ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Μέθοδοι αυτόματου εντοπισμού σφαλμάτων και βελτίωσης wrappers με χρήση επαυξητικών μεθόδων μάθησης (Wrapper Maintenance)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΧΑΡΑΛΑΜΠΟΥ Ε. ΤΣΟΥΡΑΚΑΚΗ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2006

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

Μέθοδοι αυτόματου εντοπισμού σφαλμάτων και βελτίωσης wrappers με χρήση επαυξητικών μεθόδων μάθησης (Wrapper Maintenance)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΧΑΡΑΛΑΜΠΟΥ Ε. ΤΣΟΥΡΑΚΑΚΗ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 3η Νοεμβρίου 2006.

.....	.....	.....
Τιμολέων Σελλής	Ιωάννης Βασιλείου	Ανδρέας Γ. Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2006

.....

ΧΑΡΑΛΑΜΠΟΣ Ε. ΤΣΟΥΡΑΚΑΚΗΣ

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

Copyright © Χαράλαμπος Τσουρακάκης, 2006
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.
Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου

Περίληψη

Wrappers είναι ειδικά προγράμματα που εξάγουν αυτόματα πληροφορία από HTML ιστοσελίδες ενός διαδικτυακού τόπου και την τοποθετούν σε μία δομημένη μορφή. Η εξαγωγή βασίζεται σε ένα σύνολο κανόνων που εκμεταλλεύεται το κοινό layout που συνήθως έχουν οι ιστοσελίδες ενός διαδικτυακού τόπου. Επειδή όμως το layout και το περιεχόμενο των ιστοσελίδων μεταβάλλεται συχνά, οι wrappers μπορεί να πάψουν να εξάγουν την επιθυμητή πληροφορία. Έτσι προκύπτει η ανάγκη για την ύπαρξη ενός συστήματος που θα αντιλαμβάνεται αυτόματα αν ο wrapper δεν εξάγει την επιθυμητή πληροφορία (wrapper verification πρόβλημα) και στην συνέχεια θα μπορεί να παράγει έναν νέο wrapper, προσαρμοσμένο στις αλλαγμένες ιστοσελίδες (wrapper reinduction πρόβλημα). Τα δύο αυτά προβλήματα είναι γνωστά με τον όρο wrapper maintenance πρόβλημα. Αντικείμενο αυτής της διπλωματικής είναι η ανάπτυξη ενός αξιόπιστου wrapper maintenance συστήματος. Στο wrapper verification τμήμα αναπτύχθηκε ένας πρωτότυπος, σύνθετος αλγόριθμος ο οποίος προσπαθεί να εκμεταλλευτεί όσο το δυνατόν περισσότερο τη δομή της εξαγόμενης πληροφορίας. Δέχεται σαν εισόδους την πληροφορία που εξάγεται κατά τη διάρκεια της ορθής λειτουργίας του wrapper και την πληροφορία κατά την διάρκεια της άγνωστης λειτουργίας του wrapper. Για κάθε μία είσοδο δημιουργεί ένα διάνυσμα μεταπληροφορίας και στη συνέχεια ελέγχει αν τα δύο παραγόμενα διανύσματα μπορούν να θεωρηθούν όμοια. Τα πειραματικά αποτελέσματα που λάβαμε έδειξαν ότι ο αλγόριθμος μας είναι ιδιαίτερα αξιόπιστος στην πράξη αλλά και εύρωστος, υπό την έννοια ότι λαμβάνει υπ'όψιν του μικρές ατέλειες του wrapper. Στο wrapper reinduction μέρος υλοποιήθηκε μία κλασσική προσέγγιση. Συγκεκριμένα, αναζητάμε πληροφορία που εξάχθηκε κατά τη διάρκεια της ορθής λειτουργίας του wrapper στις αλλαγμένες ιστοσελίδες. Τα συνολικά πειραματικά αποτελέσματα αποδεικνύουν ότι το wrapper maintenance σύστημα ARMAGEDDON που αναπτύχθηκε είναι ιδιαίτερα αξιόπιστο.

Λέξεις Κλειδιά: Μηχανική μάθηση, μάθηση προτύπων, wrapper maintenance, wrapper verification, wrapper reinduction, έλεγχος υποθέσεων, ολοκλήρωση δεδομένων

Abstract

Wrappers are special programs that mechanically extract data from HTML web pages and store them in a structured form. The extraction process is based on a set of rules which exploit the common layout of a web site's web pages. Because page layout and content often change, wrappers stop extracting the desirable information. Thus, we need a system that will be able to detect automatically whether the wrapper does or doesn't extract the desirable information (wrapper verification problem) and then, if any malfunction of the wrapper is diagnosed, produce a correct wrapper, adapted to the new web pages (wrapper reinduction problem). These two problems are usually called wrapper maintenance problem. The goal of this work was the development of a reliable wrapper maintenance system. In the wrapper verification part we developed a novel, complex, content based algorithm which tries to exploit the structure of the extracted data. It takes as input two sets of extracted data. The first one was extracted while the wrapper was working properly and the second one during the time interval that we perform the test. The algorithm creates for each input a meta-information vector that describes the structure of the extracted data and then performs a complex test to decide whether the two vectors are sufficiently similar. The experimental results showed that our algorithm is efficient and robust. In the wrapper reinduction part we developed a classical approach to the problem, searching valid examples of the desirable information in the altered web pages. The experimental results showed that the wrapper maintenance system ARMAGEDDON we developed is a reliable one.

Keywords: Machine learning, pattern learning, wrapper maintenance, wrapper verification, wrapper reinduction, hypothesis testing ,data integration

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή κύριο Τίμο Σελλή για το ενδιαφέρον και την εμπιστοσύνη που έδειξε στο πρόσωπό μου αλλά και για την υποδειγματική του στάση ως καθηγητής. Τα στοιχεία αυτά ήταν καθοριστικά για την λήψη προσωπικών αποφάσεων σχετικών με την πορεία μου στο χώρο της επιστήμης των υπολογιστών.

Ευχαριστώ τον ερευνητή Γεώργιο Παλιούρα για την άριστη συνεργασία που είχαμε αυτούς τους μήνες, για το θερμό ενδιαφέρον που έδειξε κατά τη διάρκεια της εκπόνησης της εργασίας αυτής αλλά και γιατί με έφερε σε μία πρώτη επαφή με το επιστημονικό πεδίο της μηχανικής μάθησης.

Τέλος θέλω να ευχαριστήσω τους γονείς μου, Ευτύχιο και Αλίκη, την αδερφή μου Μαρία, τον θείο Συμεών και τον θείο Μιχάλη καθώς και τον φίλο μου Δημήτρη Τσαπάρα που αυτά τα 5 χρόνια των σπουδών μου στο Πολυτεχνείο ήταν πάντα δίπλα μου υποστηρίζοντας και ενισχύοντας τις προσπάθειες μου με κάθε δυνατό τρόπο.

