί αλγόριθμοι εκπαίδευσης του *MLP* ένας από τους οποίους είναι και ο **back-propagation**. Το χαρακτηριστικό των δικτύων αυτών είναι ότι *οι νευρώνες ενός οποιουδήποτε στρώματος τροφοδοτούν αποκλειστικά τους νευρώνες του επόμενου στρώματος και τροφοδοτούνται αποκλειστικά από τους νευρώνες του προηγούμενου στρώματος* (Διαμαντάρας Κ., 2007).



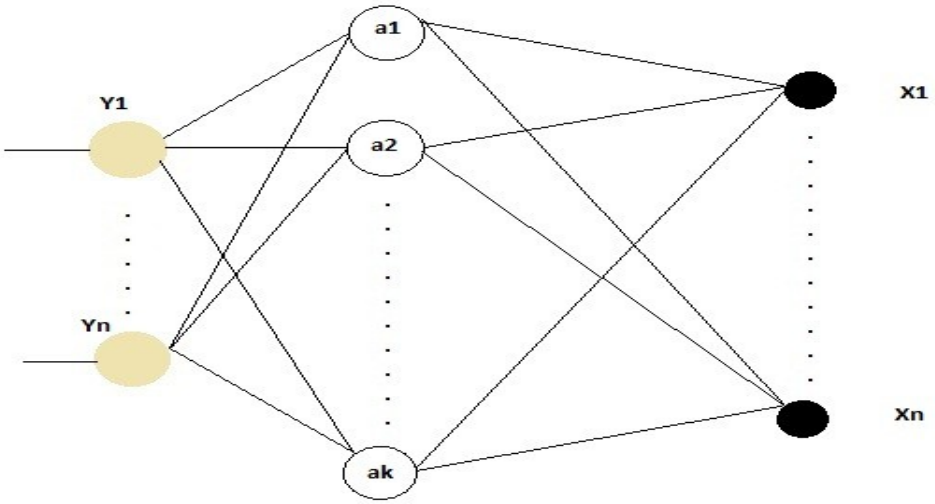
Εικόνα 3. Δίκτυο Multilayer Perceptron

1.6.2 Δίκτυα Fuzzy

Τα *δίκτυα Fuzzy* συνδυάζουν την εκμάθηση και υπολογιστική δύναμη των crisp νευρικών δικτύων με τις περιγραφές που πλησιάζουν την ανθρώπινη λογική και το συλλογισμό των fuzzy συστημάτων. Δεδομένου ότι η fuzzy λογική έχει σχέση με την ανθρώπινη αντιπροσώπευση γνώσης, έπρεπε να γίνει ένα βασικό συστατικό των συστημάτων εξόρυξης δεδομένων. Ένα σαφές πλεονέκτημα της fuzzy λογικής είναι ότι *μπορούμε να εκφράσουμε τη γνώση για μια βάση δεδομένων με έναν τρόπο που είναι φυσικός για τους ανθρώπους να κατανοήσουν*. Πρόσφατα, η ερευνητική προσοχή αφιερώνεται στην παραγωγή κανόνων που χρησιμοποιούν διάφορα δίκτυα Fuzzy.

1.6.3 Radial Base Function (RBF)

Οι *συναρτήσεις ακτινικού τύπου*(RBF) είναι οι συναρτήσεις στις οποίες υπάρχει κάποιο διάνυσμα y, που είναι το κέντρο της και η τιμή της συνάρτησης εξαρτάται μόνο από την απόσταση χ από το κέντρο *f(x) = f(||x-y||).* Το εύρος της συνάρτησης είναι επίσης μια παράμετρος που ρυθμίζεται.



Εικόνα 4. Δίκτυο RBF

Τα δίκτυα RBF (Εικόνα 4) αποτελούνται από δύο στρώματα, αυτό γίνεται επειδή δεν υπάρχουν ικανοποιητικοί αλγόριθμοι εκπαίδευσης περισσοτέρων στρωμάτων

και επειδή τα αποτελέσματα που παράγονται με δύο στρώματα είναι αρκετά ικανοποιητικά. Η μόνη ουσιαστική διαφορά των RBF με τα δίκτυα MLP είναι ότι χρησιμοποιούν **συνεχείς συναρτήσεις** αντί της βηματικής που χρησιμοποιείται στα MLP (Διαμαντάρας Κ., 2007). Τέλος τέτοιου είδους δίκτυα χρησιμοποιούνται για *προσέγγιση συναρτήσεων*, την *εκτίμηση πυκνότητας*, την *ομαδοποίηση* κ.λπ. (Wang L., Fu X., 2005)

1.7 Γενετική Αλγόριθμοι

Οι *γενετικοί αλγόριθμοι* παρακινούνται από τη φυσική εξελικτική διαδικασία. Στους γενετικούς αλγόριθμους οι λύσεις, που μπορεί να έχει ένα πρόβλημα, κωδικοποιούνται στα χρωμοσώματα ή τα άτομα. Ένας **αρχικός πληθυσμός** αυτών των χρωμοσωμάτων παράγεται *τυχαία* ή *ευριστικά*. Οι τελεστές συνδυασμού περιλαμβάνουν την *επιλογή*, τη *διασταύρωση*, και τη *μετάλλαξη*. Για να γίνει η παραγωγή μιας νέας γενεάς, χρωμοσώματα επιλέγονται σύμφωνα με τα αποτελέσματα τους, δηλαδή ένα ποιοτικό κριτήριο που χρησιμοποιείται για την αξιολόγηση των λύσεων ενός προβλήματος. Ο τελεστής επιλογής δίνει προτεραιότητα στα **καλύτερα** άτομα ως γονείς για την επόμενη γενεά. Οι γενετικοί αλγόριθμοι είναι χρήσιμοι για τις ακόλουθες περιστάσεις:

- το διάστημα προβλήματος είναι μεγάλο, σύνθετο
- η προγενέστερη γνώση είναι λιγοστή
- είναι δύσκολο να προσδιοριστεί ένα μοντέλο μηχανικής εκμάθησης να λύσει το πρόβλημα λόγω των περίπλοκων στους περιορισμούς και τους στόχους
- οι παραδοσιακές μέθοδοι αναζήτησης αποδίδουν άσχημα. (Wang L., Fu X., 2005)

1.8 Support Vector Machines

Βασική ιδέα του αλγορίθμου *SVM* είναι να *βρίσκει το βέλτιστο υπέρ-επίπεδο που*

μπορεί να μεγιστοποιήσει το περιθώριο μεταξύ δύο ομάδων δειγμάτων. Τα διανύσματα που είναι τα κοντινότερα στο βέλτιστο υπέρ-επίπεδο καλούνται **διανύσματα υποστήριξης** (support vectors) και αυτός ο αλγόριθμος καλείται *Support Vector Machine*. Έναντι άλλων αλγορίθμων, ο SVM έχει παρουσιάσει εκπληκτικές δυνατότητες στην *αντιμετώπιση προβλημάτων ταξινόμησης*. Έχει εφαρμοστεί ευρέως στα προβλήματα ταξινόμησης και μη γραμμικής οπισθοδρόμησης. Το SVMs υιοθετείται συνήθως στα προβλήματα ταξινόμησης. Εφόσον το SVM εκπαιδευτεί, *μπορεί να χρησιμοποιηθεί και για να προβλέψει τις μελλοντικές τάσεις*. (Wang L., Fu X., 2005).

1.9 Εργαλεία που χρησιμοποιούνται

Όπως αναφέρθηκε παραπάνω οι τεχνικές εξόρυξης δεδομένων είναι πολύ σημαντικές για τις επιχειρήσεις. Τα 8 πιο σημαντικά εργαλεία που χρησιμοποιούνται (σε διάφορους τομείς), σύμφωνα με μια μελέτη έγινε από το **Gartner το 2008**, είναι:

- Angoss Software
- Infor CRM Epiphany
- Portrait Software
- SAS
- G-Stat
- SPSS
- ThinkAnalytics
- Unica

Angoss Software

Η εταιρία Angoss προσφέρει μία σειρά εφαρμογών για εξόρυξη δεδομένων και υποστήριξης λήψης αποφάσεων. Οι εφαρμογές αυτές είναι:

- KnowledgeSEEKER
- KnowledgeSTUDIO
- StrategyBUILDER

KnowledgeSEEKER

Είναι ένα λογισμικό το οποίο δίνει τη δυνατότητα για **data profiling**, **προηγμένη απεικόνιση δεδομένων** και **δέντρα αποφάσεων**. Χαρακτηρίζεται από τη δυνατότητα για "προηγμένη εισαγωγή" δεδομένων, τη δειγματοληψία και την προετοιμασία στοιχείων. Μπορεί και εισάγει δεδομένα σχεδόν από κάθε πηγή. Έχει μια εκτεταμένη σειρά εργαλείων για εξερεύνηση δεδομένων και οριστικοποίηση. Τα γραφικά προσφέρουν rapid profiling και μπορεί να γίνει εξαγωγή τους σε εφαρμογές του Microsoft Office. Τέλος το KnowledgeSEEKER προσφέρει και δέντρα αποφάσεων.

KnowledgeSTUDIO

Είναι ένα λογισμικό το οποίο χρησιμοποιείται σε όλες τις φάσεις της εξέλιξης ενός μοντέλου και του κύκλου ανάπτυξης σε ένα οπτικό περιβάλλον. Προσφέρει *δέντρα αποφάσεων*, *γραμμική* και *λογική οπισθοδρόμηση*, *νευρωνικά δίκτυα*, *συσταδοποίηση*. Παρέχει ένα ενσωματωμένο σύνολο εργαλείων σημείωσης και επέκτασης. Τέλος το KnowledgeSTUDIO επιτρέπει στον αναλυτή να παραγάγει γρήγορα τον πρότυπο κώδικα που μπορεί να εξαχθεί σε ουσιαστικά οποιαδήποτε εφαρμογή, βάση δεδομένων ή μηχανή επιχειρηματικών κανόνων ή στο σύστημα διαχείρισης απόφασης με τις αυτοματοποιημένες γεννήτριες Java, SQL, XML, PMML και SAS.

StrategyBUILDER

Είναι ένα υποσύνολο του KnowledgeSEEKER και του KnowledgeSTUDIO και δίνει τις δυνατότητες σχεδιασμού, επικύρωσης και ανάπτυξης των αναγκαίων

προβλεπόμενων και επιχειρησιακών κανόνων. Προσφέρει ένα διαισθητικό εργαλείο αποφάσεων που βοηθά στη δημιουργία και στον έλεγχο των βασικών δεικτών απόδοσης. (Angoss, 2009)

Infor CRM Epiphany

Το Infor CRM Epiphany βοηθά τις επιχειρήσεις σε θέματα σχετικά με τους πελάτες τους (π.χ. μπορεί να προβλέψει τις αγοραστικές τάσεις των πελατών) έτσι ώστε βελτιώσουν τις πωλήσεις, το μάρκετινγκ, και τις διαδικασίες υπηρεσιών. Το Infor CRM Epiphany είναι ένα προϊόν που έρχεται από την νοοτροπία της Infor *"enrich, extend, and evolve"* και της Infor Open SOA. Τα ισχυρά σημεία του Infor CRM Epiphany είναι οι αιτήσεις μάρκετινγκ , οι οποίες μπορούν να χρησιμοποιηθούν και από μη ειδικούς που θέλουν να προωθήσουν τις εκστρατείες, έχουν ένα συγκεκριμένο στόχο, γρήγορα και εύκολα. Το Infor CRM Epiphany προσφέρει ικανότητες σε άλλους τομείς όπως είναι το CRM, συμπεριλαμβανομένων των πωλήσεων, της εξυπηρέτησης πελατών, αλλά παρέχει την ελάχιστη υποστήριξη για το ηλεκτρονικό εμπόριο, και τη διαχείριση στοιχείων πελατών. (Infor, 2009)

Portrait Software

Η εταιρία Portrait Software προσφέρει μία γκάμα από λογισμικά τα οποία βοηθούν τι εταιρίες να προβλέψουν και να αναλύσουν τις συμπεριφορές των πελατών τους. Παρέχουν τη δυνατότητα οπτικοποίησης των αποτελεσμάτων και γενικότερα δυνατότητες διευκόλυνσης των χρηστών αυτών των προγραμμάτων έτσι ώστε να βελτιστοποιήσει την απόδοσή τους. Αυτά τα προγράμματα είναι: το *Portrait Campaign Manager*, το *Portrait Interaction Optimizer*, *το Portrait Self Service Analytics*, το *Portrait Quadstone Analytics*, το *Portrait Uplift Optimizer* και το *Portrait Foundation*.

SAS

Τα προγράμματα SAS για να λειτουργήσουν χρειάζονται προγράμματα τα οποία είναι και αυτά γραμμένα σε SAS, είτε γράφοντας ο προγραμματιστής ο ίδιος το πρόγραμμα που θέλει για να το “τρέξει” στο λογισμικό, είτε με τη δυνατότητα που δίνεται μέσω του γραφικού περιβάλλοντος, το οποίο και αυτό τρέχει τα αντίστοιχα προγράμματα σε SAS. Είναι ένα πρόγραμμα το οποίο μπορεί να εγκατασταθεί στα πιο διαδεδομένα λειτουργικά συστήματα (Windows, Mac Os X, Unix, Linux), όπως επίσης μπορεί να τρέξει και σε IBM mainframe.

Ένα πρόγραμμα SAS είναι χωρισμένο σε 3 σημαντικά κομμάτια τα οποία είναι: το **βήμα δεδομένων**, το **βήμα των διαδικασιών** και η **γλώσσα macro**.

Το *βήμα των δεδομένων*, το οποίο είναι και το πιο ενδιαφέρον για όσους ασχολούνται με την εξόρυξη δεδομένων, τροποποιεί τα δεδομένα με ένα συγκεκριμένο τρόπο και κάνει τα αρχεία έτσι, ώστε να μπορεί το λειτουργικό σύστημα να τα αναγνωρίσει. Επίσης χρησιμοποιεί και την SQL και λόγω αυτού μπορεί ο χρήστης να κάνει ερωτήματα στη βάση για τα δεδομένα μέσω SQL queries.

Τέλος το SAS παρέχει στους χρήστες του τη δυνατότητα για:

- Εισαγωγή, διαχείριση, ανάκτηση και εξόρυξη δεδομένων.
- Στατιστική ανάλυση
- Επιχειρηματικό σχεδιασμό, υποστήριξη αποφάσεων
- Διαχείριση Εφαρμογών
- data warehousing .

G-Stat

Η G-Stat είναι μια εταιρία η οποία ειδικεύεται στην εξόρυξη δεδομένων. Έχει υλοποιήσει το λογισμικό G-STAT Analytical Platform το οποίο βασίζεται στον SQL Server 2005 της Microsoft και στη τεχνολογία *.NET.* Είναι ένα πακέτο το οποίο παρέχει λύσεις σε πραγματικό χρόνο και δίνει απαντήσεις σε σημαντικής σημασίας ερωτήσεις των επιχειρήσεων. Είναι ενσωματωμένο στο Data Warehouse της επιχείρησης, υπάρχουν και άλλες σχετικές βάσεις δεδομένων και εφαρμογές CRM, που μετασχηματίζουν τη μάζα των στοιχείων που συλλέγονται από την επιχείρηση από τα στοιχεία των πελατών της, αυτό γίνεται για στοχευμένο μάρκετινγκ και τις οικονομικές δραστηριότητες. Μερικά από τα πλεονεκτήματά του είναι:

- Είναι μια αναλυτική πλατφόρμα για όλες τις προηγμένες αναλυτικές εφαρμογές.
- Παρέχει τις batch και τις σε πραγματικό χρόνο ικανότητες ανάλυσης και ανάσυρσις δεδομένων.
- Τα πρότυπα ανάσυρσις δεδομένων αυτοδιδασκαλίας βελτιώνονται αυτόματα με την πάροδο του χρόνου
- Τέλος το G-Stat Analytic Platform περιέχει 4 κύρια συστατικά:
    1. Analytical Data Repository for each Industry
    2. Analytical Applications
    3. Real-Time Interaction Recommendation Engines
    4. Implementation Support and Professional Consulting.

(G-Stat, 2009)

SPSS

Το SPSS είναι ένα λογισμικό το οποίο αρχικά φτιάχθηκε για να χρησιμοποιηθεί στον τομέα της στατιστικής ανάλυσης, αλλά επειδή όπως αναφέρθηκε η εξόρυξη δεδομένων έχει σχέση με την στατιστική, οι επόμενες εκδόσεις του προγράμματος παρέχουν τη δυνατότητα, μέσω ειδικών συναρτήσεων και αλγορίθμων, της εξόρυξης δεδομένων. Είναι και αυτό ένα λογισμικό που μπορεί να εγκατασταθεί σε όλα τα μεγάλα λειτουργικά συστήματα, χρησιμοποιείται κυρίως από ερευνητές για διάφορους σκοπούς και από ένα μεγάλο σύνολο επιχειρήσεων. Παρέχει ένα αρκετά φιλικό προς το χρήστη περιβάλλον εργασίας το οποίο (εάν είναι γνώστης της στατιστικής) μπορεί εύκολα να τον καθοδηγήσει για να κάνει τις μελέτες και να βρει τις στατιστικές ή τους κανόνες που τον ενδιαφέρουν. Παρέχει δυνατότητες σύνδεσης σε βάση δεδομένων μέσω ODBC και SQL. Τέλος, εάν και η πρώτη εφαρμογή που έγινε (επειδή ήταν σχετικά παλιά 1968) ήταν σχεδιασμένη να λειτουργεί με διάτρητες κάρτες. Τώρα πια το γραφικό περιβάλλον του SPSS 16.0 είναι γραμμένο σε Java. (Answers, 2009), (SPSS, 2009)

ThinkAnalytics

Η εφαρμογή της ThinkAnalytics η οποία ειδικεύεται στην εξόρυξη δεδομένων είναι η **Think EDM**. Η Think EDM (προηγουμένως γνωστή ως **K.wiz**) είναι μια πλήρως ενσωματωμένη πλατφόρμα αυτοματοποίησης επιχειρησιακής ανάλυσης ενσωματώνοντας τις τεχνικές ανακάλυψης γνώσης και εξόρυξης δεδομένων με σκοπό να παρέχουν στους χρήστες έξυπνες ικανότητες ανάλυσης. Συνδυάζει την χρηστικότητα με την επιχειρηματική επιλεξιμότητα για να αποκαλύψει τη κρυμμένη γνώση από τα μεγάλα ποσά δεδομένων.

Είναι μια ανοικτή πλατφόρμα ανακάλυψης γνώσης και εξόρυξης δεδομένων που χρησιμοποιεί τα πρότυπα της βιομηχανίας όπως είναι το PMML, το HTML, XML και ODBC/JDBC. Το σύστημα είναι cross platform και στοχεύει στα πολλαπλά επιχειρησιακά κανάλια. Η think EDM πλατφόρμα παρέχει μια διεπαφή

προγραμματισμού εφαρμογής (API), συμπεριλαμβανομένης της συστατικής βιβλιοθήκης. Μπορεί να επεκταθεί πέρα της μιας αρχιτεκτονική client/server. Δεν υπάρχει κανένας περιορισμός στη πρόσβαση των δεδομένων. Μια σειρά στοιχείων του Think EDM παρέχει πρόσβαση σε εύρος σχημάτων δεδομένων, συμπεριλαμβανομένων των κύριων πλατφορμών RDBMS - DB2, Oracle, SQL Server και Teradata. Το Data Dictionary παρέχει ένα ευφυές στρώμα μεταδεδομένων και προστατεύει το χρήστη από τον αντίκτυπο της μετακίνησης στοιχείων. Η ανοικτή φύση και η δυνατότητα επέκτασης του Think EDM σημαίνουν ότι η λειτουργία είναι απεριόριστη και μπορεί να ταιριάξει με τις απαιτήσεις και των χρηστών και των εφαρμογών, και να ενσωματωθεί στις επιχειρησιακές διαδικασίες.

Η think EDM βιβλιοθήκη ενσωματώνει πάνω από 200 επιμέρους συστατικά, που ομαδοποιούνται σε σχέση με τη λειτουργία τους. Αυτές οι ομάδες περιλαμβάνουν:

- Πρόσβαση στοιχείων
- Μείωση στοιχείων
- Μετασχηματισμός στοιχείων
- Ελλείπουσες τιμές
- Στοιχεία καλαθιών
- Μεταδεδομένα
- Ημερομηνία και χειρισμός σειράς
- Geographics
- Αλγόριθμοι & στατιστικές τεχνικές
- Επιχειρησιακοί κανόνες
- Στατιστικές δοκιμές

- Συνοπτικές στατιστικές
- Visualization
- Μετασχηματισμός XML
- Connectors.

(Thinkanalytics, 2009)

Unica

Η εταιρία Unica μέσω της **Affinium Suite** προσφέρει ένα εργαλείο εξόρυξης δεδομένων το *Affinium Model*. Αυτό όπως αναφέρθηκε είναι ένα εργαλείο για εξόρυξη δεδομένων και για μοντέλα προβλέψεων. Παρέχει πλήρη υποστήριξη σε όλες τις διαδικασίες εξόρυξης δεδομένων από την εισαγωγή στοιχείων έως την ανάλυση στοιχείων, την πρότυπη επέκταση, και το scoring. Έχει ενσωματωμένες αυτοματοποιήσεις και wizards για να βοηθήσει να παραχθούν τα πρότυπα γρηγορότερα από τα συμβατικά εργαλεία. Το Affinium Model αποτελείται από τέσσερα συστατικά: προβλέπει τη νομισματικές αξία και την αποδοτικότητα ενός ιδιαίτερου πελάτη με την πάροδο του χρόνου, βοηθάει στο να αποφασιστούν συγκεκριμένες διακριτές ιδιότητες για τις ομάδες πελατών όπως οι αγοραστές εναντίον μη-αγοραστές, σημειώνει άτομα τα οποία είναι πιθανότερο να ανταποκριθούν στις διαφημίσεις και τέλος προβλέπει τι είναι πιθανό να αγοράσει κάθε πελάτης και με ποια σειρά. Τέλος είναι βασισμένο σε μια ανοιχτή αρχιτεκτονική που βοηθάει το χρήστη να κάνει και δικές του υλοποιήσεις και του επιτρέπει την αύξηση της υποδομής και των δεδομένων.

1.10 Με λίγα λόγια

Σε αυτό το κεφάλαιο έγινε μια περιγραφή του όρου της εξόρυξης των δεδομένων. Όπως είναι πολύ πιθανόν να καταλάβατε, σήμερα η εξόρυξη των δεδομένων είναι ένα από τα πιο σημαντικά πεδία στην Επιστήμη των Υπολογιστών και αυτό γιατί ο

όγκος των πληροφοριών όσο πάει και αυξάνεται. Είδαμε διάφορες τεχνικές με τις οποίες γίνεται εξόρυξη δεδομένων και επίσης είδαμε και την προσπάθεια που γίνεται για να δημιουργηθούν όσο το δυνατόν πιο έξυπνες μέθοδοι για την εξόρυξη, για να μπορέσει να γίνει μείωση του κόστους και του χρόνου που απαιτείται. Τέτοιες μέθοδοι-τεχνικές που έχουν σαν αναγνωριστικό στοιχείο τη νοημοσύνη που παρουσιάζουν είναι ο Περιορισμός του χώρου αναζήτησης των δεδομένων, η Ταξινόμηση και Ομαδοποίηση, η Εξαγωγή Κανόνων και τα Νευρωνικά Δίκτυα τα οποία περιέχουν και επιπλέον μεθόδους που χρησιμοποιούνται στην εξόρυξη δεδομένων. Τέλος επειδή όπως είδαμε οι επιχειρήσεις έχουν μεγάλο συμφέρον από την εξόρυξη δεδομένων διάφορες εταιρίες έχουν αναπτύξει εργαλεία από τα οποία τα πιο διαδεδομένα παρουσιάζονται και περιγράφονται Όπως είπαμε προηγουμένως η “έξυπνη” εξόρυξη των δεδομένων είναι σημαντική για να γίνει μείωση του κόστους και του χρόνου και ένας τρόπος για να γίνει έξυπνη εξόρυξη δεδομένων είναι να χρησιμοποιηθούν προγράμματα φτιαγμένα σε γλώσσα και με τη λογική του λογικού προγραμματισμού, για αυτό και στο επόμενο κεφάλαιο γίνετε αναφορά στο λογικό προγραμματισμό και στα προτερήματα τα οποία παρουσιάζει.