Πίνακας Περιεχομένων

1

Εισαγωγή

1.1 Γενικά

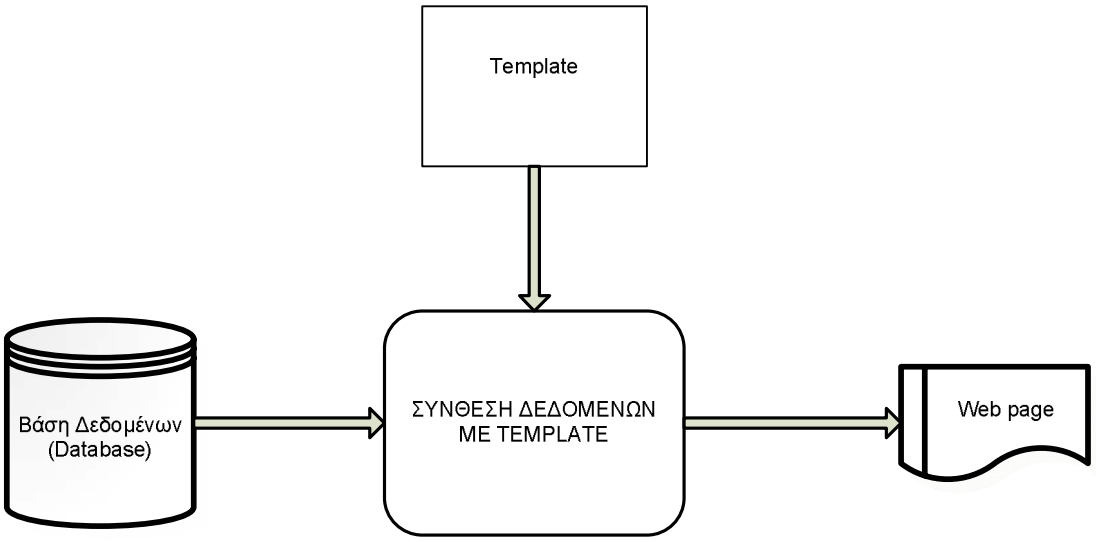
Το διαδίκτυο αποτελεί σήμερα παγκοσμίως μία από τις κυριότερες πηγές πληροφόρησης και βάση των στατιστικών στοιχείων ([IGS06]) προβλέπεται ότι στο μέλλον θα αποτελεί την κυριότερη. Στοιχεία που καθιστούν βάσιμη μια τέτοια πρόβλεψη είναι τα εξής :

- Η πρόσβαση σε αυτό είναι εύκολη με ολοένα μικρότερο κόστος.
- Είναι δημοκρατικό μέσο, παρέχοντας στον οποιοδήποτε την δυνατότητα να διαθέτει δική του ιστοσελίδα.
- Είναι παγκόσμιο μέσο καταργώντας την ύπαρξη συνόρων και αποστάσεων.

Όλα τα παραπάνω ακούγονται και είναι όντως πολύ θετικά στοιχεία . Όμως υπάρχουν και σημαντικά προβλήματα που αντιμετωπίζει ο χρήστης του διαδικτύου. Ας θεωρήσουμε ένα απλό, καθημερινό παράδειγμα ενός χρήστη που θέλει να βρει την καλύτερη προσφορά από κάποιο ταξιδιωτικό γραφείο για να κάνει τις διακοπές του σε ένα συγκεκριμένο μέρος. Μιας και το πιθανότερο είναι να μην γνωρίζει εξαρχής τη διεύθυνση κάποιου τέτοιου γραφείου, θα χρησιμοποιήσει μία μηχανή αναζήτησης (π.χ Google,Altavista,Yahoo) για να βρει κάποια sites (διαδικτυακούς τόπους) που θα ψάξει . Εδώ ήδη έχει εμφανιστεί το πρώτο πρόβλημα , αυτό του αόρατου διαδικτύου (deep web). Με αυτό τον όρο περιγράφουμε το σύνολο των ιστοσελίδων που δεν είναι ορατές από τις μηχανές αναζήτησης και κατά συνέπεια ο μόνος τρόπος για να τις επισκεφθεί ο χρήστης είναι να ξέρει το URL τους. Στη συνέχεια από το σύνολο των ιστοσελίδων που του εμφανίστηκαν από την μηχανή αναζήτησης θα αρχίσει να ψάχνει μία ικανοποιητική προσφορά Αυτή η διαδικασία είναι χρονοβόρα και πολλές φορές

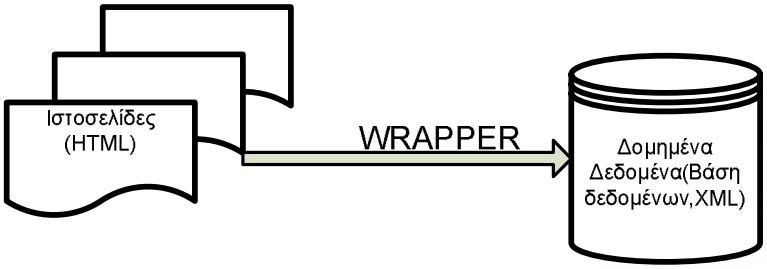
δεν δίνει και το βέλτιστο αποτέλεσμα. Για το σκοπό αυτό ήδη διεξάγεται μεγάλη έρευνα για την ανάπτυξη πρακτόρων του διαδικτύου (web agents ή softbots εκ των λέξεων Software robots), δηλαδή προγραμμάτων τα οποία θα γνωρίζουν τι θέλει ο χρήστης και θα κάνουν την παραπάνω διαδικασία αυτόματα παρέχοντας στο χρήστη του παραδείγματός μας όνομα ταξιδιωτικού γραφείου και τιμή από διάφορες ιστοσελίδες ταξιδιωτικών γραφείων ώστε χωρίς να καταβάλλει καμία ιδιαίτερη προσπάθεια να μπορεί να επιλέξει την καλύτερη γι αυτόν προσφορά. Δυστυχώς όμως το διαδίκτυο με την μορφή που έχει σήμερα δεν είναι φιλικό για τους πράκτορες. Αυτό γιατί αρχικά σχεδιάστηκε «ανθρωποκεντρικά» προοριζόμενο για άμεση επαφή με το χρήστη και όχι «προγραμματοκεντρικά» δηλαδή με σκοπό τη διευκόλυνση προγραμμάτων στην αυτόματη συλλογή, διαχείριση, και επεξεργασία της πληροφορίας από τον ιστό. Από τα παραπάνω γίνεται φανερό ότι η εύρεση χρήσιμης πληροφορίας στο διαδίκτυο είναι μία επίπονη και χρονοβόρα διαδικασία.

Αν και η πλήρης αντιμετώπιση των παραπάνω προβλημάτων στην γενική τους μορφή δεν είναι προς το παρόν εφικτή , έχουν δοθεί λύσεις σε απλούστερα υποπροβλήματα τα οποία είναι πολύ σημαντικά ως προς τις πρακτικές τους εφαρμογές. Μέσα σε αυτές τις περιπτώσεις που έχουν αντιμετωπιστεί είναι η αυτόματη εξαγωγή πληροφορίας (information extraction) α πό ιστοσελίδες με την χρήση των wrappers.Wrapper είναι ένα σύνολο κανόνων μαζί με το κατάλληλο πρόγραμμα για να εκτελεστούν οι κανόνες αυτοί. Ένα μεγάλο ποσοστό ιστοσελίδων στο διαδίκτυο (80% σύμφωνα με την πηγή [SA99]) είναι προϊόν σύνθεσης ενός template με πληροφορία που προέρχεται από μία βάση δεδομένων.



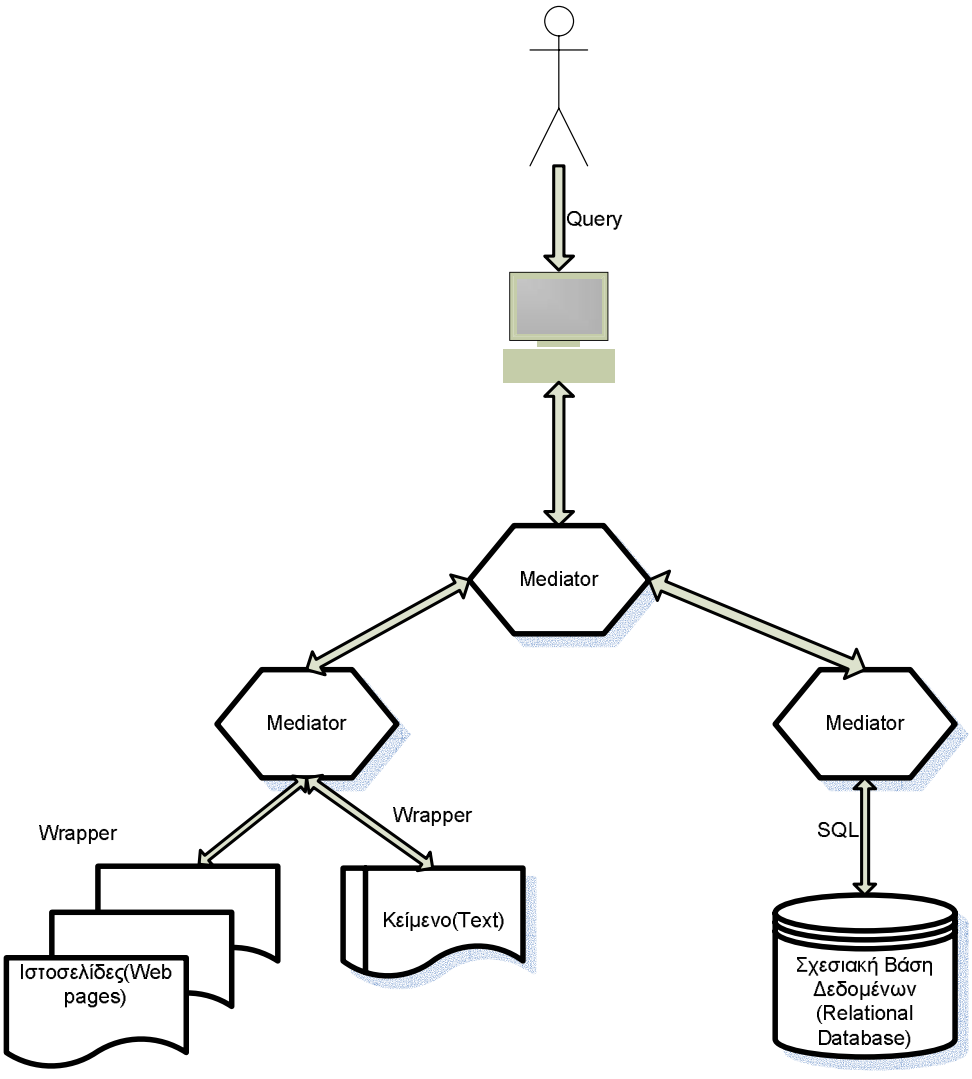
σχήμα 1.1:τρόπος δημιουργίας ιστοσελίδας με χρήση template

Δυστυχώς όμως αυτή η έμφυτη δομή της πληροφορίας χάνεται μέσα στις ιστοσελίδες. Οι wrappers ουσιαστικά καλύπτουν αυτό το κενό όπως φαίνεται και στο παρακάτω σχήμα:



σχήμα 1.2:Λειτουργία wrapper

Οι wrappers έχουν ποικίλες εφαρμογές. Μία από τις σημαντικότερες πρακτικές εφαρμογές των wrappers είναι η εκτεταμένη χρήση τους από τους πράκτορες του διαδικτύου καθώς κάθε φορά που επισκέπτονται ένα site χρησιμοποιούν τον κατάλληλο wrapper για τις ιστοσελίδες του για να εξάγουν την επιθυμητή πληροφορία . Βρίσκουν επίσης εφαρμογή στην ολοκλήρωση δεδομένων (data integration). Μπορούν να χρησιμοποιηθούν για να δημιουργήσουν μία ενιαία βάση δεδομένων από δεδομένα που βρίσκονται σε διαφορετικές πηγές. Αυτό φαίνεται στο παρακάτω σχήμα :



σχήμα 1.3:Ολοκλήρωση δεδομένων(Data Integration)

Παρ' όλα αυτά οι wrappers που παράγονται για ιστοσελίδες (web wrappers) πολύ συχνά κατά τη διάρκεια της ζωής τους αντιμετωπίζουν προβλήματα. Αυτό συμβαίνει γιατί οι κανόνες με βάση τους οποίους κάνουν την εξαγωγή των δεδομένων μπορεί να πάψουν να ισχύουν γιατί οι ιστοσελίδες υπέστησαν αλλαγή είτε στο περιεχόμενό τους είτε στο layout τους . Αυτό το φαινόμενο παρατηρείται πολύ συχνά. Αυτό σημαίνει ότι οι wrappers που είχαν σχεδιαστεί γι αυτές τις ιστοσελίδες παύουν να λειτουργούν σωστά είτε εξάγοντας λάθος δεδομένα είτε εξάγοντας λιγότερα δεδομένα από αυτά που θα έπρεπε να εξαχθούν είτε χωρίς να εξάγουν καθόλου πληροφορία. Τότε τίθεται το εξής πρόβλημα : Μπορεί να εντοπιστεί αυτόματα η δυσλειτουργία του wrapper -Wrapper Verification problem[1] (πρόβλημα επαλήθευσης ορθής λειτουργίας wrapper)- και αν ναι, μπορεί να παραχθεί ένας νέος wrapper που θα μπορεί να εξάγει την επιθυμητή πληροφορία από τις αλλαγμένες ιστοσελίδες-Wrapper Reinduction problem[1] (πρόβλημα παραγωγής σωστού wrapper); Το πρόβλημα αυτό ονομάζεται στη διεθνή βιβλιογραφία Wrapper Maintenance problem[1] (αυτόματος εντοπισμός σφαλμάτων και παραγωγής νέων, σωστών wrappers) .