Κεφάλαιο 2

Λογικός Προγραμματισμός και Prolog

Ένας από τους ορισμούς της λογικής, στο ετυμολογικό λεξικό Webster 1913, είναι: *λογική είναι η επιστήμη ή η τέχνη της γενίκευσης, της κρίσης, της ομαδοποίησης, της συλλογιστικής σκέψης καθώς επίσης και των συστηματικών ρυθμίσεων*. Η λογική όμως από τη πλευρά της πληροφορικής, σύμφωνα με το The Free On-line Dictionary of Computing (2003), *είναι οτιδήποτε αφορά το τι είναι αληθές και πως μπορούμε να ξέρουμε αν κάτι είναι αληθές*. Ο Λογικός Προγραμματισμός είναι ο προγραμματισμός με λογική δομή. Περιλαμβάνει μορφοποιήσεις τον λογικών επιχειρημάτων και αποδείξεων σε όρους ή σύμβολα τα οποία θα δημιουργήσουν προτάσεις και λογικούς συνδέσμους. Το νόημα αυτών των λογικών συνδέσμων εκφράζεται μέσα από ένα σύνολο κανόνων που έχει γίνει η υπόθεση ότι ισχύουν. Σύμφωνα με αυτά μπορούμε να πούμε πως Λογικός Προγραμματισμός είναι ένα *δηλωτικό*, *σχεσιακό στυλ* προγραμματισμού. Λόγω της ιδιότητας του να βασίζεται στη λογική, είναι καθαρά δηλωτική γλώσσα, η ευθύνη για να είναι οι δηλώσεις αυτές σωστές επέρχεται στον προγραμματιστή. Για να βεβαιωθεί, ο προγραμματιστής, ότι το πρόγραμμα λειτουργεί σωστά, εφόσον έχει κατασκευάσει δηλώσεις οι οποίες είναι αληθείς, ελέγχει εάν είναι αυτές οι δηλώσεις αρκετές ώστε να μπορούν να παράγουν σωστές λύσεις για όλα τα προβλήματα τα οποία μας ενδιαφέρουν. Επιπλέον ο προγραμματιστής, για να μπορέσει να εξασφαλίσει ταχύτητα στο πρόγραμμά του, πρέπει να διασφαλίσει πως τα αποτελέσματα/συμπεράσματα που παράγονται, είναι σύντομα. Για να γίνει αυτό είναι πολύ πιθανόν να έχουν οι δηλώσεις να έχουν ειδική μορφοποίηση για να ανταποκρίνονται καλύτερα. (Spivey M., 1995)

Σε πολλές περιπτώσεις, για να επιτευχθεί η αποδοτικότητα, κάποιος πρέπει να γνωρίζει και να εκμεταλλευτεί τη συμπεριφορά επίλυσης του προβλήματος το οποίο έχουμε θέσει για να μας δοθεί η λύση του. Έχοντας αυτά υπ' όψιν μας, δε

μπορούμε να δούμε κάποιες σημαντικές διαφορές ανάμεσα στον λογικό προγραμματισμό και στον κλασσικό διατακτικό προγραμματισμό, όπου γίνεται χρήση των προγραμμάτων για να ελεγχθεί η συμπεριφορά ενός εκτελεστή προγράμματος. Παρ' όλα αυτά όμως, τα συμβατικά επιτακτικά προγράμματα, έχουν μόνο μια διαδικαστική ερμηνεία, ενώ αντίθετα τα προγράμματα λογικής έχουν επίσης μια δηλωτική, **λογική** ερμηνεία, η οποία βοηθά στο να εξασφαλίσει την ακρίβειά τους. Επιπλέον, τέτοια προγράμματα, που είναι δηλωτικά, είναι σε πιο υψηλό εννοιολογικό επίπεδο από τα καθαρώς επιτακτικά προγράμματα και οι εκτελεστές προγράμματός τους, αναπτύσσουν δραστηριότητες σε πιο υψηλό εννοιολογικό επίπεδο από τους συμβατικούς μεταγλωττιστές και τους διερμηνείς.

Η γλώσσα προγραμματισμού η οποία έχει καθιερωθεί και θεωρείται ως η γλώσσα λογικού προγραμματισμού είναι η Prolog, η οποία χρησιμοποιήθηκε και για την κατασκευή του προγράμματος.

2.1 Σύντομη Ιστορία της Prolog

Η Prolog όπως λέει και το όνομα της (*programmation en logique, προγραμματίζοντας με λογική),* το οποίο προτάθηκε από τη γυναίκα του Phillipe Roussel (Kowalski R., 1988), είναι μια γλώσσα λογικού προγραμματισμού. Δημιουργήθηκε από τους Alain Colmerauer και Phillipe Roussel, στη Marseille περί το 1972. Σημαντικό ρόλο στην ανάπτυξη της γλώσσας είχε παίξει η ερμεινία που έδωσε ο Kowalski για τις **φράσεις Horn**. Μιά από τις πρώτες εκδόσεις τις Prolog είναι η *Edinburgh Prolog* από το πανεπιστήμιο του Ενδιβούργου, όπου ήταν ο Kowalski, το 1977. Στα πρώτα στάδια της, η Prolog χρησιμοποιήθηκε κυρίως σε εφαρμογές **τεχνητής νοημοσύνης** και ήταν περιορισμένη σε ερευνητικά πλαίσια. Τώρα πλέον η Prolog χρησιμοποιείται σε ένα μεγάλο φάσμα από κλάδους εφαρμογών όπως είναι : οι *βάσεις δεδομένων*, η *ανάπτυξη μεταφραστών γλωσσών προγραμματισμού*, τα *έμπειρα συστήματα*, *η ρομποτική*, η *αναγνώριση έντυπης φυσικής γλώσσας* και τα *έξυπνα συστήματα διδασκαλίας*. (Σταμάτης Δ., 2007)

2.2 Δομή της Prolog

Η Prolog, επειδή όπως είπαμε είναι γλώσσα λογικού προγραμματισμού, είναι γλώσσα **υψηλού επιπέδου** και η δομή των προγραμμάτων είναι πιο κοντά στο δομή της ανθρώπινης σκέψης. Τα βασικά στοιχεία που χρησιμοποιούνται για να δομηθεί ένα πρόγραμμα στη Prolog είναι οι **φράσεις** και οι **λογικοί όροι**. Τις *φράσεις* τις αποτελούν τρείς κατηγορίες οι οποίες είναι τα **γεγονότα**, οι **κανόνες** και οι **ερωτήσεις**.

2.2.1 Γεγονότα

Τα *γεγονότα* είναι η πιο απλή μορφή φράσης στη Prolog. Στη πιο απλή περίπτωση ένα πρόγραμμα μπορεί να αποτελείται **μόνο** από γεγονότα. Το γεγονός δηλώνει ότι *μια σχέση αληθεύει ανάμεσα σε δύο αντικείμενα*. Η σχέση ονομάζεται **κατηγόρημα** και τα αντικείμενα ονομάζονται **ορίσματα** του κατηγορήματος. Το κάθε κατηγόρημα έχει και έναν αριθμό, που ονομάζεται **πληθικός αριθμός**, ο οποίος στην ουσία αντιπροσωπεύει τα ορίσματα που έχει. Αυτό γίνεται για να μπορεί κάποιος να αναφέρεται με ακρίβεια σε ένα κατηγόρημα, γιατί μπορεί να υπάρχουν και άλλα κατηγορήματα με το ίδιο όνομα αλλά διαφορετικό αριθμό, κάτι που τα κάνει τελείως ξεχωριστά από τα άλλα κατηγορήματα. (Σταμάτης Δ., 2007)

Συντακτικός, στη κατασκευή γεγονότων πρέπει να ακολουθούνται οι εξής κανόνες:

- Τα ονόματα των σχέσεων και των αντικειμένων πρέπει να αρχίζουν από ένα μικρό γράμμα του αλφαβήτου
- Η σχέση γράφεται πρώτη, και τα αντικείμενα γράφονται μέσα σε παρενθέσεις που ακολουθούν τη σχέση και χωρίζονται μεταξύ τους με κόμμα “,”.
- Ο χαρακτήρας “.” τοποθετείται στο τέλος του γεγονότος.

Επειδή υπήρχε μία καθυστέρηση στο να καθιερωθεί κάποιο standard, ο κατά ISO

καθορισμός των standards της Prolog (ISO/IEC JTC1 SC22 WG17) ολοκληρώθηκε το 1996, οι διάφορες υλοποιήσεις της Prolog διαφέρουν στη συντακτική γραφή των προγραμμάτων. (Deransart et al, 1996).

Κάποια παραδείγματα γεγονότων είναι τα εξής:

father(panagiotis, giannis). (1)

father(panagiotis, fotini). (2)

father(kostas, panagiotis). (3)

mother(eirini,fotini). (4)

Που αυτοί οι κανόνες επαληθεύουν τη σχέση μεταξύ πατέρα και παιδιού.

2.2.2 Κανόνες

Οι *κανόνες*, όπως είπαμε, είναι μία άλλη μορφή φράσεων της Prolog. Μέσω των κανόνων και χρησιμοποιώντας σχέσεις, τις οποίες θεωρούμε ότι ισχύουν, μπορούμε να δημιουργήσουμε επιπλέον σχέσεις στα αντικείμενα. Σε αντίθεση με τα γεγονότα των οποίων οι σχέσεις είναι εξ' ορισμού αληθής, οι σχέσεις τις οποίες δηλώνουν αληθεύουν **μόνο** με τη προϋπόθεση ότι ισχύουν κάποιες συνθήκες, οι οποίες είναι και αυτές σχέσεις. Τα μέρη τα οποία αποτελείται ένας κανόνας είναι τα εξής:

- τις συνθήκες ή τις υποθέσεις οι οποίες βρίσκονται δεξιά του κανόνα
- το συμπέρασμα το οποίο είναι αριστερά του κανόνα.

Το *συμπέρασμα* ονομάζεται και **κεφαλή** του κανόνα ενώ το *τμήμα συνθηκών* ονομάζεται και **σώμα** του κανόνα. Επίσης μπορούμε να αναφερόμαστε στις συνθήκες που αποτελούν το σώμα ενός κανόνα και σαν **στόχους**.

Η μορφή που ακολουθείται για τη κατασκευή των κανόνων πρέπει να ακολουθεί τα παρακάτω standard:

- Η *κεφαλή* πρέπει να χωρίζεται από το σώμα με το σύμβολο ":-".
- Εάν υπάρχουν περισσότερες από μία *συνθήκες* πρέπει να χωρίζονται μεταξύ τους με **κομμα ","**, το οποίο παίζει το ρόλο της σύζευξης δηλαδή επιτελεί την ίδια λειτουργία με το λογικό **ΚΑΙ(AND)**, το οποίο σημαίνει πως πρέπει όλες οι συνθήκες να είναι αληθής για να αληθεύει το συμπέρασμα.
- Το τέλος ενός κανόνα δηλώνεται με το σύμβολο της **τελείας ".".**

Οπτικά αυτό μεταφράζεται έτσι:

κεφαλή :-

συνθήκη1,
...
συνθήκηΝ. } **σώμα**

Επίσης, μπορούμε να ορίσουμε επιπλέον σχέσεις που δεν θα είναι αναγκαστικό να αναφέρονται σε κάποιο συγκεκριμένο αντικείμενο αλλά θα έχουν τη δυνατότητα να αναφέρονται σε ολόκληρες κατηγορίες αντικειμένων. Αυτό μπορεί να επιτευχθεί με τη χρήση μεταβλητών στη θέση των ορισμάτων των κατηγορημάτων που βρίσκονται στη κεφαλή ή και στο σώμα ενός κανόνα.

Παραδείγματα:

parent(X,Y):- mother(X,Y). (5)

parent(X,Y):- father(X,Y). (6)

grandfather(X,Y) :- father(X,Z), parent(Z,Y) (7)

Σε αυτό το παράδειγμα δηλώνεται πως για να είναι ο Χ γονιός του Υ τότε πρέπει

να ισχύει ο X να είναι ή μητέρα ή πατέρας του Y. Επίσης για να είναι παππούς του Y ο X πρέπει να είναι ο X πατέρας του Z που είναι γονιός του Y. Οι μεταβλητές αυτές που χρησιμοποιήθηκαν σαν ορίσματα σε κατηγορήματα που βρίσκονται στο σώμα του κανόνα χωρίς να χρησιμοποιούνται ονομάζονται **υπαρξιακά ποσοδεικτούμενες**, ενώ οι μεταβλητές που χρησιμοποιούνται σαν ορίσματα στο κατηγόρημα που αποτελεί κεφαλή μιας φράσης ονομάζονται **καθολικά ποσοδεικτούμενες**. (Δημοσθένης Σταμάτης, 2007)

2.2.3 Ερωτήσεις

Η τρίτη μορφή φράσεων στη Prolog είναι οι *ερωτήσεις*. Μέσω των ερωτήσεων ξεκινάει η εκτέλεση των προγραμμάτων. Η ερώτηση έχει τη συντακτική μορφή που έχει και το σώμα ενός κανόνα. Στο μόνο που διαφέρει από το σώμα του κανόνα είναι ότι στις ερωτήσεις προηγείται το σύμβολο “?-”. Για να απαντηθεί μια ερώτηση πρέπει να υπάρξει μια λογική συνεπαγωγή του προγράμματος.

Ένα βασικό χαρακτηριστικό των ερωτήσεων και γενικότερα της Prolog, είναι ότι *μια μεταβλητή μπορεί να πάρει τη θέση οποιουδήποτε ορίσματος ενός στόχου*. Ένα άλλο επίσης σημαντικό χαρακτηριστικό είναι η δυνατότητα της Prolog στις ερωτήσεις, να μπορεί να δώσει όλα τα πιθανά αποτελέσματα τα οποία αληθεύουν. (Σταμάτης Δ., 2007)

Μερικά παραδείγματα:

Οι παρακάτω ερωτήσεις και τα αποτελέσματα δίνονται σύμφωνα με τους κανόνες και τα γεγονότα (1)-(7).

?- father(panagiotis,giannis).

True.

?- mother(eirini,giannis).

False

?- parent(panagiotis,X)

X = giannis ; *(το ελληνικό ερωτηματικό δίνεται για να μας δοθούν επιπλέον απαντήσεις στην ερώτηση, εάν υπάρχουν)

X = fotini ;

no

?- parent(X,Y).

X = panagiotis Y = giannis ;

X = panagiotis Y = fotini ;

X = kostas Y = panagiotis ;

X = eirini T = fotini ;

no

Αυτές οι ερωτήσεις μας απαντούν στο αν ο panagiotis είναι πατέρας του giannis, εάν η eirini είναι μητέρα του giannis, σε ποιους είναι γονιός ο panagiotis και ποιοι είναι όλοι οι γονείς και τα παιδιά.

2.3 Θετικά Στοιχεία της Prolog

Όπως έχει αναφερθεί ο όγκος των πληροφοριών στον Παγκόσμιο Ιστό είναι αρκετά μεγάλος και για να τον επεξεργαστούμε χρειάζεται να κατασκευαστούν έξυπνα προγράμματα. Σε αυτό το υποκεφάλαιο θα ασχοληθούμε με τα στοιχεία της Prolog, όπως είναι η ενοποίηση, η αναδρομή κ.α., τα οποία βοηθούν στην ευκολότερη και πιο “έξυπνη” επεξεργασία αυτών των δεδομένων.

2.3.1 Ενοποίηση

Η *ενοποίηση* είναι ένα από τα βασικά στοιχεία για τη λειτουργία της Prolog, γιατί

αποτελεί τη κεντρική διαδικασία υπολογισμού της. Για να την οποιαδήποτε επεξεργασία που χρειάζεται να γίνει, κατά την εκτέλεση ενός προγράμματος, χρησιμοποιείται η διαδικασία της ενοποίησης. Για να μπορέσει να επιτύχει η ενοποίηση δύο όρων x,y, πρέπει να ισχύει κάτι από τα παρακάτω:

- ο ένας όρος να είναι μεταβλητή
- και οι δύο όροι να είναι η ίδια σταθερά
- ο x να είναι σύνθετος όρος με συναρτησιακό σύμβολο f και n ορίσματα, ο y να είναι επίσης σύνθετος όρος με συναρτησιακό σύμβολο f και n ορίσματα και τα n ορίσματα του x μπορούν να ενοποιηθούν με τα αντίστοιχα n ορίσματα του y.

Εφόσον επιτευχθεί η ενοποίηση παράγεται ο πιο γενικός ενοποιητής, ο οποίος είναι ο όρος που παράγεται από την ενοποίηση των δύο πρώτων όρων και περιέχει όσο δυνατόν περισσότερες από τις μεταβλητές του τοποθετημένες.

Γενικότερα, μια αντικατάσταση θ ονομάζεται ενοποιητής των όρων x και y εάν xθ = yθ, η δε εφαρμογή της θ πάνω στους x και y ονομάζεται **ενοποίηση**. Μια τέτοια αντικατάσταση ονομάζεται ο **πιο γενικός ενοποιητής** των x και y, αν για κάθε άλλο ενοποιητή θ1 των x και y το x1θ1 είναι ένα στιγμιότυπο του x1θ.

2.3.2 Πολλαπλή μορφή ορισμάτων I/O

Ένα πολύ θετικό στοιχείο στη Prolog, που αναδεικνύει τη νοημοσύνη της γλώσσας είναι η *πολλαπλή μορφή* που μπορούν να πάρουν τα ορίσματα εισόδου εξόδου των κατηγορημάτων. Με αυτό εννοούμε πως εάν κάποιο όρισμα έχει χαρακτηριστεί σαν όρισμα εισόδου και κάποιο άλλο σαν όρισμα εξόδου τότε μπορούν να χρησιμοποιηθούν και αντίθετα δηλαδή να δώσουμε στο όρισμα της εξόδου τι περιμένουμε και να μας εμφανίσει ποια είναι τα ορίσματα που πρέπει να δώσουμε στην είσοδο για να έχουμε αυτό το αποτέλεσμα. Αυτή η δυνατότητα της Prolog, μας βοηθάει στο να μη κατασκευάζουμε κατηγορήματα τα οποία το ένα να

κάνει την αντίστροφη δουλεία από το άλλο και με αυτό τον τρόπο γλιτώνουμε κόπο και πολυπλοκότητα γιατί δε γράφουμε μεγάλο κώδικα.

Δίνεται ένα παράδειγμα από το manual της Swi-Prolog που ίσως βοηθήσει να γίνει πιο κατανοητή αυτή η έννοια:

Το κατηγόρημα της ***atom_chars/2*** είναι ένα κατηγόρημα το οποίο δέχεται ένα *atom* σαν όρισμα εισόδου και δίνει μία λίστα με τους χαρακτήρες εξόδου

?- atom_chars(hello, X).

X = [h, e, l, l, o]

Μπορεί όμως να χρησιμοποιηθεί και αντίθετα, δίνοντας στο δεύτερο όρισμα μία λίστα χαρακτήρων έτσι ώστε να παραχθεί ένα atom

?- atom_chars(X, [h, e, l, l, o]).

X = hello.

2.3.3 Αναδρομή

Η *αναδρομή* στον προγραμματισμό είναι μία από τις πιο συνηθισμένες μεθόδους επίλυσης προβλημάτων, εντάσσεται στη κατηγορία των μεθόδων **διαίρει και βασίλευε** και αποτελεί ένα από τα βασικά στοιχεία στον δυναμικό προγραμματισμό γιατί έχει χρησιμοποιηθεί για να σχεδιαστούν πολλοί σημαντικοί αλγόριθμοι. Τα αναδρομικά προγράμματα χωρίζουν το πρόβλημα σε υποπροβλήματα, τα οποία στην ουσία είναι το αρχικό πρόβλημα σε μικρότερη κλίμακα και για την επίλυσή τους γίνεται κλήση του εαυτού τους. Για να σταματήσει η αναδρομή γίνεται χρήση **τερματικών συνθηκών** οι οποίες εκφράζουν το σημείο στο οποίο εάν φθάσει ο αλγόριθμος υπάρχει λύση. Οι τερματικές συνθήκες δε πρέπει **ποτέ** να περιέχουν αναδρομικές κλήσεις.

Στη Prolog και γενικότερα στο λογικό προγραμματισμό, η αναδρομή γίνεται πολύ

εύκολα σε σχέση με άλλες γλώσσες. Αυτό γίνεται γιατί στη Prolog χρησιμοποιείται **αναδρομή ουράς**, οπότε το πρόβλημα θεωρείται είδη λυμένο για την ουρά, άρα πρέπει μόνο να βρει τρόπο ο προγραμματιστής να αντιμετωπίσει το πρόβλημα για τη κεφαλή και επειδή ο κώδικας που γράφεται στη Prolog και η λογική που ακολουθείται για τη κατασκευή των προγραμμάτων είναι κοντά στην ανθρώπινη λογική συνήθως οι προγραμματιστές δεν αντιμετωπίζουν σοβαρά προβλήματα σε αυτό το σημείο. Το μόνο σημείο το οποίο ίσως δυσκολεύει είναι ο ορισμός της τερματικής συνθήκης αλλά αυτό δεν είναι μόνο πρόβλημα στη Prolog αλλά γενικότερο και εξαρτάται κυρίως από το προγραμματιστή που κατασκευάζει το πρόγραμμα και τη “νοημοσύνη” που ο ίδιος διαθέτει γιατί τα προγράμματα είναι τόσο έξυπνα όσο αυτός που τα γράφει.

2.3.4 Αυτόματη οπισθοδρόμηση

Η *οπισθοδρόμηση* σύμφωνα με το The Free on-line Dictionary of Computing (2003), *είναι ένα σχέδιο για την επίλυση μιας σειράς υποπροβλημάτων κάθε ένα από τα οποία μπορεί να έχει πολλές πιθανές λύσεις και όπου η λύση που επιλέγεται για ένα υποπρόβλημα μπορεί να επιφέρει διαφορετικά αποτελέσματα στις πιθανές λύσεις των πιο πρόσφατων υποπροβλημάτων*.

Για να λύσουμε το γενικό πρόβλημα, βρίσκουμε μια λύση στο πρωταρχικό υποπρόβλημα και έπειτα γίνεται προσπάθεια να επιλυθούν, κατ' επανάληψη, τα άλλα υποπροβλήματα βασισμένα σε αυτήν την πρώτη λύση. Εάν δεν μπορούμε, ή θέλουμε όλες τις πιθανές λύσεις, οπισθοδρομούμε και δοκιμάζουμε την επόμενη πιθανή λύση στο πρώτο υποπρόβλημα. Η όλη διαδικασία ολοκληρώνεται όταν δεν υπάρχουν άλλες λύσεις στο πρώτο υποπρόβλημα.