1.2 Αντικείμενο εργασίας

Αντικείμενο αυτής της εργασίας είναι η ανάπτυξη ενός συστήματος wrapper maintenance. Ιδιαίτερη έμφαση δόθηκε στο wrapper verification μέρος για δύο κυρίως λόγους :

- Αν δεν διαπιστωθεί κάποιο πρόβλημα στην λειτουργία του wrapper δεν εκτελείται το reinduction σύστημα. Έτσι ένα πολύ καλό reinduction σύστημα δεν θα αξιοποιούταν πλήρως χωρίς την ύπαρξη ενός καλού verification συστήματος.
- Η επαγωγή ενός νέου wrapper αν υπάρχει ένα wrapper induction system[1] (σύστημα παραγωγής wrappers με χρήση επαγωγικής μάθησης) όταν έχει διαπιστωθεί το πρόβλημα απαιτεί στην χειρότερη περίπτωση κάποια λεπτά από το χρόνο του χρήστη για να δώσει κάποια παραδείγματα της πληροφορίας που θέλει να εξάγεται στις αλλαγμένες ιστοσελίδες. Αντίθετα για τον έλεγχο του αν η εξαχθείσα πληροφορία είναι η ζητούμενη, θα απαιτούνταν διαρκής έλεγχος κάθε φορά που θα λειτουργούσε ο wrapper. Συνεπώς σε πρώτη φάση το verification μέρος απαιτεί μία καλή αυτοματοποίηση.

Έτσι στην εργασία αυτή αναπτύχθηκε ένας νέος αλγόριθμος που προσπαθεί να διαγνώσει σφάλματα στην λειτουργία ενός wrapper. Ο αλγόριθμος αυτός εκμεταλλεύεται τη δομή της εξαγόμενης πληροφορίας προσδιορίζοντας κάποια χαρακτηριστικά γνωρίσματα της (content-based αλγόριθμος). Σχετικά με το wrapper reinduction πρόβλημα, στόχος μας ήταν να εντοπίσουμε στις αλλαγμένες ιστοσελίδες παραδείγματα της επιθυμητής πληροφορίας με τα

---

[1]Από δω και στο εξής θα προτιμούνται οι αγγλικοί όροι

οποία θα τροφοδοτηθεί ένα wrapper induction system ώστε να πάρουμε έναν σωστό wrapper. Στο μέρος αυτό υλοποιήθηκε μία απλή ιδέα πρακτικής φύσεως. Αν και δεν εγγυάται πάντα ότι θα βρει κάποιο παράδειγμα ακόμα και αν υπάρχει τέτοιο στις ιστοσελίδες καταφέρνει σε αρκετές περιπτώσεις να δώσει ικανοποιητικό αποτέλεσμα.

1.3 Οργάνωση τόμου

Στο πρώτο κεφάλαιο γίνεται μία εισαγωγή στο πρόβλημα με το οποίο ασχολείται η εργασία και δίνεται συνοπτικά η δομή της εργασίας.

Στο δεύτερο κεφάλαιο αναπτύσσονται κάποιες βασικές γνώσεις που αφορούν τους τομείς με τους οποίους σχετίζεται η εργασία (μηχανική μάθηση, εξαγωγή πληροφορίας).

Στο τρίτο κεφάλαιο παρουσιάζονται κάποιες από τις πιο γνωστές εργασίες πάνω στα θέματα του wrapper induction και του wrapper maintenance.

Στο τέταρτο κεφάλαιο παρουσιάζεται το σύστημα ARMAGEDDON το οποίο αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας.

Στο πέμπτο κεφάλαιο παρουσιάζονται τα αποτελέσματα που λάβαμε κατά την εκτέλεση των πειραμάτων.

Στο έκτο κεφάλαιο συνοψίζουμε, παραθέτοντας κάποια συμπεράσματα που προέκυψαν και δίνουμε κάποιες ερευνητικές προεκτάσεις της υπάρχουσας εργασίας που θα μπορούσαν να οδηγήσουν σε μία καλύτερη έκδοσή της στο μέλλον.

Τέλος στο έβδομο κεφάλαιο παρατίθεται η βιβλιογραφία που χρησιμοποιήθηκε για την εκπόνηση αυτού του τόμου.

2

Θεωρητικό Υπόβαθρο

2.1 Βασικές έννοιες μηχανικής μάθησης

Η μάθηση είναι μια κοινή δραστηριότητα όλων σχεδόν των έμβιων όντων, αναγκαία για την επιβίωση τους καθώς και βασικό χαρακτηριστικό κάθε νοήμονος όντος. Συνεπώς, η μηχανική μάθηση (machine learning) δεν θα μπορούσε παρά να είναι κεντρικός τομέας της τεχνητής νοημοσύνης (artificial intelligence). Ένας τυπικός ορισμός της είναι ο εξής : Μηχανική μάθηση είναι ο κλάδος της τεχνητής νοημοσύνης που έχει σαν αντικείμενο την ανάπτυξη αλγόριθμων και τεχνικών που δίνουν την δυνατότητα στα προγράμματα να μαθαίνουν. Ένα πρόγραμμα θεωρείται ότι μαθαίνει από την εμπειρία του E ως προς ένα σύνολο εργασιών T με μέτρο απόδοσης P αν η απόδοση του στις εργασίες του T εκτιμώμενες από το μέτρο P βελτιώνεται με την εμπειρία E.

Με κριτήριο το πώς παράγεται γνώση για τη μηχανή-πρόγραμμα διακρίνουμε την επαγωγική (inductive learning) και την παραγωγική μάθηση (deductive learning). Στην παραγωγική μάθηση γίνεται ευρεία χρήση του λογικού κανόνα modus ponens $((p \rightarrow q) \wedge p) \rightarrow q$ που σημαίνει ότι αν ισχύει το p και ο κανόνας «αν p τότε q» τότε προκύπτει λογικά και το q. Ένα παράδειγμα παραγωγικής μάθησης είναι το εξής :

1)Όλοι οι άνθρωποι είναι θνητοί ή ισοδύναμα αν κάποιος είναι άνθρωπος τότε είναι θνητός.