Η οπισθοδρόμηση είναι ένας αλγόριθμος που χρησιμοποιείται και στο λογικό προγραμματισμό όπως στη Prolog για να βρει όλους τους πιθανούς τρόπους της παρουσίασης αποδείξεων ενός στόχου. Στη Prolog αυτή η διαδικασία γίνεται αυτόματα, δηλαδή δίνοντας μια ερώτηση έτσι ώστε να ξεκινήσει το πρόγραμμα η Prolog ψάχνει όλα τα γεγονότα και τους κανόνες, με τη σειρά την οποία είναι

γραμμένα μέχρις ότου να βρει κάποιο γεγονός η κεφαλή κανόνα που να ταιριάζει με την ερώτηση. Όταν μπορέσει να βρει κάτι που να ταιριάζει τότε το αποθηκεύει το σημείο αυτό και από εκεί ξεκινάει μια νέα αναζήτηση μέσα στο κανόνα/γεγονός για να βρει αν αληθεύει η σχέση, εάν δεν αληθεύει τότε γυρνάει στο σημείο το οποίο αποθήκευσε και ψάχνει να βρει άλλα γεγονότα/κανόνες που να ταιριάζουν. Ένας τρόπος για να σταματήσει η οπισθοδρόμηση είναι χρησιμοποιώντας το κατηγόρημα της αποκοπής **!/0**, το οποίο είναι ενσωματωμένο κατηγόρημα της Prolog.

Ένα παράδειγμα χρησιμοποιώντας τους κανόνες και τα γεγονότα (1)-(7):

```
?- mother(X,_).

  X = eirini ;

  no.
```

Σε αυτό το παράδειγμα θα ψάξει τα γεγονότα (1)-(3) και δε θα μπορέσει να κάνει ενοποίηση και όταν βρει το γεγονός (4) θα δει ότι αληθεύει και θα το επιστρέψει και συνεχίζοντας και στα υπόλοιπα γεγονότα θα δει ότι δεν αληθεύει κανένα άλλο.

```
?- grandfather(X,_).

  X = kostas ;

  no.
```

Αυτή είναι μια διαφορετική περίπτωση. Για να βρει η Prolog ποιες σχέσεις αληθεύουν ψάχνει πρώτα να βρει γεγονότα father(X,Z). Το πρώτο γεγονός που βρίσκει να αληθεύει είναι το γεγονός (1). Αποθηκεύει αυτό το σημείο και στη συνέχεια ψάχνει να βρει γεγονός parent(Z,_) που να αληθεύει. Σε αυτή τη περίπτωση δε βρίσκει κανένα οπότε κάνει οπισθοδρόμηση και συνεχίζει από το επόμενο σημείο το οποίο είχε αποθηκεύσει, δηλαδή το γεγονός (2). Το ίδιο συμβαίνει και σε αυτό το γεγονός οπότε κάνει οπισθοδρόμηση και συνεχίζει στο γεγονός (3) μέσω του οποίο βρίσκει parent(Z,_) το οποίο αληθεύει οπότε και

επιστρέφει kostas, που είναι το X στο father(X,Z).

2.4 Με λίγα λόγια

Σε αυτό το κεφάλαιο αναφερθήκαμε στο λογικό προγραμματισμό, στη γλώσσα, που έχει πλέον καθιερωθεί σαν γλώσσα λογικού προγραμματισμού, τη Prolog, γράφοντας μια σύντομη ιστορία της και αναφέροντας τα κύρια δομικά στοιχεία που χρειάζεται κάποιος να ξέρει για να γράψει ή να κατανοήσει ένα πρόγραμμα της Prolog. Τέλος παρουσιάστηκαν και τα προτερήματα τα οποία έχει ο λογικός προγραμματισμός σε σχέση με τις άλλες μεθοδολογίες προγραμματισμού. Τα προτερήματα αυτά είναι η ενοποίηση, πολλαπλή μορφή που μπορούν να έχουν οι παράμετροι εισόδου εξόδου των ορισμάτων, η αναδρομή και η ευκολία κατασκευής αναδρομικών κατηγορημάτων στη Prolog και τέλος στην αυτόματη οπισθοδρόμηση που προσφέρει η Prolog.

Από το επόμενο κεφάλαιο ξεκινάει η τεκμηρίωση του προγράμματος για να μπορέσει ο αναγνώστης να καταλάβει τι κάνει το πρόγραμμα και κυρίως πως το κάνει. Πιο συγκεκριμένα το επόμενο κεφάλαιο περιγράφει το τρόμο με τον οποίο γίνεται η σύνδεση της Prolog στο Παγκόσμιο ιστό, πως γίνεται η ανάκτηση κειμένου και γιατί και επίσης αναφέρονται και μερικά σημαντικά προβλήματα που αντιμετωπίστηκαν και πως αντιμετωπίστηκαν για να λειτουργεί ομαλά το πρόγραμμα.

Κεφάλαιο 3

Σύνδεση στον Παγκόσμιο Ιστό και Ανάκτηση Κειμένου

Αναφέρθηκε νωρίτερα πως θα χρησιμοποιηθεί ο ιστότοπος της Wikipedia για το σχηματισμό των ερμηνειών που θα υπάρχουν στο γλωσσάρι. Για να μπορέσει να γίνει η ανάκτηση σελίδων από τον ιστότοπο της Wikipedia έπρεπε να λυθούν 4 προβλήματα:

1. Σύνδεση της Prolog με το Διαδίκτυο,
2. Εισαγωγή των όρων αναζήτησης
3. Εύρεση της σελίδας/όρου αναζήτησης και
4. Ανάκτηση του κατάλληλου κειμένου.

3.1 Σύνδεση της Prolog με το Διαδίκτυο

Για να συνδεθεί η Prolog στο Διαδίκτυο χρησιμοποιήθηκε, από βιβλιοθήκη της Swi-Prolog, το κατηγόρημα ***http_open/3*** που βρίσκεται στη βιβλιοθήκη **http/http_open**. Το κατηγόρημα ***http_open(+URL,-Stream,+Options)*** δέχεται ως ορίσματα εισόδου το url της σελίδας της οποίας θέλουμε να ανακτήσουμε και μία λίστα από όρους, μέσω των οποίων μπορεί κάποιος να δώσει παραμέτρους π.χ. τι μέθοδο να χρησιμοποιήσει(*get* ή *head*), να κάνει timeout μετά το πέρας κάποιου χρόνου κ.ά. Σαν έξοδο δίνει ένα Stream το οποίο περιέχει τον html κώδικα της σελίδας της οποίας δώσαμε το url.

3.2 Εισαγωγή των όρων αναζήτησης

Για να περαστούν οι όροι αναζήτησης κατασκευάστηκε το κατηγόρημα ***insert_to_list/2***, το οποίο χρησιμοποιεί το module **reading** ,που περιέχει τους κανόνες ***reading/2, iread/2, iread/3*** και το γεγονός **special_characters**. Ο λόγος κατασκευής αυτού του module είναι για να μπορεί κάποιος να δίνει χαρακτήρες που από τη Prolog έχουν χαρακτηριστεί ειδικοί και αν χρησιμοποιηθούν τότε εμφανίζεται σφάλμα. Το ***reading(-Chars)***, είναι το κατηγόρημα το οποίο μπορεί να χρησιμοποιηθεί και από άλλα modules και προγράμματα της Prolog. Έχει ως σκοπό την ενεργοποίηση του ***iread/2*** και την επιστροφή των χαρακτήρων που διαβάστηκαν από το χρήστη. Το ***iread(+Char,-Char)*** δέχεται ένα χαρακτήρα σαν είσοδο και καλεί το ***iread/3***, έτσι ώστε να επιστρέψει στο ***reading*** τους χαρακτήρες. Τέλος το ***iread(+Char,+Chars,-Char)*** παίζει το ρόλο παραμέτρου συσσώρευσης. Δέχεται δύο ορίσματα εισόδου, τα οποία είναι ένας χαρακτήρα και τους χαρακτήρες που έχει διαβάσει μέχρι τώρα. Διαβάζει ένα νέο χαρακτήρα, ο οποίος αν δεν είναι τελεία και δεν ανήκει στους ειδικούς χαρακτήρες προσθέτεται στους μέχρι στιγμής αναγνωσμένους, και καλεί τον εαυτό της για να επιστρέψει όλους τους αναγνωσμένους χαρακτήρες, μέχρι τη συνθήκη τερματισμού. Όπως γίνεται κατανοητό το ***iread/3*** λειτουργεί αναδρομικά, και έχει ως συνθήκη τερματισμού την τελεία ".", δηλαδή όταν βρει χαρακτήρα τελεία τότε σταματάει.

Στη συνέχεια χρησιμοποιείται το module **reading2**, γιατί η είσοδος των όρων πρέπει να έχει μια συγκεκριμένη μορφοποίηση (*όρος1[εσ.όρος1,εσ.όρος2],όρος2[]...,όροςN[εσ.όρος,...].*) και για να γίνουν αυτές οι αναζητήσεις πρέπει να γίνει διαχωρισμός των όρων. Το όρισμα εισαγωγής είναι έτσι τυποποιημένο ώστε να μπορεί ο χρήστης να δίνει όσους όρους για αναζήτηση επιθυμεί σε μία γραμμή. Επίσης μπορεί να δώσει και επιπλέον εσωτερικά ορίσματα έτσι ώστε, εάν υπάρχουν στο κείμενο που έχει δημιουργηθεί για τον γενικό όρο, να εμφανίζεται και μια περιγραφή και για αυτούς. Ο κανόνας ***reading2(+String, -ListA, -ListB)***, με το που δεχθεί τη συμβολοσειρά τη μετατρέπει σε μια λίστα από τους ASCII κωδικούς της συμβολοσειράς. Εφόσον

γίνει αυτή η μετατροπή καλείται ο κανόνας ***divide/3***. Το ***divide(+List, -ListA, -ListB)*** ξεχωρίζει τους εξωτερικούς από τους εσωτερικούς όρους ελέγχοντας αν πού υπάρχει ο χαρακτήρας 91, ο οποίος είναι ο χαρακτήρας "[". Μετά ψάχνει να βρει που υπάρχει ο χαρακτήρας 93, που είναι ο χαρακτήρας "]", για να ξεχωρίσει τους εξωτερικούς από τους εσωτερικούς όρους (και να τους επιστρέψει τους εξωτερικούς στη ListA και τους εσωτερικούς στη ListB). Τέλος καλείται αναδρομικά αυτό το κατηγόρημα για να γίνει αυτή η διεργασία για όλα τα ορίσματα. Η τερματική συνθήκη ενεργοποιείται όταν δεν υπάρχει ο χαρακτήρας 91 στη λίστα που έχει δοθεί σαν λίστα εισόδου και τότε επιστρέφει δύο κενές λίστες. Τέλος το ***reading2*** μετατρέπει τις δύο λίστες, που δέχθηκε από το ***divide,*** από χαρακτήρες ASCII σε λίστα συμβολοσειρών μέσω των ***list_to_text/2*** και ***lists_to_text/2***. Το ***list_to_text(-List,+List)*** μετατρέπει τη λίστα από τους ASCII κωδικούς σε μια λίστα από λέξεις. Το ***lists_to_text(-List, +List)*** μετατρέπει τη λίστα από λίστες χαρακτήρων ASCII μέσω της ***list_to_text*** σε λίστα από λίστες λέξεων.

Εφόσον πλέον έχουμε τους όρους για την αναζήτηση, πρέπει απλώς να τους φέρουμε στη κατάλληλη μορφή για να μπορέσει να ξεκινήσει η αναζήτηση. Αυτή τη στιγμή τα ορίσματα είναι σε μια λίστα σε μορφή συμβολοσειράς, που μερικές φορές μπορεί να ξεκινάει με κόμμα και πρέπει να αφαιρεθούν τα κόμματα και να γίνουν *atoms*. Στην αρχή γίνεται συνένωση όλων των όρων της λίστας των εξωτερικών/κυρίων όρων αναζήτησης, σε μία συμβολοσειρά (η λίστα των εσωτερικών όρων στις εσωτερικές λίστες έχει αυτή τη μορφή και έτσι δε χρειάζεται να μετατραπούν και αυτές) μέσω του ***make_string/2***. Μετά χρησιμοποιώντας το module **make_words,** μέσω του κανόνα ***words/2,*** δίνουμε μια λίστα με τη συμβολοσειρά των ορισμάτων και μας επιστρέφεται μια λίστα με τα ορίσματα χωρισμένα στη σωστή μορφή. Το ***words(+List,-List)*** δέχεται μια λίστα με συμβολοσειρές και τις αλλάζει σε λέξεις, έχοντας ως διαχωριστή το κόμμα "," και επιστρέφει μια λίστα με αυτές τις λέξεις. (επιπλέον πληροφορίες για το πως λειτουργεί το make_words στο Κεφάλαιο 5)

Για τους εσωτερικούς όρους που αναφέρθηκαν ανωτέρω ότι γίνεται η ίδια δουλειά,

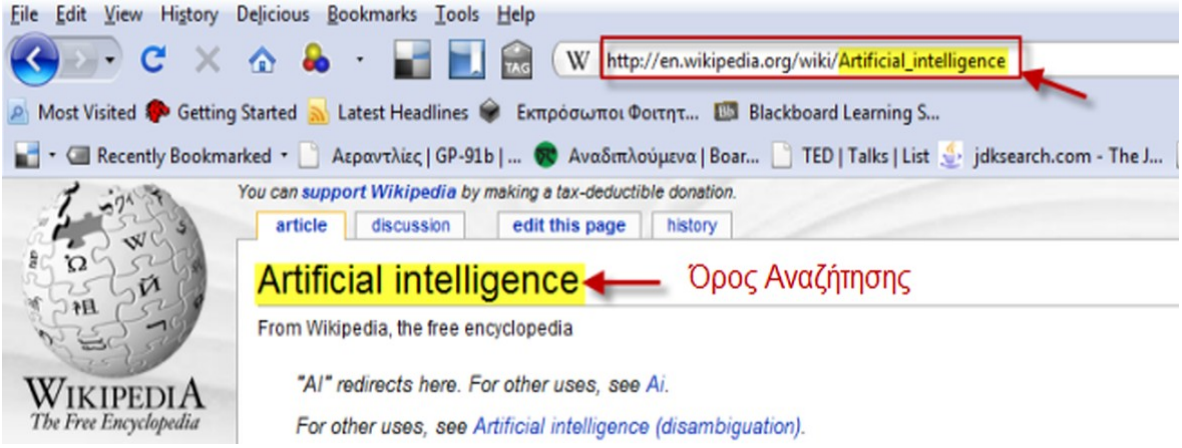
αυτή γίνεται από το κανόνα ***words2/2*** το οποίο δίνει στο ***words/2*** τις εσωτερικές λίστες ώστε να μετατραπούν οι συμβολοσειρές σε *atoms*.

Εφόσον πλέον έχει ολοκληρωθεί η εισαγωγή και επεξεργασία των όρων, οι δύο αυτές λίστες επιστρέφονται από το ***insert_to_list(-ListA,-ListB)***.

3.3 Εύρεση της σελίδας/όρου αναζήτησης

Για να την εύρεση και το "κατέβασμα" των σελίδων από το Διαδίκτυο είναι υπεύθυνο το module **open_pages**, μέσω του οποίου χρησιμοποιήθηκε το κατηγόρημα ***open_pages/4***. Όσον αφορά για το ποια θα είναι η διεύθυνση που πρέπει να δοθεί στο ***http_open***, που αναφέρθηκε προηγούμενος, μέσω παρατήρησης βλέπουμε πως η Wikipedia για κάθε της όρο χρησιμοποιεί ένα standard για να γίνεται η αναζήτηση. Αυτό το standard φαίνεται παρακάτω στην Εικόνα 5, και η μορφή που πρέπει να έχει το url για τους διάφορους όρους που αναζητούνται είναι η εξής:

http://en.wikipedia.org/wiki/Όρος (για την αγγλική εκδοχή της Wikipedia).



Εικόνα 5. Standard μορφής url σε σχέση με Όρο αναζήτησης

Σε περίπτωση που υπάρχει κενό αυτό αντικαθιστάται με την κάτω παύλα(_). Η Wikipedia είναι *case sensitive,* έκτος από το πρώτο γράμμα. Σε μερικές περιπτώσεις οι όροι της αναζήτησης, πρέπει στη url να έχουν κεφαλαία. Επειδή δε βρέθηκε κάποιο standard για τη συγκεκριμένη περίπτωση και επειδή αυτές οι περιπτώσεις είναι αρκετά λιγότερες από τις "κανονικές", ο χρήστης είναι αυτός που

πρέπει να δίνει με κεφαλαία τα ορίσματα, αλλιώς θα εμφανίσει πως η σελίδα δεν υπάρχει.

Το ***open_pages(+ListA,+ListB,-ListC,-ListD)*** είναι ένα κατηγόρημα που δέχεται δύο λίστες σαν ορίσματα, τη ListA η οποία περιέχει τους όρους που απομένουν να αναζητηθούν και τους όρους που έχει γίνει ήδη αναζήτηση για αυτούς. Κάθε φορά, επειδή η λειτουργία του κατηγορήματος ***open_pages*** είναι αναδρομική, παίρνει τη κεφαλή της ListA και κάνει κάποιες τελευταίες τροποποιήσεις στον όρο αναζήτησης, για να έρθει στην τελική του μορφή. Αυτές οι τροποποιήσεις γίνονται μέσω των κανόνων ***convert_space/2*** για να γίνει η αλλαγή του κενού, που χρειάζεται η Wikipedia, σε κάτω παύλα. Η χρήση του ***convert_space*** γίνεται μέσω του module **convert_predicates**. Το ***convert_space(+Atom,-Atom)*** δέχεται σαν όρισμα το *atom,* με τα κενά, το κάνει μία λίστα από χαρακτήρες, χρησιμοποιώντας το κατηγόρημα της Prolog ***atom_to_chars/2***. Μετά μέσω του κατηγορήματος ***space_to_underscore/2*** ελέγχει που υπάρχει κενός χαρακτήρας και τον αλλάζει σε κάτω παύλα. Τέλος αφού επιστραφεί η λίστα με τους χαρακτήρες ξανά μετατρέπεται σε *atoms* μέσω του ***atom_to_chars***.

Αφού έχει τελειώσει και αυτή η φάση της επεξεργασίας των ορισμάτων, γίνεται ένας έλεγχος εάν το τρέχων όρισμα είναι μέλος της λίστας των όρων που έχει γίνει ήδη αναζήτηση και εάν είναι τότε συνεχίζει η αναδρομή στον επόμενο όρο. Εάν δεν είναι τότε γίνεται η συνένωση του standard της Wikipedia (http://en.wikipedia.org/wiki/) με τον τροποποιημένο όρο κάνοντας χρήση του κατηγορήματος της Prolog ***concat/3***.

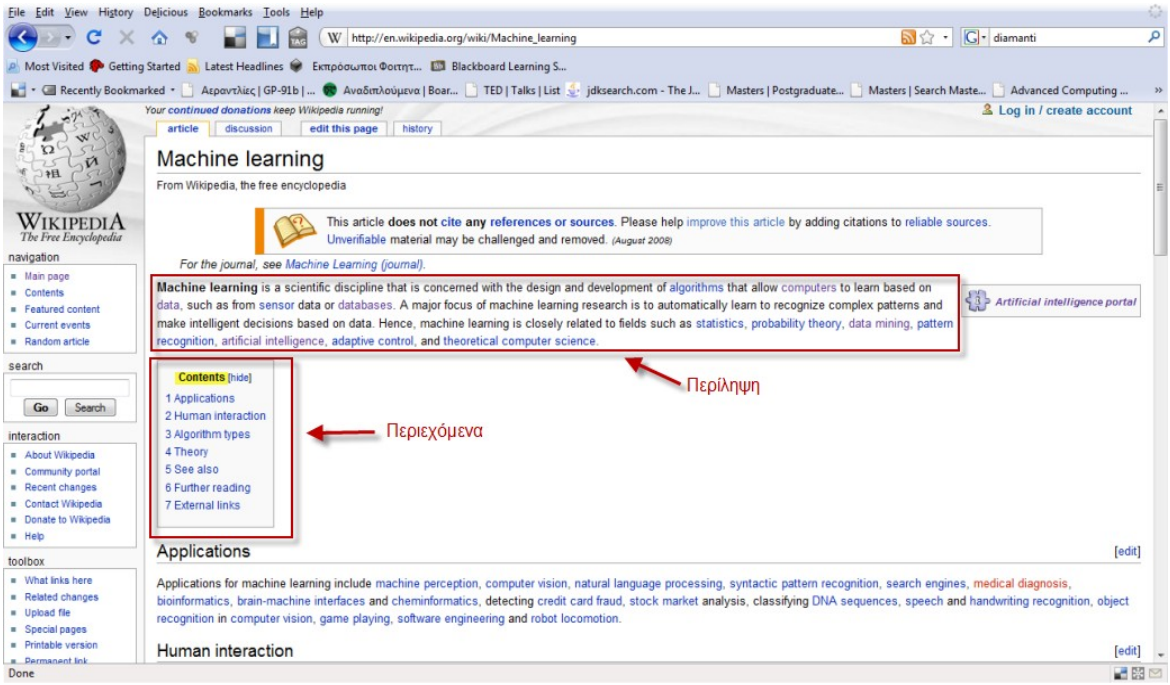
Έχοντας φτάσει πλέον στο σημείο να υπάρχει η διεύθυνση του όρου για αναζήτηση η επόμενη κίνηση είναι να δοθεί σαν όρισμα στο ***http_open*** για να γίνει η ανάκτηση του html κώδικα.(Το μόνο όρισμα που δίνεται είναι το *user_agent(wikipedia)* για να ξέρει το κατηγόρημα πως πρόκειται για τον ιστότοπο wikipedia).

3.4 Ανάκτηση του κατάλληλου κειμένου

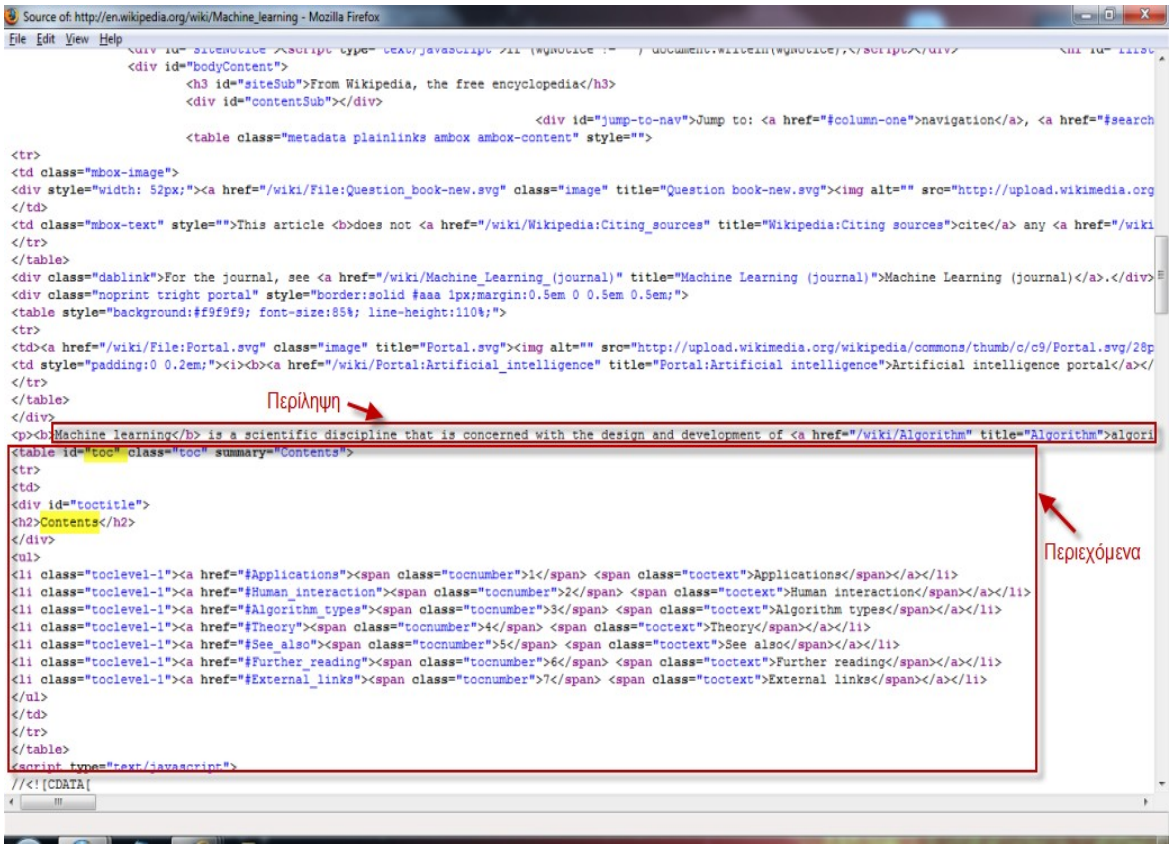
Μετά από παρατήρηση μεγάλου πλήθους λημμάτων της Wikipedia παρατηρήθηκε πως η συντριπτική πλειοψηφία των λημμάτων έχει περιεχόμενα και όσα λήμματα έχουν περιεχόμενα, έχουν και περίληψη.