2)Ο Σωκράτης είναι άνθρωπος.

Άρα συμπεραίνουμε ότι ο Σωκράτης είναι θνητός.

Αντίθετα με την παραγωγική μάθηση που το αποτέλεσμα είναι βέβαιο ότι έχει λογική ισχύ, στην επαγωγική το σύστημα προσπαθεί μέσα από ένα σύνολο παραδειγμάτων να υιοθετήσει εκείνη την υπόθεση που θα του επιτρέπει να κάνει καλές προβλέψεις σε παραδείγματα που

δεν έχει συναντήσει. Αυτή τη γενίκευση που κάνει το σύστημα μπορούμε να την δούμε σαν μία αναζήτηση σε ένα χώρο υποθέσεων. Ας δούμε ένα παράδειγμα επαγωγικής μάθησης κατά το οποίο παρουσιάζονται στο πρόγραμμα τα εξής παραδείγματα εκπαίδευσης: «Η κοκαΐνη σκοτώνει», «Η ηρωίνη σκοτώνει», «Το όπιο σκοτώνει» . Τότε μία υπόθεση που θα μπορούσε να υιοθετήσει το σύστημα είναι «Τα ναρκωτικά σκοτώνουν» υπονοώντας βέβαια ότι το σύστημα έχει και μία βάση γνώσης από την οποία εντοπίζει ότι η κοκαΐνη, ηρωίνη και το όπιο είναι ναρκωτικά. Σκοπός της επαγωγικής μάθησης είναι η παραγωγή μίας υπόθεσης η οποία θα μπορεί να αποφασίσει και για παραδείγματα τα οποία δεν έχει ξαναδεί το σύστημα. Γι αυτό το λόγο μία υπόθεση της μορφής «η ηρωίνη ή η κοκαΐνη ή το όπιο σκοτώνουν» δεν είναι αποδεκτή . Έτσι εισάγεται η έννοια του inductive bias (επαγωγική κλίση) δηλαδή κάποιων επιπρόσθετων υποθέσεων που κάνει το σύστημα ώστε να αποφύγει τέτοιου είδους γενικεύσεις όπως την παραπάνω. Μία τέτοια υπόθεση στο παράδειγμά μας είναι ότι η επαγωγική μας υπόθεση δεν θα πρέπει να είναι διάζευξη των παραδειγμάτων εκπαίδευσης.

Οι αλγόριθμοι της μηχανικής μάθησης διακρίνονται σε δύο βασικές κατηγορίες μάθησης, στην επιβλεπόμενη (supervised learning) και στην μη επιβλεπόμενη μάθηση (unsupervised learning).Στο πρώτο είδος μάθησης, παρουσιάζεται στη μηχανή-πρόγραμμα ένα σύνολο παραδειγμάτων εκπαίδευσης της μορφής <αντικείμενο εισόδου, αντικείμενο εξόδου> όπου η είσοδος συνήθως είναι ένα διάνυσμα τιμών $<x_1,x_2,\ldots,x_n>$ και η έξοδος μία τιμή y. Το πρόγραμμα εκπαιδεύεται μαθαίνοντας μία συνάρτηση η οποία εφαρμόζεται σε άγνωστα δεδομένα εισόδου, προβλέποντας μία τιμή εξόδου. Τυπικό παράδειγμα επιβλεπόμενης μάθησης είναι η ταξινόμηση (classification) στην οποία η τιμή εξόδου είναι διακριτή τιμή και αντιπροσωπεύει μία ομάδα-κλάση. Έτσι στα προβλήματα ταξινόμησης, όταν έχει εκπαιδευτεί το πρόγραμμα, μπορεί και κατατάσσει αντικείμενα εισόδου σε κατηγορίες.

Αντίθετα, στην μάθηση χωρίς επίβλεψη δεν υπάρχει προκαθορισμένο σύνολο τιμών εξόδου. Το σύστημα δέχεται κατά το στάδιο εκπαίδευσης ένα σύνολο από αντικείμενα εισόδου χωρίς αντίστοιχες τιμές εξόδου και προσπαθεί εξετάζοντας την δομή τους να βρει μία εσωτερική δομή στα δεδομένα αυτά.

Η μηχανική μάθηση έχει βρει πολλές σημαντικές πρακτικές εφαρμογές όπως αναγνώριση ομιλίας (speech recognition), αναγνώριση χειρόγραφων χαρακτήρων (handwritten character recognition), αυτόματη αναγνώριση ηλεκτρονικών απατών (fraud detection), αυτόματη εκμάθηση τεχνικών διαφόρων παιχνιδιών όπως το τάβλι και το σκάκι καθώς και άλλες πολλές. Αναμένεται τα επόμενα χρόνια να αναπτυχθεί περαιτέρω επιφέροντας σημαντικές επιπτώσεις στην κοινωνία. Ενδεικτικά παραδείγματα εφαρμογών είναι τα εξής ([AI06]):

- Ο διαχωρισμός του αργού πετρέλαιο από το φυσικό αέριο με το οποίο είναι αναμειγμένο κατά την εξόρυξή του από το έδαφος γίνεται με μια σύνθετη διαδικασία που εξαρτάται από διάφορες κρίσιμες παραμέτρους οι τιμές των οποίων

καθορίζονται από εξειδικευμένο προσωπικό. Η διαδικασία εύρεσης των ιδανικών τιμών διαρκεί περίπου μία ημέρα. Η γνωστή εταιρεία British Petroleum (BP) ανέπτυξε προγράμματα βασιζόμενα σε αλγόριθμους μηχανικής μάθησης τα οποία καθόριζαν τις τιμές των κρίσιμων παραμέτρων της διαδικασίας διαχωρισμού σε 10 λεπτά! Με αυτό το παράδειγμα καταδεικνύονται οι παρούσες και οι μελλοντικές επιδράσεις της μηχανικής μάθησης στην οικονομία.

- Glimmer είναι ένα πρόγραμμα που βασίζεται στην μηχανική μάθηση το οποίο με μεγάλη ακρίβεια αναγνωρίζει γονίδια μέσα σε ένα γονιδίωμα. Γίνεται φανερό ότι η μηχανική μάθηση μπορεί στο μέλλον να έχει και καθοριστικές συνέπειες στην ιατρική έρευνα .