Η περίληψη αυτή περιέχει τις πιο σημαντικές πληροφορίες για το λήμμα και έτσι θεωρήθηκε σωστό να γίνεται ανάκτηση αυτού του κειμένου, που περιέχει μια πλήρη περίληψη για το λήμμα, για τη χρήση του στο γλωσσάρι το οποίο θα δημιουργηθεί.

Η περίληψη όπως αναφέρθηκε και όπως μπορούμε να δούμε στην Εικόνα 6 είναι πριν από τα περιεχόμενα. Για να γίνει η ανάκτηση της περίληψης πρέπει να βρεθεί η θέση του κώδικά της στον html κώδικα. Ο κώδικας της περίληψης βρίσκεται και αυτός πριν από των κώδικα των περιεχομένων όπως φαίνεται στην Εικόνα 7.



Εικόνα 6. Θέση περίληψης και περιεχομένων στη σελίδα



Εικόνα 7. Θέση περίληψης και περιεχομώνων στον html κώδικα

Για να μπορέσει να γίνει αυτή η ενέργεια (ανάκτηση συγκεκριμένου κομματιού κώδικα) έπρεπε πρώτα να γίνει μια τροποποίηση του html κώδικα, για να γίνει δυνατή η πιό εύκολη επεξεργασία του. Η αλλαγή αυτή γίνεται μέσω της βιβλιοθήκης της Swi-Prolog **sgml**, δια μέσω του κατηγορήματος ***load_structure/3***. Το κατηγόρημα ***load_structure(+Source,-List,+Options)*** δέχεται σαν όρισμα εισόδου το stream που παράχθηκε από το http_open, που περιέχει τον html κώδικα, και μια λίστα με Options, τα οποία είναι για να καθορίσουν τη μορφή της επιστρεφόμενης λίστας (στη συγκεκριμένη περίπτωση έχουν δοθεί τα ορίσματα *space(remove),max_errors(-1), syntax_errors(quiet)* τα οποία είναι για να αφαιρούνται τα κενά από τη λίστα, να μην υπάρχουν λάθη κατά τη μετατροπή του html κώδικα και τέλος να μην εμφανίζονται τα συντακτικά λάθη που τυχόν υπάρχουν). Η μετατροπή που γίνεται από αυτό το κατηγόρημα είναι η εξής: δημιουργεί μια λίστα, η οποία περιέχει στοιχεία τύπου ***element/3***, που ακολουθούν την ενθυλάκωση του html κώδικα. Το ***element(?Tag,?Identifier,?List)*** στην ουσία

περιέχει το tag, τα προσδιοριστικά του tag και το περιεχόμενο σε μορφή λίστας του tag. Η ενθυλάκωση γίνεται στη λίστα και ακολουθείται ο ίδιος τρόπος. Εάν βρεθεί tag τότε γίνεται ***element*** και εάν βρεθεί κείμενο γίνεται συμβολοσειρά. Ένα παράδειγμα της μορφής του ***element*** είναι στις εικόνες 8 και 9 όπου στην Εικόνα 8 φαίνεται ο html κώδικα και στην Εικόνα 9 φαίνεται η τροποποίηση του σε στοιχείο ***element***.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">

<html>
<head>
<title>Demo</title>
</head>
<body>

<h1 align=center>This is a demo</title>

Paragraphs in HTML need not be closed.

This is called `omitted-tag' handling.
</body>
</html>
```

Εικόνα 8. Html κώδικας μιας τυχαίας σελίδας

```
?- load_html_file('test.html', Term),
   pretty_print(Term).