Ένας άλλος τομέας στον οποίο οι τεχνικές μηχανικής μάθησης έχουν βρει εφαρμογή με επιτυχία και που σχετίζεται άμεσα με την εργασία αυτή είναι η αυτόματη εξαγωγή πληροφορίας από τον ιστό όπως θα δούμε στο κεφάλαιο 3. Τέλος , ένας σημαντικός κλάδος την μηχανικής μάθησης είναι ο συμπερασμός γραμματικών (grammatical inference). Ο συμπερασμός γραμματικών αναφέρεται στην εκμάθηση μίας γραμματικής από ένα σύνολο προτάσεων. Αυτές οι προτάσεις που αποτελούν παραδείγματα εκπαίδευσης μπορεί να είναι και αρνητικά παραδείγματα , δηλαδή προτάσεις οι οποίες δεν ακολουθούν την γραμματική που θέλουμε.

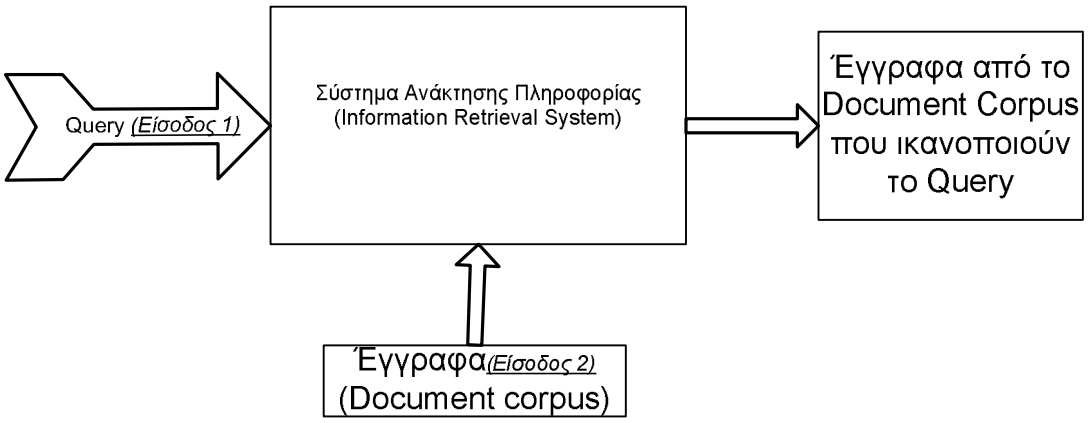
2.2 Ανάκτηση πληροφορίας (Information Retrieval) και εξαγωγή πληροφορίας (Information Extraction)

2.2.1 Κατηγορίες εγγράφων (documents) με κριτήριο τη δομή τους

Γενικά διακρίνουμε τρεις κατηγορίες εγγράφων με κριτήριο την ύπαρξη δομής στα περιεχόμενά τους : τα δομημένα, τα ημιδομημένα και τα μη δομημένα έγγραφα. Τα δομημένα έγγραφα μπορούμε να τα δούμε και σαν μία σχεσιακή βάση δεδομένων στην οποία έχουμε ένα πλήθος από εγγραφές (tuples) με συγκεκριμένο πλήθος attributes (γνωρίσματα). Συνεπώς κάθε συστατικό μίας εγγραφής έχει και μία συγκεκριμένη σημασιολογία που εκφράζεται από το attribute το οποίο αντιπροσωπεύει. Τα δεδομένα ακολουθούν μία προκαθορισμένη γραμματική (schema) με τρόπο που μπορεί να γίνει εύκολα αντιληπτός από μία μηχανή αναζήτησης. Τα μη δομημένα έγγραφα αντίθετα είναι σε μορφή ελεύθερου κειμένου (free text). Τέλος τα ημιδομημένα έγγραφα είναι σε μορφή κειμένου αλλά περιέχουν δεδομένα που ακολουθούν μία γραμματική. Χαρακτηριστικό παράδειγμα της κατηγορίας αυτής είναι οι ιστοσελίδες που δημιουργούνται με τον τρόπο που εκφράζει το σχήμα 1.1.

2.2.2 Ανάκτηση Πληροφορίας (Information Retrieval)

Καθώς η ποσότητα των εγγράφων και της πληροφορίας αυξάνεται συνεχώς, η εξεύρεση των εγγράφων που επιθυμεί ένα χρήστης δυσχεραίνεται σημαντικά. Για το σκοπό αυτό κατασκευάζονται τα συστήματα ανάκτησης πληροφορίας. Αν και ο όρος είναι πολύ γενικός, στην πληροφορική ένα σύστημα ανάκτησης πληροφορίας (information retrieval system) έχει ως στόχο την επιλογή και προβολή εγγράφων μέσα από ένα μεγάλο σύνολο εγγράφων σύμφωνα με τα κριτήρια που έχει θέσει ο χρήστης. Συνήθως με τον όρο αυτόν, εννοούμε την ανάκτηση ενός εγγράφου σχετικού με τα κριτήρια που έχουμε θέσει εκμεταλλευόμενοι το μη δομημένο περιεχόμενο του εγγράφου. Ένα τέτοιο σύστημα μακροσκοπικά εμφανίζεται όπως το παρακάτω σχήμα:



σχήμα 2.1:Μακροσκοπική όψη ενός συστήματος ανάκτησης πληροφορίας

Το σύστημα ανάκτησης πληροφορίας δέχεται δύο εισόδους, ένα σύνολο εγγράφων (Document corpus) -συνήθως πολύ μεγάλο σε μέγεθος- και μία ερώτηση από το χρήστη -συνήθως σε φυσική γλώσσα- η οποία εκφράζει το τι είδους έγγραφα επιθυμεί. Τότε το σύστημα αναζητεί έγγραφα που θεωρεί ότι θα ενδιαφέρουν το χρήστη μέσα στο Document corpus. Υπάρχουν διάφορες τεχνικές που εφαρμόζονται κατά τη διαδικασία αυτή. Ενδεικτικά αναφέρουμε την κατηγοριοποίηση κειμένου (text classification) και την αναζήτηση λέξεων κλειδιών (keyword searching).

2.2.3 Εξαγωγή Πληροφορίας (Information Extraction)

Αν και η εξαγωγή πληροφορίας είναι στενά συνδεόμενη σαν έννοια με την ανάκτηση πληροφορίας είναι κάτι το διαφορετικό. Στην ανάκτηση πληροφορίας παρουσιάζεται στο χρήστη ένα σύνολο από έγγραφα τα οποία θεωρούνται ότι ταιριάζουν με τα κριτήρια που έχει θέσει. Αντίθετα στην εξαγωγή πληροφορίας παρουσιάζεται στο χρήστη συγκεκριμένα η πληροφορία που τον ενδιαφέρει από τα έγγραφα.