[ element(html,
          [],
          [ element(head,
                    [],
                    [ element(title,
                              [],
                              [ 'Demo'
                              ])
                    ]),
            element(body,
                    [],
                    [ '\n',
                      element(h1,
                              [ align = center
                              ],
                              [ 'This is a demo'
                              ]),
                      '\n\n',
                      element(p,
                              [],
                              [ 'Paragraphs in HTML need not be closed.\n'
                              ]),
                      element(p,
                              [],
                              [ 'This is called `omitted-tag\' handling.'
                              ])
                    ])
```

Εικόνα 9. Μετατροπή του html κώδικα της Εικόνας 8 σε στοιχείο element

Για να καταφέρουμε να πάρουμε το κείμενο πριν από τον πίνακα περιεχομένων δημιουργήθηκε το module **get_p_tags** και ο κανόνας ***get_tags/2***. Στο ***get_tags(+List,-List,+Atom,-Atom,+Boolean)***, που λειτουργεί και αυτό αναδρομικά, διοχετεύεται η λίστα με τα ***element***. Ελέγχουμε αν η κεφαλή είναι ***element***, εάν δεν είναι τότε γίνεται αναδρομή για την ουρά της λίστας. Εάν είναι ***element*** αλλά δεν έχει *tag = p* γίνεται διπλή κλήση της ***get_tag***, δίνοντας σαν λίστα πρώτα την εσωτερική λίστα του element και μετά την ουράς της αρχικής λίστας, που είχε δοθεί σαν όρισμα. Τέλος εάν είναι ***element*** και έχει *tag = p*, γίνεται ένα "φιλτράρισμα" αφαιρώντας τα εσωτερικά tags, χρησιμοποιώντας το κανόνα ***remove_inner_tags/2***. Το ***remove_inner_tags(+ListA,-ListB)*** δέχεται τη ListA και αν είναι η κεφαλή της ***element*** συνεχίζει κάνοντας αναδρομική κλήση δύο φορές. Η μία κλήση είναι για τον κώδικα που είναι εσωτερικός στο tag και η άλλη για την ουρά της αρχικής λίστας. Η κλήση για την ουρά γίνεται για να πάρει το κείμενο που εσωκλείεται στα tags. Τέλος συνενώνει όλα τα κείμενα σε μια λίστα και την επιστρέφει. Η τερματική συνθήκη ενεργοποιήται όταν δεν υπάρχουν άλλα στοιχεία στη λίστα και επιστρέφει κενή λίστα.

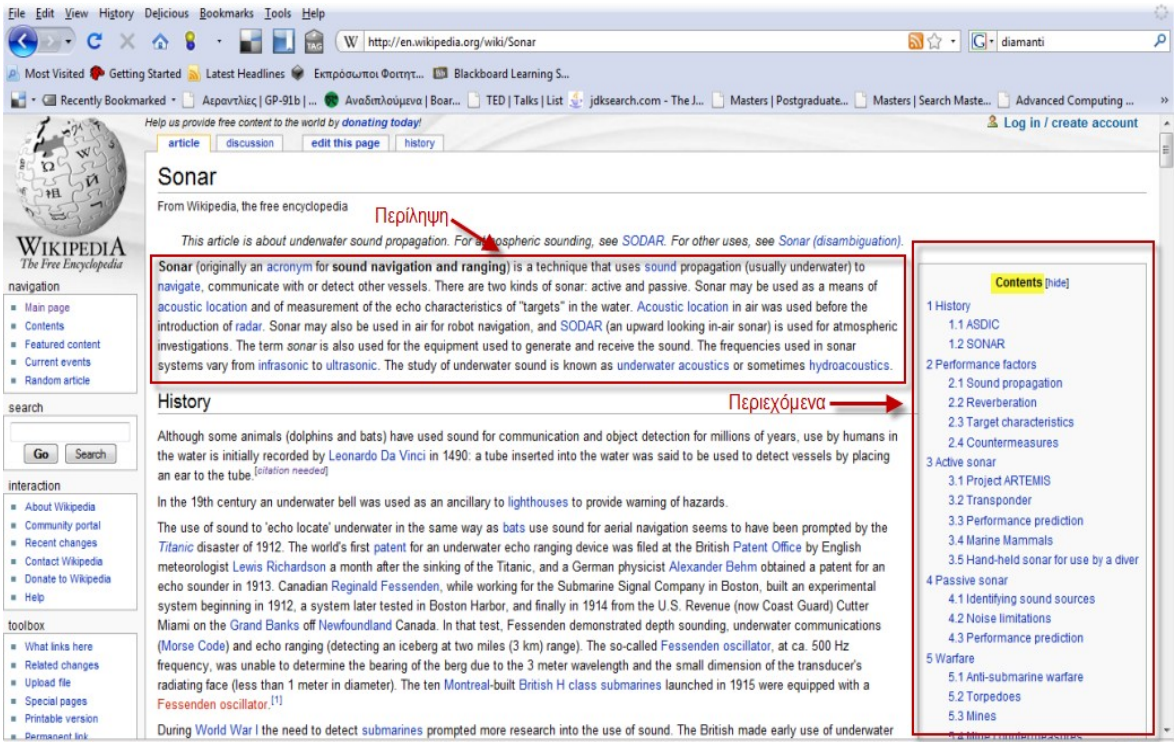
Το ***get_tags*** αφού δεχθεί τη λίστα από το ***remove_inner_tags,*** την ενώνει με τη λίστα που του επιστρέφεται από την αναδρομική κλήση και έτσι τελειώνει η ανάκτηση του κατάλληλου κειμένου.

3.5 Ειδικές Περιπτώσεις

Κατά την αντιμετώπιση αυτών των προβλημάτων υπήρξαν και κάποιες ειδικές περιπτώσεις που έπρεπε να ληφθούν υπόψη για τη σωστή λειτουργία του προγράμματος. Τέτοιου είδους πρόβλημα ήταν η μη ύπαρξη σύνδεσης στο διαδίκτυο. Αυτό το πρόβλημα αντιμετωπίστηκε τροποποιώντας από τη βιβλιοθήκη **http** της Swi-Prolog το module **http_open**. Πιο συγκεκριμένα η τροποποίηση έγινε στο κατηγόρημα ***open_socket/4***, προσθέτοντας το ***catch*** και κλήση της ***tcp_socket*** και σε περίπτωση που υπάρξει κάποιο *throw* τότε εμφανίζεται μήνυμα *"You have no Internet Connection"* και κάνει ***abort*** το πρόγραμμα.

Μια άλλη αλλαγή που έγινε πάλι στο **http_open**, στο κατηγόρημα ***http_open.*** Χρησιμοποιήθηκε πάλι το ***catch*** για τη περίπτωση που δε βρεθεί η σελίδα του όρου αναζήτησης. Αν δε βρεθεί επιστρέφει ένα *stream = ' '*, για να μην υπάρχει *exception* την ώρα της εκτέλεσης και σταματάει το πρόγραμμα. Εάν στο open_pages αναγνωριστεί stream ' ' στο ***http_open*** τότε εμφανίζεται το μήνυμα *"Page όρος does not exist"*.

Ειδικές περιπτώσεις συναντήθηκαν και κατά την επιλογή του κατάλληλου κειμένου. Όπως αναφέρθηκε πρωτύτερα η ανάκτηση γίνεται στο κείμενο που είναι (εμφανισιακά) πριν από τα περιεχόμενα. Σε μερικές περιπτώσεις, όπως φαίνεται και στην Εικόνα 10, τα περιεχόμενα βρίσκονται πλάγια από το κείμενο. Σε αυτές τις περιπτώσεις το κείμενο έχει παρατηρηθεί πως είναι πριν από την πρώτη κεφαλίδα μορφής h2. Οπότε η τερματική συνθήκη του ***get_tags*** θα είναι είναι εάν βρεθεί ***element*** με *tag h2*. Σε μερικές άλλες περιπτώσεις δεν υπάρχει περίληψη. Σε αυτές τις περιπτώσεις το κείμενο είναι ελάχιστο και κρίθηκε σκόπιμο να ανακτηθεί όλο το κείμενο. Οπότε η τερματική συνθήκη θα είναι η κενή λίστα.



Εικόνα 10. Θέση περιεχομένων πλάγια από τη περίληψη

Ένα τελευταίο πρόβλημα που αναδείχθηκε είναι να μην μπορούν να αναπαρασταθούν κάποιοι χαρακτήρες που βρίσκονται στο κείμενο της σελίδας. Ο έλεγχος για την ύπαρξη τέτοιων χαρακτήρων γίνεται μέσω του module ***dia_pretty_print***, αλλάζοντας το κατηγόρημα pp/2 βάζοντας ***catch*** και καλώντας το κατηγόρημα ***writeq*** και σε περίπτωση που βγάλει *error* εμφανίζει το μήνυμα *“This page contains characters that cannot be represented”*.

3.6 Με λίγα λόγια

Όπως πολύ πιθανόν να έγινε κατανοητό οι ενέργειες που γίνονται για την σύνδεση του προγράμματος στο Παγκόσμιο Ιστό. Η αναζήτηση και η εύρεση των σελίδων στον ιστότοπο της Wikipedia καθώς και η ανάκτηση των κειμένων που υπάρχουν σε αυτές τις σελίδες και έχουν περιγραφεί σε αυτό το κεφάλαιο είναι το πιο σημαντικό κομμάτι, από πλευράς κώδικα, για να γίνει δυνατή η δημιουργία του κειμένου για την εισαγωγή του όρου στο γλωσσάρι. Για να επιτευχθούν όλα αυτά, όπως φάνηκε και στο κείμενο, εκτός από τη μελέτη της δομής της Wikipedia χρειάστηκε και να κατασκευαστούν και αρκετά modules. Αυτή η κατασκευή έγινε για γίνει η επεξεργασία και η μορφοποίηση είτε των κειμένων ή του κώδικα που σχετίζονται με τη σελίδα της Wikipedia είτε των όρων που δίνονται από το χρήστη για αναζήτηση για να φθάσουν να είναι συμβατά με τη δομή της Wikipedia. Στο κεφάλαιο που ακολουθεί περιγράφονται πάλι ενέργειες ανάκτησης/εξόρυξης δεδομένων οι οποίες μοιάζουν με τις ενέργειες που γίνονται σε αυτό το κεφάλαιο αλλά επειδή τα δεδομένα που ανακτούνται είναι σύνδεσμοι και έχουν διαφορετική μορφοποίηση από το κείμενο των σελίδων έπρεπε να γίνει κατασκευή διαφορετικών module για την επεξεργασία τους.

Κεφάλαιο 4

Εξόρυξη Δεδομένων στο Πρόγραμμα

Η εξόρυξη δεδομένων στη παρούσα εργασία θεωρείται η εξόρυξη των συνδέσμων που υπάρχουν στη σελίδα ενός όρου της Wikipedia και είναι σχετικοί με τον όρο. Σχετικοί με τον όρο θεωρήθηκαν οι σύνδεσμοι που βρίσκονται στο κείμενο της περίληψης που ανακτάται και θα εμφανίζεται στο γλωσσάρι, οι σύνδεσμοι που βρίσκονται στα μεταδεδομένα μιας σελίδας και τέλος οι σύνδεσμοι που είναι στο infobox (πίνακα πληροφοριών) ενός όρου. Η εξόρυξη αυτών των συνδέσμων θεωρήθηκε χρήσιμη γιατί μέσω αυτών μπορεί ο χρήστης να ενημερωθεί με έννοιες/όρους που σχετίζονται με τους όρους αναζήτησης που έδωσε. Επίσης αυτό έγινε για να υπάρχει από το πρόγραμμα μια επιπλέον δυνατότητα έτσι ώστε να μπορεί να κάνει μία αναζήτηση σε βάθος. Αναζήτηση σε βάθος είναι η αναζήτηση κάποιων επιπλέον όρων, που δίνει ο χρήστης μαζί με τον κύριο όρο, και υπάρχει η πιθανότητα, αυτοί οι όροι, να είναι εσωτερικοί ή σχετικοί όροι με τον πρωτεύον όρο. Μετά εάν όντος υπάρχει τέτοιος σύνδεσμος μέσα στο κείμενο, το πρόγραμμα θα ανακτήσει και το δικό του κείμενο και θα το εμφανίσει σαν εσωτερικό του αρχικού.

4.1 Ανάκτηση συνδέσμων κειμένου

Για να γίνει η ανάκτηση των συνδέσμων του κειμένου κατασκευάστηκε ένας καινούργιος κανόνας το ***get_a_tags/2*** το οποίο στη λειτουργία του είναι παρόμοιο με αυτό του ***get_tags***. Η ανάκτηση των συνδέσμων ξεκινάει όταν κληθεί ο κανόνας ***get_tags*** και βρεί *p tag* για να κάνει την ανάκτηση του κειμένου. Όταν λοιπόν βρεθεί στην αναδρομική κλήση ένα ***element*** με *p tag,* τότε η πρώτη κλήση που κάνει το κατηγόρημα ***get_tag*** είναι το ***get_a_tags***. Το ***get_a_tags(+ListA,-ListB)***, λειτουργεί και αυτό αναδρομικά όπως και ο μεγαλύτερος αριθμός των

κατηγορημάτων, δέχεται σαν όρισμα τον εσωτερικό κώδικα του *p tag* που έχει βρεθεί. Εφόσον πάρει τον κώδικα (πάντα σαν λίστα ***element***) κάνει έλεγχο των *tag* και αυτό όπως και το *get_tags*, αλλά ο έλεγχος εδώ γίνεται για τη περίπτωση που βρει *tag_a*, δηλαδή σύνδεσμο. Σε περίπτωση που δε βρεθεί, τότε γίνεται αναδρομή και στην εσωτερική λίστα του ***element*** αλλά και στην ουρά της κανονικής λίστας. Σε περίπτωση που βρεθεί a tag, καλείται ο κανόνας ***get_title/2***, δίνοντας σαν όρισμα τη λίστα Identifiers του ***element*** και γίνεται αναδρομή στην ουρά της λίστας. Η τερματική συνθήκη της αναδρομής είναι εάν βρει κενή λίστα να επιστρέψει κενή λίστα.

Το *get_title(+ListA,-ListA)* είναι ένας κανόνας που δέχεται σαν είσοδο τη λίστα με τα Identifiers του ***element*** και με αναδρομική κλήση παίρνει το κείμενο του πεδίου *title*. Αυτό γίνεται γιατί παρατηρήθηκε πως το *title* τις περισσότερες φορές περιλαμβάνει το σωστό κείμενο για το άνοιγμα του συνδέσμου, ενώ το κείμενο που αναγράφεται πολλές φορές δεν αντιστοιχείται σωστά.

Εφόσον επιστραφεί η λίστα με τα *titles* από το ***get_titles*** στο ***get_a_tags***, επιστρέφονται και από το ***get_a_tags*** στο ***get_tags***. Έπειτα αυτή η λίστα περνάει από ένα φιλτράρισμα. Το φιλτράρισμα αυτό γίνεται μέσω του κανόνα ***filter/2*** που ανήκει στο module **filtering**. Το ***filter(+ListA,-ListB)*** κάνει “επίπεδη” τη λίστα, δηλαδή χωρίς εσωτερικές λίστες, μέσω του κατηγορήματος ***flat/2***. Μετά αφαιρεί τις κενές λίστες κάνοντας κλήση το κανόνα ***remove_empty_lists/2***. Τέλος με τη χρήση του κανόνα ***add_words/2*** ελέγχει αν η κεφαλή της λίστας είναι *atom* και αν είναι τότε τη διατηρεί έτσι, ενώ αν δεν είναι γίνεται αναδρομική κλήση ώστε να προστεθούν και τα εσωτερικά *atoms*. (περισσότερες πληροφορίες για το module filtering στο Κεφάλαιο 5). Εφόσον επιστραφούν οι σύνδεσμοι μέσω του ***get_tags*** στο ***open_pages*** δέχονται τη τελευταία τροποποίηση. Με τη χρήση του κατηγορήματος ***remove_empty_elements/2*** αφαιρεί τα κενά στοιχεία που υπάρχουν στη λίστα.

4.2 Ανάκτηση μεταδεδομένων σελίδας

Όπως φαίνεται στην παρακάτω εικόνα (Εικόνα 11) η Wikipedia στον html κώδικά της περιέχει και μεταδεδομένα. Αυτά τα μεταδεδομένα υπάρχουν για να γίνεται καλύτερα η περιγραφή της σελίδας, ειδικότερα σε μηχανές αναζήτησης στο διαδίκτυο. Στη συγκεκριμένη περίπτωση η Wikipedia έχει μεταδεδομένα για να δώσει σχετικούς όρους με τον όρο που εμφανίζεται στη σελίδα. Το όνομα αυτών τον μεταδεδομένων είναι *keywords*.

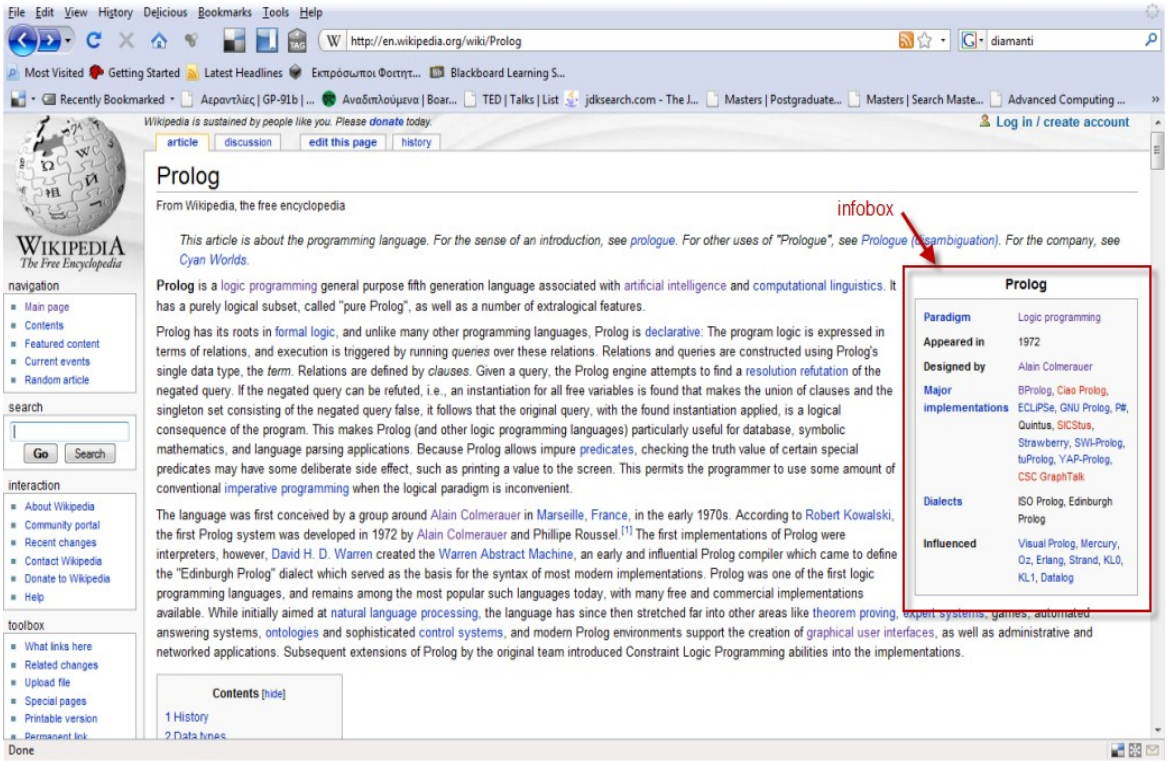
Εικόνα 11. Μεταδεδομένα στον html κώδικα της Wikipedia

Για να γίνει η απόσπαση αυτών των δεδομένων γίνεται χρήση, από το module **open_pages** και πιο συγκεκριμένα από το κατηγόρημα ***open_pages***, του module **get_see_also**. Από αυτό το module χρησιμοποιείται ο κανόνας ***get_meta/2***. Αυτό το κατηγόρημα δέχεται σαν είσοδο μια λίστα με τον html κώδικα κατάλληλα διαμορφωμένο και κάνει έλεγχο για να βρει τα ***elements*** εκείνα που έχουν *tag ίσο με meta*. Εάν βρεθούν *meta tags* τότε γίνεται κλήση του κανόνα ***get_name/2*** για να βρεθεί το *meta* με όνομα *keywords* που, όπως αναφέρθηκε παραπάνω, περιέχει τα δεδομένα που περιγράφουν κατάλληλα τον όρο. Αφού γίνει και αυτός ο έλεγχος τότε το μόνο που απομένει είναι, μέσω του κατηγορήματος ***get_content/2***, να επιστραφεί το κείμενο που περιέχεται μέσα στο ***element*** *meta* με το όνομά *keywords*. Αυτό το κείμενο επιστρέφεται και γίνεται στη τελική του μορφή, από τη ***get_meta***, η οποία είναι λίστα από κείμενα και αυτό γυρίζει στο κατηγόρημα ***open_pages***. Το ***open_pages*** κάνει κάποιες τελευταίες τροποποιήσεις σε αυτή τη

λίστα. Κάνει το ***filter*** που αναφέρθηκε και για τους συνδέσμους του κειμένου και επίσης κάνει και απομάκρυνση των κενών στοιχείων μέσω της ***remove_empty_elements/2***.

4.3 Ανάκτηση συνδέσμων από Infobox

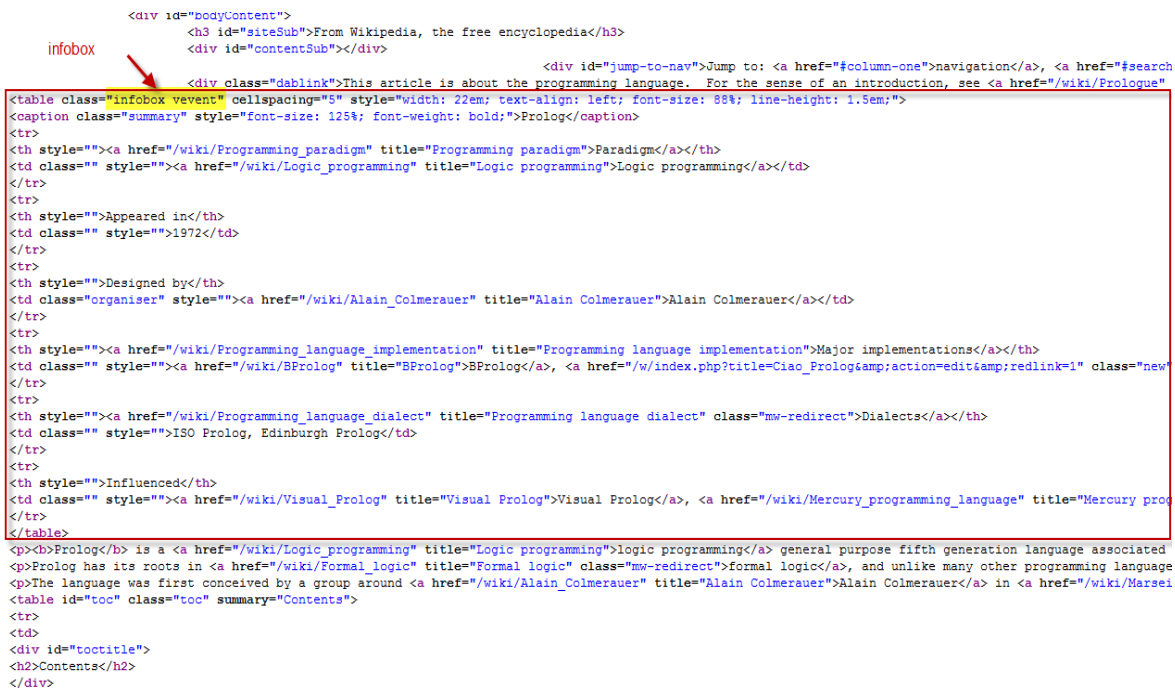
Το infobox (πίνακας πληροφοριών) είναι ένας πίνακας που περιέχει κάποια σημαντικά στοιχεία για τον όρο τον οποίο έχει γίνει η αναζήτηση. Δε περιέχουν όλες οι σελίδες infobox αλλά επειδή οι πληροφορίες, σε όσες σελίδες υπάρχει infobox, θεωρούνται σημαντικές κατασκευάστηκε το module **get_info_box**. Ένα infobox φαίνεται στην Εικόνα 12 που βρίσκεται στην επόμενη σελίδα.



Εικόνα 12. Infobox

Η ενέργεια, για να ξεκινήσει η ανάκτηση των δεδομένων του infobox, γίνεται και αυτή από το κατηγόρημα ***open_pages*** κάνοντας κλήση του κανόνα ***get_table_tags/2***, που όπως είπαμε ανήκει στο module **get_info_box**. Ο κανόνας ***get_table_tags(+ListA,-ListB)*** δέχεται μια λίστα με τον μορφοποιημένο html κώδικα. Επειδή στο html κώδικα της Wikipedia, όπως φαίνεται και στην Εικόνα 13,

έχει το Identifier *class*, του *table*, ίσο με *infoboxvevent*, γίνεται αναζήτηση των ***element*** *table*. Αφού βρεθεί κάποιο ***element*** που να είναι ίσο με *table*, καλείται ο κανόνας ***get_class/3*** με όρισμα τη λίστα των Identifiers και τη λίστα του εσωτερικού κώδικα του *table*. Το ***get_class*** κάνει αναζήτηση να βρει το Indentifier *class*. Όταν το βρει καλεί το κανόνα ***get_info/3*** δίνοντας του τη συμβολοσειρά που αντιστοιχεί στο *class* καθώς και τον εσωτερικό κώδικα του *table*. Το get_info, αφού μετατρέψει τη συμβολοσειρά σε ένα πίνακα με τους ASCII κωδικούς που αντιστοιχούν στο κάθε χαρακτήρα της συμβολοσειράς, μέσω του κατηγορήματος ***name*** της Prolog, ελέγχει αν οι πρώτοι αριθμοί της λίστας είναι 105, 110, 102, 111, 98, 111, 120 που αντιστοιχούν σε infobox. Τότε μέσω του ***get_a_tags***, που αναφέρθηκε σε προηγούμενο υποκεφάλαιο, γίνεται ανάκτηση των συνδέσμων και επιστρέφονται οι σύνδεσμοι. Στις διαφορετικές περιπτώσεις κάθε κανόνα απλώς γίνεται αναδρομή μέχρις ότου βρεθεί το ζητούμενο ή μέχρι να φθάσουν στις συνθήκες τερματισμού. Η συνθήκη τερματισμού για το ***get_table_tags*** είναι να βρεθεί κενή λίστα, και να επιστρέφει κενή λίστα. Για το ***get_class*** είναι πάλι η κενή λίστα και επιστρέφει μία κενή συμβολοσειρά και τέλος για το ***get_info*** είναι η εύρεση του infobox γιατί δεν υπάρχει δεύτερο σε καμιά σελίδα.



Εικόνα 13. Html κώδικας ενός infobox

4.4 Αναζήτηση εσωτερικών όρων

Όπως αναφέρθηκε προηγουμένως, ο χρήστης έχει τη δυνατότητα να κάνει μια αναζήτηση με βάθος. Αυτή την αναζήτηση έχουμε ορίσει να γίνεται μέσω των συνδέσμων που είναι εσωτερικοί στο κείμενο που ανακτήθηκε και όχι στο σύνολο των συνδέσμων που έχουν βρεθεί.

Για να γίνει αυτή η επιπλέον αναζήτηση στην αρχή γίνεται έλεγχος εάν οι επιπλέον όροι που έχει δώσει ο χρήστης υπάρχουν στους εσωτερικούς συνδέσμους του κειμένου. Αυτόν τον έλεγχο τον αναλαμβάνει ο κανόνας ***check_compability/3*** του module **check_compability**. Ο κανόνας αυτός δέχεται δύο τροποποιημένες λίστες (η τροποποίηση που έχει γίνει είναι να γίνουν οι όροι σε πεζά μέσω του ***downcase_atom/2*** και του ***make_it_lowercase/2***) οι οποίες είναι μία οι λίστα με τους όρους για αναζήτηση σε βάθος, που έδωσε ο χρήστης και η άλλη είναι η λίστα με τους συνδέσμους που έχουν ανακτηθεί. Εφόσον βρει ίδιο όρο τον προσθέτει στη λίστα με τους υπόλοιπους όρους που υπάρχουν. Έπειτα επιστρέφει αυτή τη λίστα στο ***open_pages/4*** και καλείται το ***open_pages/3*** το οποίο κάνει ακριβώς την ίδια δουλειά με το ***open_pages/4*** απλώς όμως δε κάνει την επιπλέον αναζήτηση για τους εσωτερικούς όρους του κειμένου του.

4.5 Με λίγα λόγια

Σε αυτό το κεφάλαιο αναφέρθηκαν οι τρόποι που γίνεται η εξόρυξη των δεδομένων, δηλαδή η ανάκτηση των συνδέσμων, καθώς και τα κριτήρια με τα οποία επιλέγονται οι σύνδεσμοι και επίσης πως επιλέγεται το περιεχόμενο το οποίο θα φαίνεται στο χρήστη. Περιγράφονται τα modules και οι λειτουργίες τους, όπως και όλα τα κατηγορήματα που χρησιμοποιούνται, για να γίνει η ανάκτηση των συνδέσμων. Όπως είπαμε γίνεται εκτενής αναφορά στα κριτήρια με τα οποία γίνεται επιλογή για ανάκτηση συνδέσμων αλλά αναφέρεται και αναλυτικά ο τρόπος υλοποίησης των κατηγορημάτων τα οποία χρησιμοποιούνται στη κάθε περίπτωση. Τέλος γίνεται η περιγραφή και ο τρόπος υλοποίησης μιας επιπλέον λειτουργίας την

οποία δίνει το πρόγραμμα, της εύρεσης όρου δεύτερου επιπέδου. Αυτή η λειτουργία εν συντομία είναι εύρεση όρων που δίνει ο χρήστης που υπάρχουν σαν σύνδεσμοι στους πρωτεύοντες όρους για να γίνει η περιγραφή τους και η εισαγωγή τους στο γλωσσάρι. Στο επόμενο κεφάλαιο περιγράφονται οι ενέργειες οι οποίες γίνονται για την επεξεργασία του κειμένου.

Κεφάλαιο 5

Επεξεργασία κειμένου

Η επεξεργασία κειμένου είναι ένα από τα πιο σημαντικά σημεία κατά τη διάρκεια εκτέλεσης του προγράμματος. Είναι σημαντική γιατί μέσω αυτής μπορεί το πρόγραμμα να αποφύγει κάποιες ιδιαιτερότητες αναφορικά με ειδικούς χαρακτήρες και τελεστές που έχουν να κάνουν με την υλοποίηση της Prolog. Επίσης βοηθάει στη δημιουργία προτάσεων, λέξεων χρησιμοποιώντας κάποιο κατάλληλο διαχωριστή, που μπορεί και να μην είναι η τελεία. Μπορεί να χωρίζει κείμενο για να γίνει ορθότερη επεξεργασία, να ελέγχει κατά πόσο είναι όμοια λίστες κειμένων και *atoms* και τέλος στην επεξεργασία θεωρούμε και τη καταγραφή του κειμένου σε αρχεία κειμένου, με κατάλληλη μορφοποίηση.

5.1 Παράβλεψη ειδικών χαρακτήρων – Ορθότερη επεξεργασία

Αρχίζοντας από την αρχή, το πρώτο σημείο της επεξεργασίας κειμένου είναι στην εισαγωγή των ορισμάτων. Για να μπορέσει το πρόγραμμα να αποφύγει ανεπιθύμητα σφάλματα και για να υπάρχει και ένα καλύτερο interface κατασκευάστηκε κατ' αρχάς το module **reading,** του οποίου η λειτουργία έχει περιγραφεί στο κεφάλαιο 3. Όπως αναφέρεται πάλι στο ίδιο κεφάλαιο, για την ορθότερη επεξεργασία και για να είναι φιλικότερη προς το χρήστη η αναζήτηση, το interface έγινε ένα module, το **reading2,** έτσι ώστε να δίνει ο χρήστης τους όρους αναζήτησης σε μια συγκεκριμένη μορφή. Μετά το πρόγραμμα καταλαβαίνει τη μορφή και ποιοι είναι οι όροι αναζήτησης πρώτου και δεύτερου βαθμού. Επίσης σε αυτή τη κατηγορία επεξεργασίας κειμένου εντάσσεται και η επιπλέον επεξεργασία που έγινε στα ορίσματα ούτως ώστε να γίνει αλλαγή του κενού χαρακτήρα ' ' με κάτω παύλα '_'.

5.2 Έλεγχος διπλότυπων

Ο έλεγχος των διπλότυπων γίνεται σε δύο φάσεις, μία κατά την οποία στην ουσία είναι εύρεση ομοίων ορισμάτων μίας λίστας σε μία άλλη και μία κατά την οποία γίνεται πραγματικός έλεγχος σε μία λίστα και βρίσκονται τα διπλότυπα κείμενα τα οποία εν συνεχεία αφαιρούνται. Η πρώτη περίπτωση έχει αναλυθεί προηγουμένως και είναι η περίπτωση του ***check_compatibility***, όπου γίνεται έλεγχος εάν υπάρχει ο σύνδεσμος, για αναζήτηση δεύτερου επιπέδου, στο κείμενο που ανακτήθηκε.

Στη δεύτερη περίπτωση, για να μπορεί το πρόγραμμα να αναγνωρίζει σωστά τα διπλότυπα και να γίνεται η αφαίρεσή τους δημιουργήθηκε το module **filtering**. Για να χρησιμοποιηθεί το module filtering πρέπει να κληθεί το κατηγόρημα ***filter/2***. Αυτή η κλήση γίνεται από το κατηγόρημα ***open_pages/4,*** κάθε φορά που γίνεται επιστροφή από κάποιο κατηγόρημα μίας λίστας με συνδέσμους(links). Η δουλειά του ***filter(+ListA,-ListB)*** είναι απλή, δέχεται σαν όρισμα εισόδου τη ListA, η οποία είναι και η λίστα από την οποία θέλουμε να γίνει η αφαίρεση αυτών των διπλότυπων εγραφών. Περνάει αυτή τη λίστα σαν είσοδο στο κατηγόρημα ***flat/2***. Το ***flat(+ListA,-ListB)*** είναι ένα γνωστό κατηγόρημα το οποίο κατασκευάζεται για να γίνεται “σβήσιμο” των εσωτερικών εμφολεύσεων, έτσι ώστε η λίστα να έχει μόνο ένα επίπεδο. Στην ουσία όπως λέει και το όνομά του στα αγγλικά κάνει τη λίστα “επίπεδη”. Για να το κάνει αυτό το κατηγόρημα ***flat*** είναι αναδρομικό και “ψάχνει” να βρει στην εκάστοτε κεφαλή της λίστας στοιχεία που δεν είναι *atomic*. Εάν βρεί στοιχείο που δεν είναι *atomic* καλεί τον εαυτό της για την κεφαλή της λίστας αλλά και για την ουρά και μετά τα αποτελέσματα τα ενώνει και τα επιστρέφει σαν μια καινούργια λίστα. Αν πάλι βρει στοιχείο που να είναι *atomic,* τότε το κάνει αναδρομική κλήση, μόνο για την ουρά της λίστας, επιστρέφει το αποτέλεσμα αυτό σαν μια νέα λίστα και τοποθετεί σαν κεφαλή της καινούργιας λίστας την ίδια κεφαλή. Τερματική συνθήκη είναι η εύρεση κενής λίστας κατά την οποία επιστρέφει μία κενή λίστα. Στη συνέχεια, και αφού έχουν αφαιρεθεί τα εσωτερικά επίπεδα, γίνεται αφαίρεση από τη κυρίως λίστα, των κενών λιστών που μπορεί να έχουν δημιουργηθεί από τη χρήση του ***flat.*** Αυτό μπορεί να γίνει επειδή το ***flat*** έχει στην

τερματική λίστα επιστροφή κενής λίστας και μπορεί να ενεργοποιηθεί πάνω από μία φορές. Οπότε αυτές οι κενές λίστες παραμένουν σαν στοιχεία. Η αφαίρεση που αναφέραμε, γίνεται μέσω του κανόνα ***remove_empty_lists(+ListA,-ListB)*** που ελέγχει εάν είναι η κεφαλή της λίστας κενή λίστα και την αφαιρεί, αλλιώς κάνει αναδρομική κλήση. Κατά την τερματική συνθήκη ελέγχει εάν δεν υπάρχει άλλη κενή λίστα επιστρέφει τη λίστα ή τις λίστες που έχουν απομείνει. Ο έλεγχος για την ύπαρξη της κενής λίστας γίνεται μέσω του κατηγορήματος της Prolog ***member/2***. Τέλος, εφόσον έχουν αφαιρεθεί και οι κενές λίστες, γίνεται η κλήση του κανόνα ***add_words/2***. Ο κανόνας ***add_words(+ListA,-ListB)*** κάνει έναν τελευταίο διαχωρισμό έτσι ώστε οι λέξεις που δημιουργούνται να είναι κατάλληλες για την επιπλέον αναζήτηση. Το ***add_words*** παίρνει τη κεφαλή της ListA και ελέγχει αν είναι *atom*. Εάν είναι συνεχίζει κανονικά την αναδρομή, καλώντας μόνο την ουρά και μετά τη λίστα που επιστρέφεται την ενώνει με τη κεφαλή και την επιστρέφει. Εάν δεν είναι *atom* τότε κάνει αναδρομή και για τη κεφαλή και για την ουρά και μετά τις λίστες που δέχεται τις συνενώνει και τις επιστρέφει σαν ListB. Αυτή τη ListB επιστρέφει και το κατηγόρημα ***filter*** στο ***open_pages***.

5.3 Δημιουργία προτάσεων-λέξεων

Όπως και στη προηγούμενη ενότητα έτσι και εδώ η δημιουργία προτάσεων και λέξεων χωρίζεται σε δύο ξεχωριστές μεταξύ τους ενότητες, η μία είναι για να ενώνει το κείμενο που ανακτάται από τα εκάστοτε ***tags*** ενώ η άλλη είναι για να διαχωρίζει λέξεις μέσα σε ένα κείμενο. Στη πρώτη περίπτωση δε κατασκευάστηκε κάποιο ειδικό module για να γίνει αυτή η επεξεργασία, αλλά χρησιμοποιήθηκε το κατηγόρημα της Prolog ***concat***. Οπότε κάθε φορά που γινόταν ανάκτηση κάποιου κειμένου τότε το κείμενο αυτό ενώνονταν μέσω του concat με το ήδη υπάρχον κείμενο.