Ένα σύστημα εξαγωγής πληροφορίας (Ι.Ε σύστημα εκ των λέξεων Information Extraction) λειτουργεί εφαρμόζοντας ένα σύνολο κανόνων που είτε παράγεται αυτόματα είτε κατασκευάζεται χειρωνακτικά με σκοπό την εξαγωγή πληροφορίας από οποιουδήποτε τύπου

έγγραφα. Η λογική με την οποία λειτουργεί το σύστημα καθορίζεται σε πρώτο επίπεδο από το είδος των εγγράφων από τα οποία θα εξαχθεί η πληροφορία. Έτσι διακρίνουμε τα συστήματα εξαγωγής πληροφορίας από μη δομημένα, από δομημένα και από ημιδομημένα έγγραφα Επειδή στα δομημένα έγγραφα (π.χ XML, βάσεις δεδομένων) είναι σχετικά εύκολο να εξαχθεί η πληροφορία που επιθυμεί ο χρήστης αφού είναι γνωστή η σημασιολογία της πληροφορίας, συμπεραίνουμε ότι τα πιο δύσκολα προβλήματα κατασκευής ΙΕ συστημάτων σχετίζονται με τα μη δομημένα και τα ημιδομημένα έγγραφα. Έτσι μπορεί να θεωρηθεί πως δύο είναι οι λογικές κατασκευής ενός I.E συστήματος: αυτή του ελευθέρου κειμένου και του ημιδομημένου κειμένου. Η πρώτη λογική βασίζεται στη χρήση ενός λεκτικού και ενός συντακτικού αναλυτή καθώς και ενός συστήματος που ονομάζεται semantic tagger (σημασιολογητής θα μπορούσε να είναι μία μετάφραση αυτού του όρου στα ελληνικά), δηλαδή ενός προγράμματος το οποίο αποδίδει σημασιολογία στις λέξεις του κειμένου. Μερικά από τα πιο γνωστά I.E συστήματα για ελεύθερο κείμενο είναι το AutoSlog, το LIEP, το HASTEN, το PALKA, το WHISK και το CRYSTAL. Η λογική κατασκευής ενός I.E συστήματος για την περίπτωση που μας ενδιαφέρει στα πλαίσια αυτής της εργασίας, δηλαδή για ημιδομημένο κείμενο προερχόμενο από ιστοσελίδες, γραμμένο σε γλώσσα HTML, βασίζεται στο layout της ιστοσελίδας το οποίο εμφανίζει μία κανονικότητα ως προς τα html tags τα οποία εμπερικλείουν την πληροφορία που συνήθως θέλουμε να εξάγουμε. Περισσότερα για ΙΕ από ημιδομημένο κείμενο στο κεφάλαιο 3^{ο} όπου μελετάμε το Wrapper Induction (παραγωγή wrappers με χρήση επαγωγικής μάθησης)

3

Wrapper Induction και Wrapper Maintenance

3.1 Wrapper Induction (Παραγωγή wrappers με χρήση επαγωγικής μάθησης)

3.1.1 Εισαγωγικά

Όπως αναφέρθηκε και στο εισαγωγικό κεφάλαιο οι wrappers είναι προγράμματα τα οποία με βάση ένα σύνολο κανόνων εξάγουν δεδομένα από μία ιστοσελίδα και τα φέρνουν σε μία δομημένη μορφή . Οι κύριες λειτουργίες που πραγματοποιεί ένας wrapper είναι οι εξής :

- Κατεβάζει ιστοσελίδες HTML (web pages) από ένα web site (διαδικτυακό τόπο).
- Με βάση το σύνολο κανόνων που έχει, προσδιορίζει την πληροφορία που θα εξάγει.
- Αποθηκεύει την πληροφορία αυτή σε μία δομημένη μορφή που μπορεί να επεξεργαστεί περαιτέρω από κάποιο άλλο πρόγραμμα. Μία συνήθης τέτοια μορφή είναι η XML.

Ένας τρόπος για να κατασκευάσουμε έναν wrapper για τις ιστοσελίδες ενός site είναι να γράψουμε ένα πρόγραμμα σε κάποια γλώσσα προγραμματισμού (π.χ JAVA, Perl) χρησιμοποιώντας κανονικές εκφράσεις. Αυτό όμως είναι μία διαδικασία που αφενός απαιτεί προγραμματιστικές γνώσεις από το χρήστη, αφ' ετέρου είναι αρκετά δύσκολο, χρονοβόρο και εμπεριέχει πιθανότητα σφάλματος. Αυτά τα μειονεκτήματα δημιούργησαν την ανάγκη για έναν εναλλακτικό τρόπο παραγωγής wrappers ο οποίος θα προσπαθούσε να ελαχιστοποιήσει την ανθρώπινη ανάμειξη. Έτσι εφαρμόστηκαν τεχνικές μηχανικής μάθησης για την ευκολότερη παραγωγή wrappers. Έτσι αναπτύχθηκε ο αυτόματος (automatic wrapper induction systems) και ο ημιαυτόματος τρόπος (semiautomatic wrapper induction systems).

Τα αυτόματα συστήματα παραγωγής wrappers εκμεταλλεύονται αυτό που φαίνεται στο σχήμα 1.1 και ισχύει για πολλές ιστοσελίδες, διαχωρίζοντας το template από την πληροφορία. Αντίθετα στα ημιαυτόματα συστήματα υπάρχει ανθρώπινη ανάμειξη. Ο χρήστης παρέχει στο σύστημα ένα σύνολο ιστοσελίδων μαζί με παραδείγματα της πληροφορίας που τον ενδιαφέρει και το σύστημα με βάση τα παραδείγματα εκπαιδεύεται ώστε να μπορεί να εξάγει από αντίστοιχες ιστοσελίδες την επιθυμητή πληροφορία (στο σημείο αυτό γίνεται σιωπηρά η παραδοχή ότι οι ιστοσελίδες ενός web site έχουν κοινό layout, δηλαδή η πληροφορία εμφανίζεται με ίδιο τρόπο στον χρήστη. Στην πραγματικότητα αυτή η παραδοχή ισχύει σε μεγάλο βαθμό ώστε να γίνεται σιωπηρά και να αποδίδει και στη πράξη).

Παρακάτω αναλύουμε δύο σημαντικές εργασίες. Η πρώτη περιγράφει την λειτουργία ενός αυτόματου συστήματος εξαγωγής πληροφορίας και η δεύτερη ενός ημιαυτόματου.