Στη δεύτερη περίπτωση για το διαχωρισμό των λέξεων έχει κατασκευαστεί ένα module. Το module αυτό είναι **make_words**, από το οποίο χρησιμοποιούμε το κανόνα ***words/2***. Το ***make_words*** είναι ένα module το οποίο μετατρέπει

συμβολοσειρές σε λέξεις χρησιμοποιώντας ως διαχωριστή το κόμμα. Θα μπορούσε εύκολα αντί για το κόμμα να χρησιμοποιεί ως διαχωριστή τη τελεία έτσι ώστε να κάνει διαχωρισμό προτάσεων. Το ***words(+ListA,-ListB)*** καλείται από το ***open_pages*** και αφού έχει γίνει το φιλτράρισμα των ορισμάτων αναζήτησης, για να διαχωρίσει τους όρους μεταξύ τους, ούτως ώστε να μπορεί να γίνεται η αναδρομική κλήση σωστά για τον κάθε όρο. Η λειτουργία του words είναι η εξής: δέχεται σαν όρισμα εισόδου τη ListA, η οποία περιέχει συμβολοσειρές. Σε κάθε κύκλο αναδρομής η ListA περνάει σαν όρισμα εισόδου στο κατηγόρημα ***tokenize/2,*** για να γίνει η επεξεργασία όλων των συμβολοσειρών και μετά αφού γίνει αυτή η επεξεργασία αφαιρούνται οι κενές λίστες με τη χρήση του κατηγορήματος ***remove_empty_lists/2*** του module **filtering**, τα οποία περιγράφηκαν λεπτομερώς στη προηγούμενη υποενότητα.

Το ***tokenize(+ListA,-ListB)*** σε κάθε κύκλο αναδρομής περνάει τη κεφαλή της ListA σαν όρισμα εισόδου σε ένα άλλο κανόνα ***make_words/2***, το οποίο είναι υπεύθυνο για την επεξεργασία και το διαχωρισμό των string. Στη συνέχεια κάνει αναδρομική κλήση για την ουρά και τέλος καλεί τη flat για τις λίστες που του επεστράφησαν από τις κλήσεις των προηγούμενων κανόνων και το αποτέλεσμα το επιστρέφει σαν ListB.

Ο κανόνας ***make_words(+StringA,-StringB)*** κάνει το εκάστοτε StringA, σε κάθε κλήση του, μια λίστα από κωδικούς ASCII μέσω του κατηγορήματος της Prolog ***sting_to_list/2.*** Αυτό το κατηγόρημα μετατρέπει τον κάθε χαρακτήρα του string στο αντίστοιχο του ASCII κωδικό και μετά αυτόν το κωδικό το βάζει σε μια λίστα. Στη συνέχεια, και αφού έχει γίνει η μετατροπή, η λίστα με τους κωδικούς που έχει παραχθεί δίνεται σαν όρισμα εισόδου στο κανόνα ***split_list/2*** το οποίο είναι υπεύθυνο για το διαχωρισμό. Το ***split_list(+ListA,-ListB)*** κάνει έλεγχο, μέσω του ενσωματομένου κατηγορήματος της Prolog ***member/3***, να βρει εάν υπάρχουν στοιχεία 44 (ASCII κωδικός του κόμματος) μέσα στη λίστα. Εάν δεν υπάρχει αυτό το στοιχείο τότε είναι η τερματική συνθήκη και επιστρέφεται το κομμάτι της λίστας που έχει απομείνει. Εάν είναι μέλος της λίστας το 44, τότε χρησιμοποιείται το

ενσωματωμένο κατηγόρημα της Prolog ***append/3***, με την αντίθετη χρήση που συνήθως χρησιμοποιείται, δηλαδή συνήθως η χρήση του είναι για να γίνεται ένωση λιστών αλλά σε αυτή τη περίπτωση χρησιμοποιείται η “εξυπνάδα” της Prolog και κάνουμε διαχωρισμό. Στο διαχωρισμό παίρνουμε μόνο το πρώτο αποτέλεσμα που επιστρέφει και στη συνέχεια κάνουμε αναδρομική κλήση. Αυτό που επιστρέφεται, αφού προστεθεί σαν κεφαλή η λίστα που έχουμε από το διαχωρισμό, τότε επιστρέφεται και αυτή η λίστα. Τελευταία ενέργεια του ***make_words*** είναι, αφού αφαιρέσει της κενές λίστες με το γνωστό πλέον κατηγόρημα ***remove_empty_lists/2***, να ξανακάνει τις λίστες από κωδικούς ASCII σε συμβολοσειρά και να επιστρέψει τη λίστα. Αυτή η μετατροπή όμως δε μπορεί να γίνει μέσω του κανόνα ***string_to_list*** γιατί πλέον δεν υπάρχει μία λίστα από κωδικούς αλλά μία λίστα από λίστες κωδικών. Όπως μπορεί να γίνει αντιληπτό για αυτή τη δουλειά κατασκευάστηκε ένας κανόνας το ***convert_words/2*** το οποίο δέχεται σαν όρισμα μία λίστα και παίρνει κάθε φορά τη κεφαλή της τη μετατρέπει σε string μέσω του κατηγορήματος ***name/2***, μετά κάνει αναδρομική κλήση και βάζει σα κεφαλή της λίστας που του επιστράφηκε από την αναδρομική κλήση τη λέξη που έχει μετατρέψει και μετά επιστρέφει αυτή τη λίστα. Αυτή η διαδικασία τελειώνει όταν βρεθεί κενή λίστα τότε επιστρέφεται η κενή λίστα.

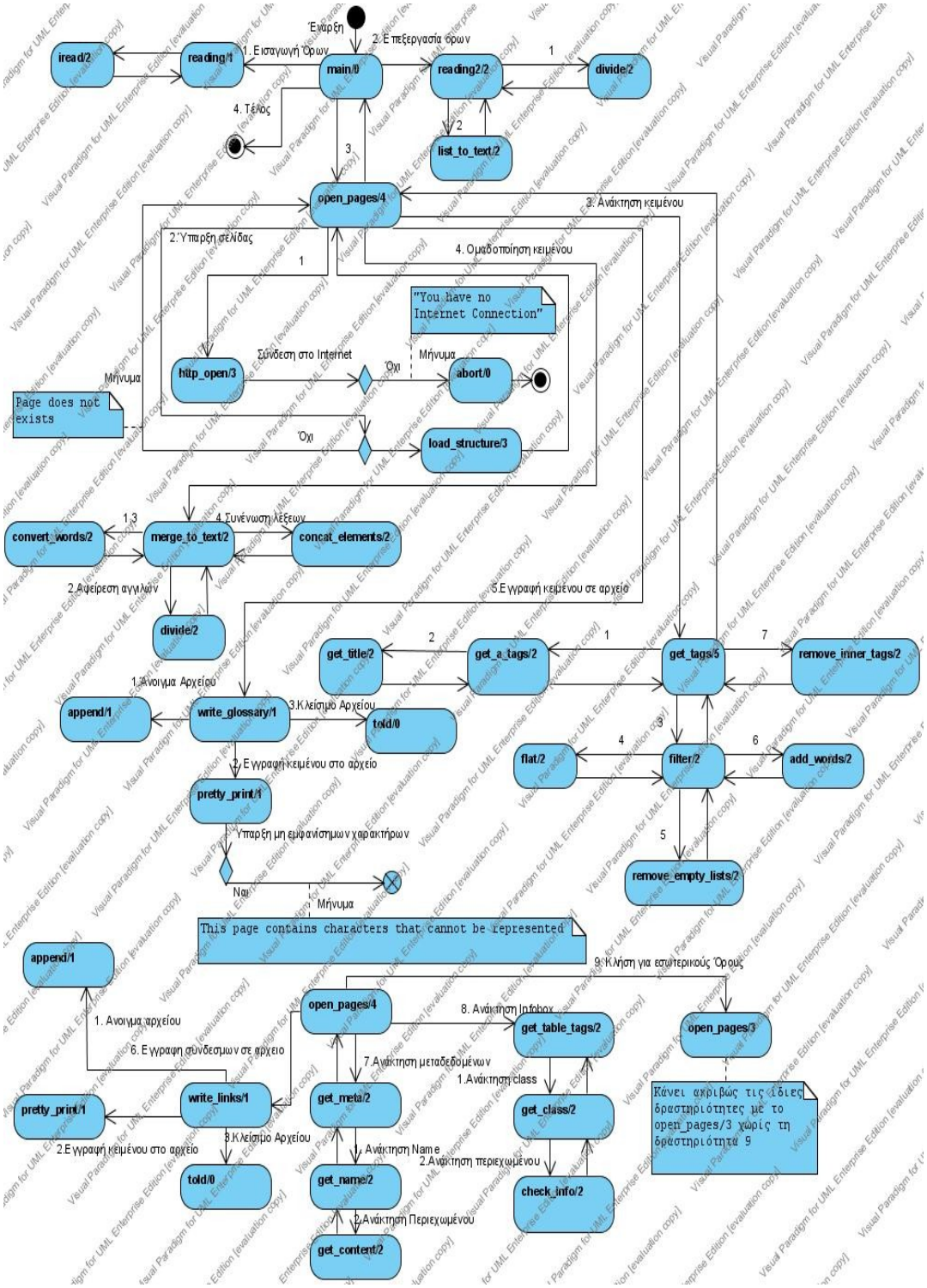
5.4 Εγγραφή σε αρχείο κειμένου

Οι εγγραφές σε αρχείο κειμένου είναι μια πολύ απλή διαδικασία στη Prolog. Στο συγκεκριμένο πρόγραμμα δημιουργούνται δύο αρχεία ένα για να περιέχει τη περίληψη των όρων το *glossary.txt* και ένα για να έχει τους συνδέσμους που έχουν ανακτηθεί, το *links.txt*. Για να μπορέσει να γίνει ακόμα πιο εύκολη η διαδικασία και να γλιτώσουμε γραμμές κώδικα κατασκευάστηκε και ένα module το **writing**.

Η εγγραφή σε αρχείο στη Prolog γίνεται ως εξής: εάν θέλει ο χρήστης να σβηστεί ότι υπάρχει από πριν στο κείμενο τότε ξεκινάει γράφοντας ***tell('onoma_arxeiou')***, έτσι μπορεί και να δημιουργηθεί και ένα νέο κενό αρχείο. Εάν πάλι θέλει να προσθέσει κάτι στο τέλος του αρχείου τότε γράφει ***append('onoma_arxeiou')***. Στη

συνέχεια γράφει αυτό που θέλει η έξοδος τους να πάνε στο αρχείο όπως π.χ. μια εντολή ***write*** και εφόσον έχει τελειώσει με όλες τις ενέργειες που θέλει να κάνει στο αρχείο τότε γράφει ***told.***

Στο παρόν πρόγραμμα, κάθε φορά που εκτελείται το κατηγόρημα ***main*** γίνεται μια διαδικασία εκκαθάρισης των δύο αρχείων που αναφέρθηκαν προηγουμένως. Η διαδικασία αυτή είναι άνοιγμα των αρχείων με ***tell*** εγγραφή, με ***write*** του κενού και μετά κλείσιμο του αρχείου με ***told***. Το module που αναφέρθηκε προηγουμένως περιέχει δύο κανόνες οι οποίοι κάνουν την ίδια δουλεία μόνο που γράφουν σε διαφορετικά αρχεία. Αυτό έγινε για λόγους ευκολότερης κατανόησης. Οι κανόνες αυτοί είναι το ***write_glossary/1*** και ***write_links/1***. Όπως γίνεται κατανοητό το πρώτο γράφει το όρισμα εισόδου, με χρήση append, στο αρχείο *glossary.txt* ενώ το άλλο τα γράφει στο *links.txt*. Τέλος αυτά καλούνται μέσα στο πρόγραμμα στο open_pages κάθε φορά που γίνεται ανάκτηση κειμένου και συνδέσμων. Σε μερικές περιπτώσεις στους συνδέσμους για λόγους μορφοποίησης του κειμένου και του αρχείου μπορεί να μη γίνει κλήση του ***write_links,*** αλλά να γίνει επιτόπου η εγγραφή του κειμένου και της κατάλληλης μορφοποίησης, για να μην υπάρχει πολλές φορές η ίδια διαδικασία, άνοιγμα κλείσιμο αρχείου. Εδώ τελειώνει η τεκμηρίωση της εργασίας και για τη καλύτερη κατανόηση παραθέτονται: ένα UML σχεδιάγραμμα (Εικόνα 14) που δείχνει τη διαγραμματική ροή που ακολουθεί το σύστημα καθώς και στο Παράρτημα Β δύο παραδείγματα εκ των οποίο το πρώτο χρησιμοποιεί τα παραδείγματα που είναι στις εικόνες.



Εικόνα 14. Διαγραμματική δομή του προγράμματος

5.5 Με λίγα λόγια

Η επεξεργασία του κειμένου είναι το τελευταίο αντικείμενο το οποίο και αναλύεται και κατασκευάζεται στο πρόγραμμα. Λέγοντας επεξεργασία κειμένου εννοούνται οποιεσδήποτε ενέργειες γίνονται πάνω σε κανονικό κείμενο, όχι σε html κώδικα, και γίνεται για να μπορέσουμε να γράψουμε το κείμενο στο γλωσσάρι. Τέτοιες ενέργειες είναι η παράβλεψη ειδικών χαρακτήρων για την ορθότερη επεξεργασία, οι διάφοροι έλεγχοι διπλότυπων καθώς επίσης και ο τρόπος με τον οποίο εάν χρειαστεί γίνεται η δημιουργία προτάσεων-λέξεων. Όλα αυτά για να γίνουν χρειάστηκε και εδώ να κατασκευαστούν modules τα οποία θα κάνουν όλες αυτές τις ενέργειες. Τέλος σε αυτό το κεφάλαιο περιγράφεται και πως γίνεται η εγγραφή σε αρχεία κειμένου. Επιπλέον παραθέτονται 2 παραδείγματα για καλύτερη κατανόηση των όσων αναφέρθησαν στα κεφάλαια 3, 4 και 5.

Επίλογος

Οι πληροφορίες παίζουν πολύ σημαντικό ρόλο στη σημερινή κοινωνία, επειδή οι επιχειρήσεις, και όχι μόνο, μπορούν να επωφεληθούν από αυτές. Επειδή όμως πλέον ο όγκος των πληροφοριών είναι πολύ μεγάλος και συνεχώς αυξάνεται, έπρεπε να βρεθεί τρόπος να μπορεί να επεξεργαστούν αυτές οι πληροφορίες γρήγορα και χωρίς μεγάλο κόστος και έτσι αναπτύχθηκαν οι τεχνικές της εξόρυξης δεδομένων. Όπως είδαμε αυτές οι τεχνικές διαφέρουν, και εμείς ασχοληθήκαμε με την ευφυή εξόρυξη δεδομένων,των οποίων μερικές τεχνικές είναι τα νευρωνικά δίκτυα, η εξαγωγή κανώνων, η ομαδοποίηση και ταξινόμηση και η data dimensionallity reduction. Επίσης είδαμε και τα πιο διαδεδομένα προγράμματα που χρησιμοποιούνται από τις επιχειρήσεις.

Στην εργασία χρησιμοποιήθηκε λογικός προγραμματισμός, πιο συγκεκριμένα η γλώσσα λογικού προγραμματισμού Swi-Prolog, για να γίνει η εξόρυξη δεδομένων. Αυτή η επιλογή έγινε λόγω των προτερημάτων των οποίων παρουσιάζει ο λογικός προγραμματισμός τα οποία είναι η ενοποίηση, η πολλαπλή μορφή παραμέτρων εισόδου εξόδου των ορισμάτων των κατηγορημάτων, την αυτόματη οπισθοδρόμηση και οι λογικοί κανόνες οι οποίοι δημιουργούνται στη Prolog. Αυτά τα προτερήματα βοήθησαν στο να γίνει το πρόγραμμα πιο απλό και κατανοητό και να γίνει πιο “γρήγορο”. Απλό και κατανοητό κάνει το πρόγραμμα το ότι είναι μικρότερο, λόγο της πολλαπλής μορφής των παραμέτρων και της ενοποίησης, από κάποιο πρόγραμμα το οποίο θα είχε γραφεί σε κάποια διαδικαστική γλώσσα, ο κώδικας είναι γραμμένος με τρόπο που πλησιάζει πιο κοντά στο τρόπο σκέψης του ανθρώπου κάτι που κάνει τη κατανόηση του πιο εύκολη όπως και οι λογικοί κανόνες που επιτελούν στην πιο εύκολη κατανόηση. Γρηγορότερο κάνουν το πρόγραμμα η ευφυής επεξεργασία και η ευφυΐα της Prolog καθώς επίσης και η αυτόματη οπισθοδρόμηση που προσφέρει η Prolog.

Για την κατασκευή του προγράμματος έπρεπε πρώτα να επιλεχθεί ο τόπος από

τον οποίο θα γίνεται η ανάκτηση των δεδομένων, ο οποίος είναι η Wikipedia, να γίνει κατανοητός ο τρόπος με τον οποίο είναι δομημένος αυτός ο τρόπος και τέλος να γραφεί ο κώδικας. Τα πιο δύσκολα σημεία στη κατασκευή του κώδικα ήταν εξ' αρχής η σύνδεση με το Internet και να η μορφοποίηση του κειμένου το οποίο ανακτάται. Αυτά τα προβλήματα λύθηκαν χρησιμοποιώντας της ενσωματωμένες βιβλιοθήκες της Swi-Prolog, και αυτό έγινε επειδή και η υλοποίηση της Prolog ήταν ικανοποιητική και όπου χρειάστηκε έγινε κάποιες τροποποιήσεις. Ένα άλλο πρόβλημα το οποίο έπρεπε να λυθεί ήταν αυτό των σφαλμάτων που παρουσιάστηκαν κατά την ανάκτηση των κειμένων από σελίδες που είχαν διαφορετική μορφοποίηση, από κείμενα που είχαν χαρακτήρες που δε μπορούν να αναπαρασταθούν κτλ. Αυτά λύθηκαν με χειρισμό των εξαιρέσεων στα σημεία ακριβώς που εμφανίζονται, γιατί αυτός είναι ο μόνος τρόπος να γίνει διαχείριση λαθών από τη Prolog.

Το πρόγραμμα αυτό μπορεί να δημιουργεί σχετικά γρήγορα ένα γλωσσάρι όρων καθώς επίσης και ένα αρχείο που περιέχει όρους οι οποίοι συνδέονται με τον όρο που δόθηκε για αναζήτηση. Αυτό μπορεί να χρησιμοποιηθεί εάν κάποιος θέλει να αποκτήσει σε ένα σύντομο χρονικό διάστημα το περιεχόμενο του κάθε όρου καθώς και να δει με ποιους όρους συνδέονται. Επίσης μπορεί με μερικές τροποποιήσεις, λόγω των modules, να γίνει και ένα εργαλείο μέσω του οποίου μπορεί να γίνεται εξόρυξη κειμένων και συνδέσμων από σελίδες του Παγκόσμιου Ιστού. Μπορεί να γίνει εργαλείο για επεξεργασία κειμένων,όπως είναι ο διαχωρισμός προτάσεων, εγγραφή κειμένων κτλ. Τέλος μπορεί να βοηθήσει μέσω των modules να γίνει καλύτερη μορφοποίηση για τα δεδομένα εισόδου.

Σαν συμπέρασμα μπορούμε να πούμε πως το συγκεκριμένο πρόγραμμα μπορεί να βοηθήσει στη κατασκευή γλωσσαρίου που περιέχει τις περιλήψεις όρων που θέλει ο χρήστης καθώς και όρους οι οποίοι σχετίζονται με τους βασικούς όρους. Επίσης μπορεί με τροποποιήσεις να χρησιμοποιηθεί και για ανάκτηση των κειμένων από σελίδες του διαδικτύου, οπότε τώρα πια που ο χρόνος δεν είναι αρκετός για αναζήτηση κειμένων μπορεί κάποιος να χρησιμοποιήσει αυτό το

εργαλείο για εξόρυξη κειμένων που των ενδιαφέρουν από το Παγκόσμιο Ιστό έτσι ώστε να τα έχει όλα σε ένα αρχείο όπως επίσης και να έχει και τα πιο σημαντικά κομμάτια από αυτά που τον ενδιαφέρουν και όχι εκτενής αναφορές στους όρους.

Βιβλιογραφία

Krzysztof R. Apt (2001): The Logic Programming Paradigm and Prolog.

Chitta Baral and Michael Gelfond (1994): Logic Programming and Knowledge Representation.

Deransart B., Ed, Dbali A., Cervoni L (1996): Prolog: The Standard Reference Manual, Springer erlang.

W. Frawley, G. Piatetsky-Shapiro, C. Matheus (Fall 1992): "Knowledge Discovery in Databases: An Overview", AI Magazine.

D. Hand, H. Mannila, P. Smyth (2001): Principles of Data Mining, MIT Press, Cambridge, MA.

Robert A. Kowalski (1988): THE EARLY YEARS OF LOGIC PROGRAMMING.

Ulf Nilsson and Jan Maluszy´skin (2000): LOGIC, PROGRAMMING AND PROLOG (2ED).

Raghu Ramakrishnan, Johannes Gehrke (2002): Συστήματα Διαχείρισης Βάσεων Δεδομένων, 2η έκδοση, Τόμος Β, εκδόσεις Τζιόλα.

Michael Spivey (1995): An introduction to logic programming through Prolog.

Lipo Wang , Xiuju Fu (2005): Data Mining with Computational Intelligence, Springer.

Ian h. Witten, Eibe Frank, Morgan Kaufmann (2003): Data Mining Practical

Machine Learning Tools and Techniques 2d ed.

Βλαχάβας Ιωάννης, Κεφάλας Πέτρος, Βασιλειάδης Νικόλαος, Κόκκορας Φώτης, Σακελλαρίου Ηλίας, Β. (2006): Τεχνητή Νοημοσύνη, Γ' έκδοση, Γκιούρδας Εκδοτική.

Κωνσταντίνος Διαμαντάρας (2007): Τεχνητά Νευρωνικά Δίκτυα, εκδόσεις Κλειδάριθμος.

Δημοσθένης Ι. Σταμάτης (2007): Λογικός Προγραμματισμός – Η γλώσσα Προγραμματισμού Prolog.

Ηλεκτρονικές Πηγές

http://en.wikipedia.org/wiki/Angoss

http://en.wikipedia.org/wiki/Data_mining

http://en.wikipedia.org/wiki/SAS_System

http://findarticles.com/p/articles/mi_m0EIN/is_2007_May_21/ai_n27246280

http://hcs.science.uva.nl/projects/SWI-Prolog/Manual/

http://softwarefinder.mbtmag.com/software/195-12246/Enterprise-Marketing-Management-EMM/Unica-Affinium-Suite.html

http://www.angoss.com/

http://www.answers.com/topic/spss

http://www.forrester.com/Research/Document/Excerpt/0,7211,41046,00.html

http://www.g-stat.com/?CategoryID=203

http://www.infor.com/product_summary/crm/epiphany/

http://www.macs.hw.ac.uk/~alison/ai3notes/subsectionstar2_3_3_3.html#SECTION00333000000000000000

http://www.portraitsoftware.com/Products/portrait_uplift_optimizer

http://www.przoom.com/news/36134/

http://www.sas.com/

http://www.spss.com

http://www.swi-prolog.org/pldoc/index.html

http://www.thinkanalytics.com/productServices/edm/index.htm

http://www.unica.com/products/predictive_modeling.htm

Παραρτήματα

Παράρτημα Α: Οδηγός χρήσης λογισμικού

Το πρόγραμμα αυτό, για να μπορέσει να λειτουργήσει πρέπει ο χρήστης να έχει εγκατεστημένη στο μηχάνημα του τη swi-Prolog version 5.6.64. Οι απαιτήσεις του συστήματος που πρέπει να έχει είναι οι ίδιες με αυτές της swi-Prolog. Επίσης πρέπει να έχει εγκατεστημένες και όλες τις βιβλιοθήκες που παρέχει η swi. Το πρόγραμμα έχει δοκιμαστεί και τρέχει σε Windows XP, Windows Vista και Windows Seven (στα οποία κατά τη λειτουργία του προγράμματος παρουσιάζονται μερικές διαφορές). Για να μπορέσει ο χρήστης να εκτελέσει το πρόγραμμα πρέπει να ανοίξει το αρχείο Thesis_v1.04.pl Μόλις ανοίξει αυτό το αρχείο τότε το πρόγραμμα εκτελείται αυτόματα. Εάν θέλει αφόσον κάνει χρήση του προγράμματος μια φορά να το ξαναεκτελέσει πρέπει να κάνει κλήση του κατηγορήματος main (?- main.). Για να μπορέσει να κάνει αναζήτηση τότε πρέπει να γράψει τους όρους που θέλει να αναζητήσει με την εξής μορφή :

Όρος1[εσ.όρος1,...,εσ.όροςN], ..., ΌροςM[εσ.όρος1,...,εσ.όροςZ].

Η τελεία είναι σημαντική στη Prolog και πρέπει να χρεισιμοποιήται όποσδηποτε γιατί δηλώνει το τέλος. Πρόσοχη επίσης πρέπει να δωθεί και σε αυτή τη μορφή που περιγράψαμε γιατί ενώ οι εξωτερικοί όροι μπορεί μετά από το κόμμα “,” που είναι το διαχωριστικό τους να έχουν κενό (και να μην έχουν δε πειράζει) οι εσωτερικοί όροι πρέπει οπωσδήποτε να **μην** έχουν κενό μετά το κόμμα “,”. Τέλος εάν δεν θέλει ο χρήστης να δώσει εσωτερικό όρο τότε απλώς αφήνει τη λίστα κενή δηλαδή:

Όρος[].

Παράρτημα Β: Παραδείγματα

Παράδειγμα 1ο:

Σε αυτό το παράδειγμα οι όροι αναζήτησης που θα δώσουμε είναι οι ακόλουθοι:

artificial intelligence[intelligence,mind,computer science],machine learning[adaptive website,adaptive control],sonar[],prolog[programming paradigm],sdfasdf[].

Αυτά σαν αποτέλεσμα αυτής της αναζήτησης στο γλωσσάρι εισάγεται το κείμενο που φαίνεται παρακάτω:

'artificial intelligence'.

' Artificial Intelligence (AI) is the intelligence of machines and the branch of computer science which aims to create it. Major AI textbooks define the field as "the study and design of intelligent agents ," where an intelligent agent is a system that perceives its environment and takes actions which maximize its chances of success. John McCarthy , who coined the term in 1956, defines it as "the science and engineering of making intelligent machines." The field was founded on the claim that a central property of human beings, intelligenceâ\200\\224\the sapience of Homo sapiens â\200\\224\can be so precisely described that it can be simulated by a machine. This raises philosophical issues about the nature of the mind and limits of scientific hubris, issues which have been addressed by myth , fiction and philosophy since antiquity . Artificial intelligence has been the subject of breathtaking optimism, has suffered stunning setbacks and, today, has become an essential part of the technology industry, providing the heavy lifting for many of the most difficult problems in computer science. AI research is highly technical and specialized, deeply divided into subfields that often fail to communicate with each other.

Subfields have grown up around particular institutions, the work of individual researchers, the solution of specific problems, longstanding differences of opinion about how AI should be done and the application of widely differing tools. The central problems of AI include such traits as reasoning, knowledge, planning, learning, communication, perception and the ability to move and manipulate objects. General intelligence (or " strong AI ") is still a long-term goal of (some) research. > '.

' > Intelligence is an umbrella term used to describe a property of the mind that encompasses many related abilities, such as the capacities to reason , to plan , to solve problems , to think abstractly , to comprehend ideas, to use language , and to learn . There are several ways to define intelligence. In some cases, intelligence may include traits such as creativity , personality , character , knowledge , or wisdom . However there is no agreement on which traits define the phenomenon of intelligence agreed upon by a majority across the various concerned disciplines. Theories of intelligence can be divided into those based on a unilinear construct of general intelligence and those based on multiple intelligences . Francis Galton , influenced by his cousin Charles Darwin , was the first to advance a theory of general intelligence. For Galton, intelligence was a real faculty with a biological basis that could be studied by measuring reaction times to certain cognitive tasks. Galton\'s research on measuring the head size of British scientists and ordinary citizens led to the conclusion that head size had no relationship with the person\'s intelligence. Alfred Binet and the French school of intelligence believed that intelligence was an average of numerous dissimilar abilities, rather than a unitary entity with specific identifiable properties. The Stanford-Binet intelligence test has been used by both theorists of general intelligence and multiple intelligence. '.

' Computer science (or computing science) is the study of the theoretical foundations of information and computation , and of practical techniques for their implementation and application in computer systems. It is frequently

described as the systematic study of algorithmic processes that describe and transform information. According to Peter J. Denning , the fundamental question underlying computer science is, "What can be (efficiently) automated?" Computer science has many sub-fields ; some, such as computer graphics , emphasize the computation of specific results, while others, such as computational complexity theory , study the properties of computational problems . Still others focus on the challenges in implementing computations. For example, programming language theory studies approaches to describing computations, while computer programming applies specific programming languages to solve specific computational problems, and human-computer interaction focuses on the challenges in making computers and computations useful, usable, and universally accessible to people .The general public sometimes confuses computer science with vocational areas that deal with computers (such as information technology), or think that it relates to their own experience of computers, which typically involves activities such as gaming, web-browsing, and word-processing. However, the focus of computer science is more on understanding the properties of the programs used to implement software such as games and web-browsers, and using that understanding to create new programs or improve existing ones. '.

' Mind (pronounced /Ë\210\maÉªnd/) refers to the aspects of intellect and consciousness manifested as combinations of thought , perception , memory , emotion , will and imagination , including all of the brain\'s conscious and unconscious cognitive processes. "Mind" is often used to refer especially to the thought processes of reason . Subjectively, mind manifests itself as a stream of consciousness .There are many theories of the mind and its function. The earliest recorded works on the mind are by Zarathushtra , the Buddha , Plato , Aristotle , Adi Shankara and other ancient Greek , Indian and Islamic philosophers . Pre-scientific theories, based in theology , concentrated on the relationship between the mind and the soul , the supernatural, divine or god-given essence of the person. Modern theories, based on scientific understanding of the brain, theorize that the mind is a product of the brain and has both conscious and unconscious

aspects.The question of which attributes make up the mind is also much debated. Some argue that only the "higher" intellectual functions constitute mind: particularly reason and memory . In this view the emotionsâ\200\\224\ love , hate , fear , joy â\200\\224\are more "primitive" or subjective in nature and should be seen as different from the mind. Others argue that the rational and the emotional sides of the human person cannot be separated, that they are of the same nature and origin, and that they should all be considered as part of the individual mind.In popular usage mind is frequently synonymous with thought . It is that private conversation with ourselves that we carry on "inside our heads." Thus we "make up our minds," "change our minds" or are "of two minds" about something. One of the key attributes of the mind in this sense is that it is a private sphere to which no one but the owner has access. No-one else can "know our mind." They can only interpret what we consciously or unconsciously communicate. '.

'machine learning'.

' Machine learning is a scientific discipline that is concerned with the design and development of algorithms that allow computers to learn based on data , such as from sensor data or databases . A major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data. Hence, machine learning is closely related to fields such as statistics , probability theory , data mining , pattern recognition , artificial intelligence , adaptive control , and theoretical computer science . '.

' Adaptive control involves modifying the control law used by a controller to cope with the fact that the parameters of the system being controlled are slowly time-varying or uncertain. For example, as an aircraft flies, its mass will slowly decrease as a result of fuel consumption; we need a control law that adapts itself to such changing conditions. Adaptive control is different from robust control in the sense that it does not need a priori information about the bounds on these

uncertain or time-varying parameters; robust control guarantees that if the changes are within given bounds the control law need not be changed, while adaptive control is precisely concerned with control law changes. '.

---

sonar.

' Sonar (originally an acronym for sound navigation and ranging) is a technique that uses sound propagation (usually underwater) to navigate , communicate with or detect other vessels. There are two kinds of sonar: active and passive. Sonar may be used as a means of acoustic location and of measurement of the echo characteristics of "targets" in the water. Acoustic location in air was used before the introduction of radar . Sonar may also be used in air for robot navigation, and SODAR (an upward looking in-air sonar) is used for atmospheric investigations. The term sonar is also used for the equipment used to generate and receive the sound. The frequencies used in sonar systems vary from infrasonic to ultrasonic . The study of underwater sound is known as underwater acoustics or sometimes hydroacoustics . '.

---

prolog.

' Prolog is a logic programming general purpose fifth generation language associated with artificial intelligence and computational linguistics . It has a purely logical subset, called "pure Prolog", as well as a number of extralogical features.Prolog has its roots in formal logic , and unlike many other programming languages, Prolog is declarative : The program logic is expressed in terms of relations, and execution is triggered by running queries over these relations. Relations and queries are constructed using Prolog\'s single data type, the term . Relations are defined by clauses . Given a query, the Prolog engine attempts to

find a resolution refutation of the negated query. If the negated query can be refuted, i.e., an instantiation for all free variables is found that makes the union of clauses and the singleton set consisting of the negated query false, it follows that the original query, with the found instantiation applied, is a logical consequence of the program. This makes Prolog (and other logic programming languages) particularly useful for database, symbolic mathematics, and language parsing applications. Because Prolog allows impure predicates , checking the truth value of certain special predicates may have some deliberate side effect, such as printing a value to the screen. This permits the programmer to use some amount of conventional imperative programming when the logical paradigm is inconvenient.The language was first conceived by a group around Alain Colmerauer in Marseille , France , in the early 1970s. According to Robert Kowalski , the first Prolog system was developed in 1972 by Alain Colmerauer and Phillipe Roussel. The first implementations of Prolog were interpreters, however, David H. D. Warren created the Warren Abstract Machine , an early and influential Prolog compiler which came to define the "Edinburgh Prolog" dialect which served as the basis for the syntax of most modern implementations. Prolog was one of the first logic programming languages, and remains among the most popular such languages today, with many free and commercial implementations available. While initially aimed at natural language processing , the language has since then stretched far into other areas like theorem proving , expert systems , games, automated answering systems, ontologies and sophisticated control systems , and modern Prolog environments support the creation of graphical user interfaces , as well as administrative and networked applications. Subsequent extensions of Prolog by the original team introduced Constraint Logic Programming abilities into the implementations. '.

---

sdfasdf.

'Page does not exist'.

Στο αρχείο με τους συνδέσμους έχουμε τα εξής αποτελέσματα:

'artificial intelligence'.

['Intelligence',

'Computer science',

'Intelligent agent',

'John McCarthy (computer scientist)',

'Sapience',

'Homo sapiens',

'Mind',

'History of AI',

'Artificial intelligence in fiction',

'Philosophy of AI',

'Antiquity',

'Strong AI'

].

See also:

['Artificial intelligence',

'Articles to be expanded from July 2009',

'Artificial intelligence researchers',

'Metatheory',

'Philosophers of mind',

'Philosophers of science',

'Philosophy of mind',

'Philosophy of science',

'Special:Search/Artificial Intelligence',

'613 Commandments',

'A'*

].

Other Information:

[].

'Intelligence'.

['Umbrella term',

'Mind',

'Reason',

'Plan',

'Problem solving',

'Abstraction',

'Language',

'Learning',

'Creativity',

'Personality psychology',

'Character structure',

'Knowledge',

'Wisdom',

'Multiple intelligences',

'Francis Galton',

'Charles Darwin',

'Biological',

'Cognitive',

'Scientist',

'Alfred Binet',

'Stanford-Binet'

].

See also:

['Intelligence',

'Articles with unsourced statements from March 2008',

'Articles with unsourced statements from January 2008',

'Articles with unsourced statements from April 2009',

'Articles to be expanded from May 2008',

'Articles with unsourced statements from June 2008',

'Articles with unsourced statements from October 2008',

'Philosophers of mind',

'Philosophy of mind',

'Special:Search/Intelligence',

'Abstract object'

].

Other Information:

[].

'Computer science'.

['Information',

'Computation',

'Computer',

'Algorithm',

'Peter J. Denning',

'List of computer science fields',

'Computer graphics',

'Computational complexity theory',

'Computational problem',

'Programming language theory',

'Computer programming',

'Programming language',

'Human-computer interaction',

'Humans',

'Information technology'

].

See also:

['Computer science',

'Articles to be expanded from June 2008',

'Special:Search/Computer science',

'ACM Computing Classification System',

'Abacus',

'Ada Lovelace',

'Alan Turing',

'Algebra',

'Algorithm',

'Algorithm design',

'Algorithmic trading'

].

Other Information:

[].

mind.

['Wikipedia:IPA for English',

'Intellect',

'Consciousness',

'Thought',

'Perception',

'Memory',

'Emotion',

'Free will',

'Imagination',

'Reason',

'Stream of consciousness (psychology)',

'Zarathushtra',

'Gautama Buddha',

'Plato',

'Aristotle',

'Adi Shankara',

'Greek philosophy',

'Indian philosophy',

'Islamic psychological thought',

'Theology',

'Soul',

'Divinity',

'Unconscious mind',

'Reason',

'Memory',

'Love',

'Hate',

'Fear',

'Joy'

].

See also:

['Mind',

'Articles needing additional references from September 2007',

'Articles with unsourced statements from March 2007',

'Metaphysical theories',

'Metaphysicians',

'Metaphysics',

'Philosophy stubs',

'Special:Search/Mind',

'Abstract object',

'Academic',

'Academic journals'

].

Other Information:

[].

'machine learning'.

['Algorithm',

'Computer',

'Data',

'Sensor',

'Database',

'Statistics',

'Probability theory',

'Data mining',

'Pattern recognition',

'Artificial intelligence',

'Adaptive control',

'Theoretical computer science'

].

See also:

['Machine learning',

'Articles lacking sources from August 2008',

'Articles with unsourced statements from May 2008',

'Citation needed',

'Adaptive control',

'Adaptive website',

'Algorithm',

'Artificial Neural Network',

'Artificial intelligence',

'Autonomous robot',

'Bayesian statistics'

].

Other Information:

[].

'adaptive control'.

['Robust control'

].

See also:

['Adaptive control',

'Articles lacking sources from May 2008',

'Dual control theory',

'Gain scheduling',

'Intelligent control',

'Nonlinear control',

'Robust control',

'Robustness',

'System identification',

'Citing sources',

'Reliable sources'

].

Other Information:

[].

sonar.

['Acronym',

'Sound',

'Navigation',

'Acoustic location',

'Acoustic location',

'Radar',

'SODAR',

'Infrasonic',

'Ultrasonic',

'Underwater acoustics',

'Hydroacoustics'

].

See also:

['Sonar',

'Articles with unsourced statements from April 2009',

'Articles needing additional references from January 2009',

'Citation needed',

'AQS-20A',

'Absorption (acoustics)',

'Acoustic Doppler Current Profiler',

'Acoustic Seabed Classification',

'Acoustic Tags',

'Acoustic location',

'Acoustic mine'

].

Other Information:

[].

prolog.

['Logic programming',

'Artificial intelligence',

'Computational linguistics',

'Formal logic',

'Declarative programming',

'Resolution (logic)',

'Refutation',

'Predicates',

'Imperative programming',

'Alain Colmerauer',

'Marseille',

'France',

'Robert Kowalski',

'David H. D. Warren',

'Warren Abstract Machine',

'Natural language processing',

'Automated theorem proving',

'Expert systems',

'Ontology',

'Control system',

'Graphical user interface'

].

See also:

['Prolog',

'Articles with unsourced statements from March 2009',

'ISO standards',

'OSI protocols',

'Special:Search/Prolog',

'110 film',

'135 film',

'ANSI escape code',

'Aaron Sloman',

'Abstract interpretation',

'Abstract syntax tree'

].

Other Information:

['Programming paradigm',

'Logic programming',

'Alain Colmerauer',

'Programming language implementation',

'BProlog',

'Ciao Prolog (page does not exist)',

'ECLiPSe',

'GNU Prolog',

'Logic Programming Associates',

'Mercury programming language',

'Poplog',

'Oz programming language',

'P Sharp',

'Erlang programming language',

'SICStus (page does not exist)',

'Strand (programming language)',

'Strawberry Prolog',

'KL0',

'SWI-Prolog',

'KL1',

'TuProlog',

'Datalog'

'YAP-Prolog',

].

'CSC GraphTalk (page does not exist)',

sdfasdf.

'Programming language dialect',

'Visual Prolog',

'Page does not exist'.

Σε αυτό το αρχείο οι σύνδεσμοι που είναι από μεταδεδομένα εμφανίζονται σαν see also και τα infobox σαν other information.

Μπορεί κάποιος να παρατηρήσει αυτό που έχει αναφερθεί οτι οι εσωτερικοί όροι οι οποίοι αναζητούνται και βρίσκονται με επιτυχία είναι μόνο οι όροι που είναι στο κυρίως κείμενο και όχι στα μεταδεδομένα και στο infobox. Αυτό φαίνεται στην αναζήτηση στον όρο machine learning/ adaptive website το οποίο ανήκει στο μεταδεδομένα και η περιγραφή του δεν εμφανίζεται στο γλωσσάρι, καθώς επίσης και στον όρο prolog/ programming paradigm ο οποίος είναι όρος του infobox ο οποίος και πάλι δεν εμφανίζεται στο γλωσσάρι.

Παράδειγμα 2ο:

Οι όροι αναζήτησης σε αυτό το παράδειγμα είναι:

logic programming[protection policy,compiler],alain colmerauer[],data mining[marketing,alpha consumer,data dredging].

Τα αποτελέσματα στο γλωσσάρι είναι:

'*logic programming*'.

' Logic programming is, in its broadest sense, the use of mathematical logic for computer programming. In this view of logic programming, which can be traced at least as far back as John McCarthy \'s advice-taker proposal, logic is used as a purely declarative representation language, and a theorem-prover or model-generator is used as the problem-solver. The problem-solving task is split between the programmer, who is responsible only for ensuring the truth of programs expressed in logical form, and the theorem-prover or model-generator, which is responsible for solving problems efficiently.However, logic programming, in the narrower sense in which it is more commonly understood, is the use of logic as both a declarative and procedural representation language. It is based upon the fact that a backwards reasoning theorem-prover applied to declarative sentences in the form of implications:treats the implications as goal-reduction procedures:For example, it treats the implication:as the procedure:Note that this is consistent with the BHK interpretation of constructivist logic, where implication would be interpreted as a solution of problem H given solutions of B 1 â\200\¦ B n . However, the defining feature of logic programming is that sets of formulas can be regarded as programs and proof search can be given a computational meaning. This is achieved by restricting the underlying logic to a "well-behaved" fragment such as Horn clauses or Hereditary Harrop formulas . See D. Miller et al., 1991.As in the purely declarative case, the programmer is responsible for ensuring the truth of programs. But since automated proof search is generally infeasible, logic programming as commonly understood also relies on the programmer to ensure that inferences are generated efficiently (see #Problem solving). In many cases,

to achieve efficiency, one needs to be aware of and to exploit the problem-solving behavior of the theorem-prover. In this respect, logic programming is comparable to conventional imperative programming ; using programs to control the behaviour of a program executor. However, unlike conventional imperative programs, which have only a procedural interpretation, logic programs also have a declarative, logical interpretation, which helps to ensure their correctness. Moreover, such programs, being declarative, are at a higher conceptual level than purely imperative programs; and their program executors, being theorem-provers, operate at a higher conceptual level than conventional compilers and interpreters . '.

'A compiler is a computer program (or set of programs) that transforms source code written in a computer language (the source language) into another computer language (the target language , often having a binary form known as object code). The most common reason for wanting to transform source code is to create an executable program.The name "compiler" is primarily used for programs that translate source code from a high-level programming language to a lower level language (e.g., assembly language or machine code). A program that translates from a low level language to a higher level one is a decompiler . A program that translates between high-level languages is usually called a language translator , source to source translator , or language converter . A language rewriter is usually a program that translates the form of expressions without a change of language.A compiler is likely to perform many or all of the following operations: lexical analysis , preprocessing , parsing , semantic analysis, code generation , and code optimization .Program faults caused by incorrect compiler behavior can be very difficult to track down and work around and compiler implementors invest a lot of time ensuring the correctness of their software .The term compiler-compiler is sometimes used to refer to a parser generator , a tool often used to help create the lexer and parser . '.

'alain colmerauer'.

'Page does not exist'.

'data mining'.

' Data mining is the process of extracting patterns from data. As more data are gathered, with the amount of data doubling every three years, data mining is becoming an increasingly important tool to transform these data into information. It is commonly used in a wide range of profiling practices , such as marketing , surveillance , fraud detection and scientific discovery.While data mining can be used to uncover patterns in data samples, it is important to be aware that the use of non-representative samples of data may produce results that are not indicative of the domain. Similarly, data mining will not find patterns that may be present in the domain, if those patterns are not present in the sample being "mined". There is a tendency for insufficiently knowledgeable "consumers" of the results to attribute "magical abilities" to data mining, treating the technique as a sort of all-seeing crystal ball. Like any other tool, it only functions in conjunction with the appropriate raw material: in this case, indicative and representative data that the user must first collect. Further, the discovery of a particular pattern in a particular set of data does not necessarily mean that pattern is representative of the whole population from which that data was drawn. Hence, an important part of the process is the verification and validation of patterns on other samples of data.The term data mining has also been used in a related but negative sense, to mean the deliberate searching for apparent but not necessarily representative patterns in large numbers of data. To avoid confusion with the other sense, the terms data dredging and data snooping are often used. Note, however, that dredging and snooping can be (and sometimes are) used as exploratory tools when developing and clarifying hypotheses. '.

' Product / Pricing / Promotion > Distribution / Service / Retail > Brand management > Account-based marketing > Marketing ethics > Marketing

effectiveness > Market research > Market segmentation > Marketing strategy > Marketing management > Market dominance > Advertising / Branding > Direct marketing / Personal Sales > Product placement / Publicity > Sales promotion / Gender in advertising > Underwriting Printing / Publication / Broadcasting > Out-of-home / Internet marketing > Point of sale / Novelty items > Digital marketing / In-game > Word of mouth > > Marketing is an integrated communications-based process through which individuals and communities discover that existing and newly-identified needs and wants may be satisfied by the products and services of others. Marketing is defined by the American Marketing Association as the activity, set of institutions, and processes for creating, communicating, delivering, and exchanging offerings that have value for customers, clients, partners, and society at large. The term developed from the original meaning which referred literally to going to market, as in shopping, or going to a market to buy or sell goods or services.The Chartered Institute of Marketing defines marketing as "The management process responsible for identifying, anticipating and satisfying customer requirements profitably." Marketing practice tended to be seen as a creative industry in the past, which included advertising , distribution and selling . However, because marketing makes extensive use of social sciences , psychology , sociology , mathematics , economics , anthropology and neuroscience , the profession is now widely recognized as a science, allowing numerous universities to offer Master-of-Science (MSc) programmes. The overall process starts with marketing research and goes through market segmentation , business planning and execution, ending with pre and post-sales promotional activities. It is also related to many of the creative arts. The marketing literature is also infamous for re-inventing itself and its vocabulary according to the times and the culture. > Seen from a systems point of view, sales process engineering views marketing as a set of processes that are interconnected and interdependent with other functions , whose methods can be improved using a variety of relatively new approaches. '.

' Data dredging (data fishing , data snooping) is the inappropriate (sometimes

deliberately so) use of data mining to uncover misleading relationships in data. These relationships may be valid within the test set but have no statistical significance in the wider population.The conventional frequentist statistical hypothesis testing procedure is to formulate a research hypothesis, such as "people in higher social classes live longer", then collect relevant data, then carry out a statistical significance test to see whether the results could be due to the effects of chance. (The last step is called testing against the null hypothesis).A key point in proper statistical analysis is to test a hypothesis with evidence (data) that was not used in constructing the hypothesis. This is critical because every data set will contain some patterns due entirely to chance. If the hypothesis is not tested on a different data set from the same population, it is impossible to determine if the patterns found are chance patterns. See testing hypotheses suggested by the data .As a simplistic example, first throwing five coins, with a result of 2 heads and 3 tails, might lead one to ask why the coin favors tails by fifty percent, whereas first forming the hypothesis might lead one to conclude that only a 5-0 or 0-5 result would be very surprising, since the odds are 93.75% against this happening by chance. In the latter case, it becomes obvious that the data is not anomalous.As a more lyrical example, on a cloudy day, try the experiment of looking for figures in the clouds; if one looks long enough one may see castles, cattle, and all sort of fanciful images; but the images are not really in the clouds, as can be easily confirmed by looking at other clouds.It is important to realize that the alleged statistical significance here is completely spurious â\200\\223\ significance tests do not protect against data dredging. When testing a data set on which the hypothesis is known to be true, the data set is by definition not a representative data set, and any resulting significance levels are meaningless. '.

Στο αρχείο με τους συνδέσμους έχουμε τα εξής:

'logic programming'.

['Mathematical logic',

'John McCarthy (computer scientist)',

'Advice taker',

'Declarative programming language',

'Automated theorem proving',

'Declarative programming language',

'Backwards reasoning',

'BHK interpretation',

'Horn clause',

'Harrop formula',

'Imperative programming',

'Compiler',

'Interpreter (computing)'

].

See also:

['Logic programming',

'Protection policy',

'Programming paradigms',

'Programming language',

'Computable knowledge',

'A.I. Artificial Intelligence',

'ALF (programming language)',

'Abductive Logic Programming',

'Abductive logic programming',

'Abductive reasoning',

'Actor model'

].

Other Information:

[].

compiler.

['Computer program',

'Source code',

'Programming language',

'Object code',

'Executable',

'High-level programming language',

'Assembly language',

'Machine code',

'Decompiler',

'Translator (computing)',

'Rewriting',

'Lexical analysis',

'Preprocessing',

'Parsing',

'Code generation (compiler)',

'Code optimization',

'Compiler correctness',

'Compiler-compiler',

'Parser generator',

'Lexical analysis',

'Parser'

].

See also:

['Compiler',

'Computer science',

'Computer science articles needing expert attention',

'Articles needing expert attention from December 2008',

'Special:Search/Compiler',

'A-0 programming language',

'APL (programming language)',

'Abstract interpretation',

'Alfred V. Aho',

'Algol60',

'Alias analysis'

].

Other Information:

[].

'alain colmerauer'.

'Page does not exist'.

'data mining'.

['Profiling practices',

'Marketing',

'Surveillance',

'Fraud',

'Verification and validation',

'Data dredging',

'Data-snooping bias'

].

See also:

['Data mining',

'Articles needing additional references from April 2009',

'Articles with unsourced statements from July 2009',

'Articles lacking reliable references from April 2009',

'Articles with unsourced statements from July 2008',

'ADVISE',

'Academic journal',

'Adverse drug reaction',

'Alpha consumer',

'Artificial neural networks',

'Association for Computing Machinery'

].

Other Information:

[].

marketing.

['Product marketing',

'Pricing',

'Promotion (marketing)',

'Distribution (business)',

'Service (economics)',

'Retailing',

'Brand management',

'Account-based marketing',

'Marketing ethics',

'Marketing effectiveness',

'Market research',

'Market segmentation',

'Marketing strategy',

'Marketing management',

'Dominance (economics)',

'Advertising',

'Brand',

'Direct marketing',

'Sales',

'Product placement',

'Publicity',

'Sales promotion',

'Gender in advertising (page does not exist)',

'Underwriting spot',

'Printing',

'Publication',

'Broadcasting',

'Out-of-home advertising',

'Internet marketing',

'Point of sale display',

'Promotional item',

'Digital marketing',

'In-game advertising',

'Word of mouth',

'American Marketing Association',

'Chartered Institute of Marketing',

'Advertising',

'Distribution (business)',

'Sales',

'Social sciences',

'Psychology',

'Sociology',

'Mathematics',

'Economics',

'Anthropology',

'Neuroscience',

'Market segmentation',

'Creativity',

'Sales process engineering'

].

See also:

['Marketing',

'Cleanup from May 2008',

'Articles needing additional references from February 2008',

'Books',

'Books/Marketing',

'Citing sources',

'Cleanup',

'Manual of Style',

'Reliable sources',

'Verifiability',

'Special:Search/Marketing'

].

Other Information:

['Product marketing',

'Pricing',

'Promotion (marketing)',

'Distribution (business)',

'Service (economics)',

'Retailing',

'Brand management',

'Account-based marketing',

'Marketing ethics',

'Marketing effectiveness',

'Market research',

'Market segmentation',

'Marketing strategy',

'Marketing management',

'Dominance (economics)',

'Advertising',

'Brand',

'Direct marketing',

'Sales',

'Product placement',

'Publicity',

'Sales promotion',

'Gender in advertising (page does not exist)',

'Underwriting spot',

'Printing',

'Publication',

'Broadcasting',

'Out-of-home advertising',

'Internet marketing',

'Point of sale display',

'Promotional item',

'Digital marketing',

'In-game advertising',

'Word of mouth',

'Template:Marketing',

'Template talk:Marketing',

'http://en.wikipedia.org/w/index.php?title=Template:Marketing&action

=edit'

].

'data dredging'.

['Data mining',

'Test set',

'Frequency probability',

'Statistical hypothesis testing',

'Significance test',

'Null hypothesis',

'Data set',

'Testing hypotheses suggested by the data'

].

See also:

['Data dredging',

'Articles needing additional references from September 2007',

'Cleanup from March 2007',

'Advertising campaign',

'Base rate fallacy',

'Bonferroni correction',

'Bonferroni inequalities',

'Data-snooping bias',

'Data mining',

'Data set',

'False discovery rate'

].

Other Information:

[].

Παράρτημα Γ: Προβλήματα που συνατήθησαν

1ο Πρόβλημα: Ανάκτηση Σελίδων από το Wikipedia.

Η ανάκτηση των σελίδων γίνεται με την ανάκτηση του html κώδικα της σελίδας. Παρατηρήθηκε πως οι σελίδες για τους όρους της wikipedia έχουν τη μορφή http://en.wikipedia.org/wiki/όρος. Για να ανακτηθεί ο html κώδικας χρησιμοποιήθηκε η βιβλιοθήκη της swi-prolog http/http_open μέσω του κατηγορήματος http_open, το οποίο καλεί τη σελίδα μέσω ενός ορίσματος το οποίο είναι το url της σελίδας. Επίσης μπορεί να δεχθεί ως όρισμα και μία λίστα η οποία περιέχει έναν αριθμό από παραμέτρους για τον τρόπο που θα ανακτάται και θα εμφανίζεται ο html κώδικας. Για παράδειγμα στο συγκεκριμένο πρόγραμμα η επιπλέον παράμετρος που χρησιμοποιήθηκε είναι η user_agent(wikipedia) που είναι για να ενημερώσει το κατηγόρημα ότι θα επισκεφθεί σελίδες τις wikipedia. Η επιστροφή του κώδικα γίνεται μέσω ενός stream.

2ο Πρόβλημα: Όροι που δεν έχουν εκχωρήσεις στη wikipedia.

Επειδή υπάρχει περίπτωση να μην έχει γίνει εκχώρηση, στη wikipedia, για κάποιον όρο αναζήτησης, έχει γίνει μια τροποποίηση στο http/http_open για να μπορέσει αντί το πρόγραμμα να βγάζει σφάλμα να αναφέρει πως δεν βρέθηκε η σελίδα. Αυτό έγινε με τροποποίηση στο σημείο που εμφανίζεται το σφάλμα έτσι ώστε να επιστρέφει stream ίσο με '' και όταν το κατηγόρημα που τη καλεί βρει κενό stream αναφέρει πως η σελίδα δεν υπάρχει.

3ο Πρόβλημα: Ανάκτηση κειμένου σελίδας.

Η ανάκτηση κειμένου γίνεται με το να ανακτηθούν το περιεχόμενο των p tags του html κώδικα. Για να γίνει αυτό έπρεπε πρώτα να γίνει μορφοποίηση του html κώδικα σε κατάλληλη μορφή. Αυτό επιτεύχθηκε μέσω της χρήσης του κατηγορήματος load_structure/3 της βιβλιοθήκης sgml. Το κατηγόρημα αυτό δέχεται ως όρισμα το stream που έχει επιστραφεί από το http_open/3 και το

μετατρέπει σε μια δομή (λίστα) μετατρέποντας το κάθε tag σε element(Tag,Attributes,Context). Επίσης μπορεί να δεχθεί και μιά λίστα από χαρακτηριστικά (όπως και το http_open) για τη μορφή που θα έχει η επιστρεφόμενη δομή. Για παράδειγμα εδώ χρησιμοποιήθηκαν τα χαρακτηριστικά space(remove), max_errors(-1), syntax_errors(quiet) τα οποία είναι να μην επιστρέφει στη δομή κενά, να μη βγάζει λάθη και τέλος αν υπάρξουν κάποια λάθη να μην εμφανιστούν.

Εφόσον έγινε η τροποποίηση μέσω του κατηγορήματος get_tags/3 το οποίο είναι υλοποιημένο στο αρχείο get_p_tags.pl. Μέσω αυτού του κατηγορήματος γίνεται έλεγχος (αναδρομικά) και βρίσκει τα element(p,_,Index) και παίρνει το Index, αφαιρεί τα εσωτερικά tags και παίρνει το κείμενο τους και κάνει τη flag true.

Πρόβλμα 4ο: Ανάκτηση Περίληψης wikipedia

Εφόσον ανακτήθηκε το κείμενο όλης της σελίδας η ανάκτηση της περίληψης γίνεται ως εξής:

Έχει παρατηρηθεί οτι εάν έχει περιεχόμενα η σελίδα υπάρχει και περίληψη μέσα στη wikipedia και γίνεται ανάκτηση αυτής της περίληψης.

Υποπρόβλημα 4.1: Περιπτώσεις περίληψης.

α)Συνήθως στις σελίδες που υπάρχει πίνακας περιεχομένων η περίληψη βρίσκεται πριν από αυτόν (και εμφανισιακά και μέσα στον html κώδικα).

β)Υπάρχουν όμως περιπτώσεις που ο πίνακας περιεχομένων βρίσκεται (εμφανισιακά) πλάγια από το κείμενο. Σε αυτές τις περιπτώσεις ο html κώδικας είναι πριν τη περίληψη.

Στην α περίπτωση η ανάκτηση γίνεται όταν βρεθεί element(table,[id = toc|_],_)(id = toc είναι ο πίνακας περιεχομένων) και η flag είναι true, να σταματάει η αναδρομή

και να επιστρέφει ότι έχει ανακτηθεί εώς τώρα.

Στη β περίπτωση εφόσον η flag είναι false και βρεί το element(table,[id = toc|_],_) τότε συνεχίζει να κάνει ανάκτηση p tag μέχρι να βρεί το πρώτο element(h2,_,_) που σημαίνει πως έχει σταματήσει η περίληψη.

Υπάρχει και μια γ περίπτωση στην οποία δεν υπάρχουν καθόλου περιεχόμενα. Σε τέτοιες σελίδες το κείμενο είναι μικρό και κρίθηκε σωστό να ανακτάτε όλο.

Πρόβλημα 6ο: Ανάκτηση links των κειμένων που ανακτούνται

Για να μπορέσουμε να ανακτήσουμε τα links των κειμένων που έχουμε μέχρι στιγμής χρειάστηκε να κατασκευαστεί καινούργιο κατηγόρημα το οποίο θα είναι διαφορετικό από το get_tags/3 λόγω της διαφορετικής δομής του a tag από το p tag. Το κατηγόρημα get_a_tags/2 το οποίο με τη σειρά του καλεί το get_title/2 το οποίο ανακτά τους τίτλους των links. Αυτό γίνεται γιατί έχει παρατηρηθεί πως ο τίτλος του link, που είναι και ο όρος που δείχνει το link, μπορεί να είναι διαφορετικός από το κείμενο που εμφανίζεται στη σελίδα. Για να μπορέσουμε όμως να πάρουμε μόνο τα links που είναι στο κείμενο που εμφανίζουμε, δίνουμε τη δομή με element(_,_,_) και από εκεί πέρα εξάγουμε τα a tags.

Πρόβλημα 7ο: Ανάκτηση των μεταδεδομένων της σελίδας

Η ανάκτηση των μεταδεδομένων μιας σελίδας γίνεται με το κατηγόρημα get_meta/2. Για να μπορέσουμε να το κάνουμε αυτό παίρνουμε από τον html κώδικα της σελίδας τα δεδομένα meta τα οποία έχουν name keywords και μετά παίρνουμε το κείμενο του content και έτσι ανακτούμε όλους τους όρους που χρησιμοποιεί η wikipedia για να περιγράψει τον όρο.

Πρόβλημα 8ο: Ανάκτηση του πίνακα με τις πληροφορίες για τον συγκεκριμένο όρο

Για να μπορέσουμε να πάρουμε τα links από το πίνακα πληροφοριών

καταστεύασα το κατηγόρημα get_table_tags/2 το οποίο παίρνει τα table tags του html κώδικα και μετά παίρνει τον εσωτερικό κώδικα του πίνακα του οποίου το class αρχίζει από infobox και μέσα από αυτόν τον κώδικα παίρνει τα a tags.

Πρόβλημα 9ο: Άνοιγμα των μεταδεδομένων που έχουν παρθεί.

Αυτό γίνεται ως εξής. Όταν ξεκινάμε το open_pages απο το Thesis δίνουμε μία flag η οποία είναι true. Όσο είναι true η flag τότε ανοίγουμε με το open_pages και τη λίστα με τα μεταδεδομένα αλλά βάζουμε στη flag false. Και στις δύο περιπτώσεις ανοίγουμε και την ουρά.

Παράρτημα Δ: Επεξήγηση Module

check_compability

Το **check_compability** είναι ένα module το οποίο ελέγχει εάν τα στοιχεία μίας λίστας υπάρχουν σε μία δεύτερη λίστα. Το module αυτό αποτελείται από 2 κανόνες τον κανόνα ***check_compability/3***, το οποίο είναι και αυτό το οποίο έχει πρόσβαση οποιοδήποτε άλλο πρόγραμμα το οποίο χρησιμοποιεί αυτό το module, και το ***make_it_lowercase/2***.

Το ***check_compatibility/3*** είναι ο κανόνας ο οποίος κάνει το έλεγχο εάν τα στοιχεία της μίας λίστας, σε πεζά γράμματα, υπάρχουν στα στοιχεία της άλλης λίστας, πάλι σε πεζά γράμματα, και εάν υπάρχουν τα ενσωματώνει σε μία νέα λίστα την οποία στο τέλος και επιστρέφει εκεί από όπου κλήθηκε.

Το ***make_it_lowercase/2*** κάνει τη μετατροπή τον στοιχείων σε πεζά.

convert_predicates

Το **convert_predicates** χρησιμοποιείται για να γίνει η μετατροπή των όρων αναζήτησης που έχει δώσει ο χρήστης σε κατάλληλη μορφή για να γίνει η αναζήτηση. Περιέχει τους κανόνες ***convert_space/2*** και ***space_to_underscore/2***.

Το ***convert_space/2*** είναι ο κανόνας που επιτρέπεται να χρησιμοποιηθεί εάν γίνει χρήση αυτού του module. Αυτός ο κανόνας κάνει τον όρο που δέχεται σε μια λίστα με χαρακτήρες και μετά αλλάζει όλους τους κενούς χαρακτήρες με underscore και τέλος ξανακάνει τη λίστα σε όρο.

Το ***space_to_underscore/2*** είναι ο κανόνας που μετατρέπει τους χαρακτήρες που αντιστοιχούν σε κενό να αντιστοιχούν σε underscore.

filtering

Το **filtering** είναι ένα module που αφαιρεί τις διπλότυπες εγγραφές και αφαιρεί κενές λίστες και κενά στοιχεία μέσα από τις λίστες που έχουν δοθεί σαν ορίσματα. Αυτά επιτυγχάνονται μέσω των κανόνων ***filter/2***, ***remove_empty_lists/2***, ***flat/2***, ***add_words/2*** και ***remove_empty_elements/2***. Οι κανόνες οι οποίοι μπορούν να χρησιμοποιηθούν και από άλλα αρχεία είναι το ***filter/2*** και το ***remove_empty_elements/2***.

Το ***filter/2*** έχει σαν σκοπό να αφαιρέσει τις κενές λίστες από τα ορίσματα εισόδου. Αυτό γίνεται μόνο εφόσον οι λίστες έχουν γίνει “επίπεδες”.

Οι λίστες γίνονται επίπεδες μέσω της ***flat/2***.

Το ***remove_empty_elements/2*** είναι ένας κανόνας που χρησιμοποιείται για να αφαιρεθούν τα στοιχεία που είναι οι κενοί χαρακτήρες μέσα από τις λίστες των λέξεων.

get_info_box

Το module **get_info_box** έχει γίνει για να γίνεται η ανάκτηση των πληροφοριών που βρίσκονται στο infobox της wikipedia. Για να γίνει χρήση αυτού του module γίνεται κλήση του κανόνα ***get_table_tags/2***.

Το ***get_table_tags/2*** είναι ένας κανόνας ο οποίος από τη λίστα των element βρίσκει ποια στοιχεία έχουν class = 'infobox...' και από εκεί κάνει εξόρυξη τα στοιχεία που είναι σύνδεσμοι.

Το ***get_class/3*** είναι ο κανόνας που είναι υπεύθυνος να βρίσκει τη class του element. Μετά για να βρεθεί το 'infobox..' χρησιμοποιείται το ***check_info/3***.

Το ***check_info/3*** για να μπορέσει να το αντιστοιχίσει πρέπει να κάνει μετατροπή

του string που έχει σε μία λίστα από ascii κωδικούς και μετά να κάνει τον έλεγχο και αφού γίνει ο έλεγχος και βρεθεί ταίριασμα τότε γίνεται επιστροφή των περιεχομένων του infobox.

get_p_tags

Το module **get_p_tags** χρησιμοποιείται για την ανάκτηση των σημάνσεων παραγράφων όπως επίσης και για την εξόρυξη των συνδέσμων μέσα από τα κείμενα τα οποία ανακτούνται. Για να γίνει αυτό οι κύριοι κανόνες είναι ο ***get_tags/5*** και ο ***get_a_tags/2***.

Ο ***get_tags/5*** είναι ένας κανόνας ο οποίος κάνει ανάκτηση του κατάλληλου κειμένου για κάθε όρο και επίσης μέσω κλήσης του ***get_a_tags/2*** κάνει εξόρυξη και των συνδέσμων που βρίσκονται μέσα σε αυτό το κείμενο.

Το ***remove_inner_tags/2*** είναι ένα κατηγόρημα το οποίο βοηθάει στην εξάλειψη εσωτερικών σημάνσεων σε περιπτώσεις σημάνσεων που είναι κείμενο (p tag). Αυτό βέβαια δε σημαίνει πως το κείμενο δεν αφαιρείται αλλά απλώς επιστρέφεται σαν απλό κείμενο και όχι σαν σήμανση.

Το ***get_a_tags/2*** παίρνει τη λίστα των element και βρίσκει τους συνδέσμους μέσα σε αυτή τη λίστα και παίρνει τους τίτλους και τους επιστρέφει σε μία λίστα.

Το ***get_title/2*** είναι ο κανόνας που χρησιμοποιείται για την εξόρυξη των τίτλων από τους συνδέσμους.

get_see_also

Το module **get_see_also** είναι το module το οποίο χρησιμοποιείται για να γίνει εξόρυξη των μεταδεδομένων από τις σελίδες της wikipedia. Μέσω του ***get_meta/2*** γίνεται χρήση του module.

Το ***get_meta/2*** είναι ένα κατηγόρημα το οποίο παίρνει τη λίστα με τα element και βρίσκει τα μεταδεδομένα μέσω των element meta και τα επιστρέφει σε μορφή λίστας.

Το ***get_name/2*** είναι το κατηγόρημα το οποίο παίρνει τα elements meta και μετά με τη χρήση του ***get_content/2*** παίρνει τα μεταδεδομένα.

http_open

Το module **http_open** είναι ένα module από τη βιβλιοθήκη της Swi-Prolog που χρησιμοποιείται σε αυτό το πρόγραμμα για να γίνει η σύνδεση στο διαδίκτυο. Στο παρόν πρόγραμμα σε αυτό το module έχουν γίνει μερικές τροποποιήσεις ώστε να είναι εφικτή η διαχείριση των σφαλμάτων με ένα συγκεκριμένο τρόπο.

make_words

Το **make_words** είναι ένα module που χρησιμεύει στη δημιουργία λίστας λέξεων. Η χρήση του module αυτού γίνεται με κλήση 2 κανόνων είτε με τη κλήση του ***words/2*** ή του ***remove_empty_lists/2*** ή τέλος του convert_words.

Το ***word/2*** χρησιμοποιείται για να διασπάσει τις προτάσεις σε λέξεις και μετά να αφαιρέσει τις κενές λίστες που τυχόν δημιουργούνται από τη διάσπαση.

Ο κανόνας που είναι υπεύθυνος για τη διάσπαση των προτάσεων σε λέξεις είναι το ***tokenize/2***, ενώ η διαγραφή των κενών λιστών γίνεται με τη χρήση του κανόνα remove_empty_lists.

Το ***tokenize/2*** είναι ένας κανόνας ο οποίος δέχεται μια λίστα από λίστες και επιστρέφει αυτή τη λίστα με λέξεις και χωρίς εσωτερικές λίστες.

Το ***make_words/2*** κάνει ένα αρχείο από string σε μια λίστα από λίστες-λέξεις κάνοντας το string σε λίστες ascii κωδικών.

Η μετατροπή του string σε λίστα από λίστες-λέξεις γίνεται μέσω του ***split_list/2***, που ελέγχει εάν υπάρχει ο κενός χαρακτήρας και δημιουργεί τις λίστες λέξεων.

Για να γίνουν ξανά λέξεις οι λίστες των ascii κωδικών χρησιμοποιείται το ***convert_words/2***.

Το ***remove_symbols/2*** παίρνει μια λίστα από το γεγονός ***symbols/1***, που περιέχει τους ascii κωδικούς των συμβόλων, και αφαιρεί αυτά τα σύμβολα.

merge_to_text

Το module **merge_to_text** είναι το υπεύθυνο module του να κάνει τις λίστες λέξεων που έχουν γίνει από το **make_words** σε κείμενο. Αυτή η διεργασία γίνεται μέσω του κανόνα ***merge_to_text/2***. Ένας άλλος κανόνας που είναι προσβάσιμος σε αυτό το module είναι και το ***concat_elements/2***.

Το ***merge_to_text/2***, όπως αναφέρθηκε, κάνει τη μετατροπή των λιστών που περιέχουν λέξεις σε κείμενο. Επίσης αφαιρεί τις αναφορές που τυχόν υπάρχουν μέσα στο κείμενο.

Η απαλοιφή των αναφορών γίνεται μέσω του ***divide/2*** το οποίο δέχεται μια λίστα με λίστες ascii κωδικών και ψάχνει να βρει τους κωδικούς τους οποίους αντιστοιχούν σε '[' και αφαιρεί το περιεχόμενο που είναι αριθμός, αυτό ορίζεται μέσω του γεγονότος ***reference/1***, μέχρι να βρει ']'.

Το ***import_space/2*** είναι ένας κανόνας που προσθέτει κενά στο κείμενο, γιατί σε μερικές περιπτώσεις γίνεται συνένωση προτάσεων από προηγούμενη αφαίρεση κενών.

Το ***concat_elements/2*** δέχεται μια λίστα από λέξεις τις συγχωνεύει και μετά τις επιστρέφει σαν string.

open_pages

Το **open_pages** είναι το module που συντονίζει όλες τις ενέργειες για να γίνει η ανάκτηση των κειμένων και των συνδέσμων καθώς επίσης και η εγγραφή τους στα αρχεία κειμένου.

Ο κανόνας ***open_pages/4*** είναι ο κανόνας που συντονίζει τις ενέργειες για τους εξωτερικούς όρους καθώς επίσης και καλεί το ***open_pages/3*** που συντονίζει τις ενέργειες για τους εσωτερικούς όρους.

pretty_print

Το **pretty_print** είναι και αυτό ένα module της βιβλιοθήκης της Swi-Prolog και ο μόνος λόγος που χρησιμοποιείται είναι για να είναι πιο ωραία η εμφάνιση των κειμένων και των συνδέσμων που γίνεται στη παρούσα εργασία.

reading

Το **reading** είναι ένα module ανάγνωσης από το πληκτρολόγιο. Αυτό που κάνει ο κανόνας ***reading/1*** είναι να παίρνει το πρώτο χαρακτήρα και μετά να τους ενώνει όλους σχηματίζοντας το όρισμα μέσω του ***iread/2***.

Το ***iread/2*** ενεργοποιεί το ***iread/3***. Το ***iread/3*** παίρνει επαναληπτικά όλους τους χαρακτήρες που έχει δώσει ο χρήστης και τους ενώνει μεταξύ τους εφόσον έχει κάνει έλεγχο εάν κάποιος από τους χαρακτήρες είναι ειδικός χαρακτήρας μέσω του γεγονότος ***special_characters/1***.

reading2

Το module **reading2** είναι μια επέκταση του module **reading** και ο σκοπός του είναι να καταλαβαίνει και να ξεχωρίζει εσωτερικά και εξωτερικά ορίσματα από τη μορφοποίηση που έχει δοθεί.

Το ***reading2/3*** είναι ο κανόνας που μπορεί να χρησιμοποιηθεί και από άλλα προγράμματα και αυτό είναι ο κανόνας που καλεί τις υπόλοιπες ενέργειες για να γίνει ο διαχωρισμός των εσωτερικών και των εξωτερικών όρων.

Το ***divide/3*** είναι ο κανόνας που κάνει το διαχωρισμό αλλά σε μορφή ascii κωδικών.

Ο κανόνας ***list_to_text/2*** είναι αυτός που μετατρέπει τις λίστες που δημιουργούνται από το ***divide/3*** σε χαρακτήρες.

Το ***lists_to_text/2*** χρησιμοποιεί το ***list_to_text/2*** για να μετατρέψει μια λίστα από λίστες κωδικών σε κείμενο.

sgml

Το τελευταίο module που χρησιμοποιούμε από τη βιβλιοθήκη της Swi-Prolog είναι το **sgml** το οποίο χρησιμεύει για να γίνει η μορφοποίηση του html κώδικα που έχουμε ανακτήσει από τις σελίδες του διαδικτύου.

writing

Το **writing** είναι το module που χρησιμοποιήθηκε για να γίνονται οι εγγραφές κειμένων σε αρχεία μέσω 2 κανόνων του κανόνα ***write_glossary/1*** και του ***write_links/1***.

Το ***write_glossary/1*** είναι ο κανόνας που είναι υπεύθυνος για τη συγγραφή του γλωσσάρι.

Το ***write_links/1*** είναι ο κανόνας που είναι υπεύθυνος για να αναγράφονται τα σχετικά links με κάθε όρο σε νέο αρχείο κειμένου.