3.1.2 EXALG: Αυτόματη εξαγωγή δομημένης πληροφορίας από ιστοσελίδες

Η πλήρης αυτοματοποίηση της εξαγωγής πληροφορίας είναι επιθυμητή γιατί η ανθρώπινη ανάμειξη (human-input) είναι χρονοβόρος και επιρρεπής σε λάθη. Η κατασκευή ενός αυτόματου συστήματος είναι δύσκολη γιατί αφ'ενός δεν υπάρχει κάποιος προφανής τρόπος να διακρίνουμε αν το κείμενο είναι μέρος του template ή της πληροφορίας που προέρχεται από την βάση δεδομένων και αφ'ετέρου μπορεί σε κάποια εγγραφή για ένα attribute να μην υπάρχει κάποια τιμή ή να έχουμε περισσότερες από μία τιμές. Ο EXALG ([AG02]) παίρνει στην είσοδό του ένα σύνολο ιστοσελίδων που περιέχουν δομημένα δεδομένα (structured data) δηλαδή ένα σύνολο από τιμές δεδομένων (data values) το οποίο ακολουθεί ένα τύπο δεδομένων. Ένας τύπος δεδομένων ορίζεται επαγωγικά ως εξής:

ΤΥΠΟΣ ΔΕΔΟΜΕΝΩΝ

1.Ο βασικός τύπος Β αναπαριστά ένα string (συμβολοσειρά) από tokens (λεκτικές μονάδες) όπου το token μπορεί να είναι μία λέξη ή ένα HTML tag.

2.Αν $T_1,..,T_n$ τύποι τότε και $<T_1,..,T_n>$ τύπος. Λέμε ότι ο τύπος $<T_1,..,T_n>$ κατασκευάζεται από τους τύπους $T_1,..,T_n$ χρησιμοποιώντας έναν tuple constructor τάξης n (κατασκευαστής εγγραφής τάξης n).

3.Αν T_1 τύπος, τότε και $\{T_1\}$ τύπος. Λέμε ότι ο τύπος $\{T_1\}$ κατασκευάζεται από το τύπο T_1 χρησιμοποιώντας έναν set constructor (κατασκευαστής συνόλου).

Ακολουθεί ο ορισμός του στιγμιότυπου ενός τύπου που επίσης είναι επαγωγικός.

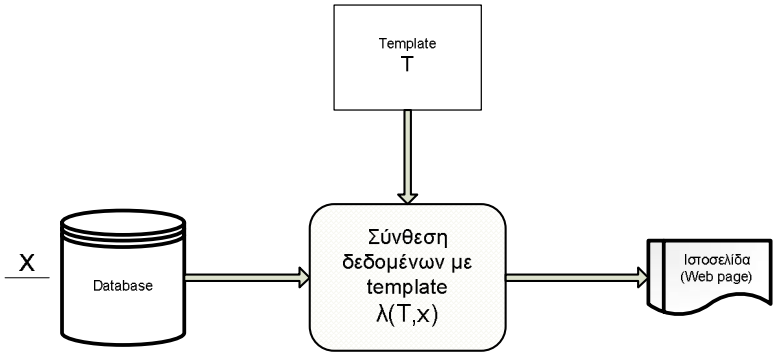
ΣΤΙΓΜΙΟΤΥΠΟ ΤΥΠΟΥ ΔΕΔΟΜΕΝΩΝ

1.Στιγμιότυπο ενός βασικού τύπου είναι οποιοδήποτε string από tokens.

2.Στιγμιότυπο του $<T_1,..,T_n>$ είναι οποιαδήποτε εγγραφή $<i_1,..,i_n>$ όπου $i_1,..,i_n$ στιγμιότυπα των $T_1,..,T_n$ αντίστοιχα .

3. Στιγμιότυπο του {T} είναι οποιοδήποτε σύνολο στοιχείων από $\{e_1,, e_n\}$ όπου e_i $(1 \le i \le n)$ στιγμιότυπο του T.

Στο παρακάτω σχήμα (ίδιο με το 1.1, για ευκολία το ξαναβλέπουμε και εδώ) φαίνεται ότι η δημιουργία μίας τυπικής ιστοσελίδας συνίσταται στο να πάρουμε την πληροφορία x από μια βάση δεδομένων (database) και κάνοντας χρήση ενός template να κάνουμε κωδικοποίηση (encoding) λ(T,x).



σχήμα 3.1 : Τρόπος δημιουργίας ιστοσελίδας με βάση ένα template T

Παρακάτω ακολουθούν οι ορισμοί που αφορούν την παραπάνω διαδικασία :

TEMPLATE

Ένα template T για ένα τύπο δεδομένων S ορίζεται σαν μία συνάρτηση που απεικονίζει τον κάθε constructor τ του S σε ένα διατεταγμένο σύνολο από strings T(τ). Συγκεκριμένα:

1.Αν τ ένας tuple constructor τάξης n, τότε το template T(τ) είναι το διατεταγμένο σύνολο από strings $< C_{\tau(1)}, C_{\tau(2)}, .. C_{\tau(n+1)} >$.

2. Αν τ ένας set constructor, τότε το template T(τ) είναι ένα string.

Για να δείξουμε ότι το template T αναφέρεται σε ένα schema S γράφουμε T^S.

ΚΩΔΙΚΟΠΟΙΗΣΗ λ(T,x) (Encoding)

1.Αν x είναι βασικού τύπου τότε λ(T, x)= x.

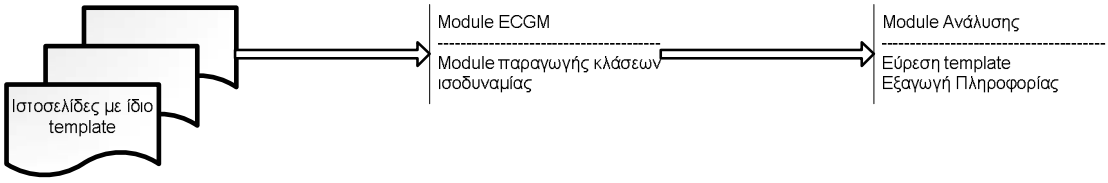
2.Αν x είναι της μορφής $<x_1,x_2,\ldots,x_n>_\tau$,τότε λ(T,x) είναι το string $C_1\lambda(T,x_1)C_2\lambda(T,x_2)...\lambda(T,x_n)C_{n+1}$ όπου $T(\tau) =< C_1,..,C_{n+1} >$

3.Αν το x είναι της μορφής $\{e_1,e_2,\ldots,e_n\}_\tau$, τότε λ(T, x)=$\lambda(T,e_1)S\lambda(T,e_2)...S\lambda(T,e_m)$ όπου $T(\tau) = S$.

Με βάση τους παραπάνω ορισμούς το πρόβλημα που έχει να αντιμετωπίσει ένα αυτόματο σύστημα εξαγωγής πληροφορίας είναι το εξής :

Δοθέντος ενός συνόλου από n σελίδες , $p_i = \lambda(T, x_i)$ $1 \le i \le n$ που δημιουργήθηκαν από κάποιο άγνωστο template T και τιμές $\{x_1,.., x_n\}$ να βρεθεί το T και οι τιμές $\{x_1,.., x_n\}$.

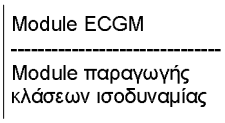
Ο αλγόριθμος EXALG δουλεύει σε δύο στάδια όπως φαίνεται από το σχήμα:

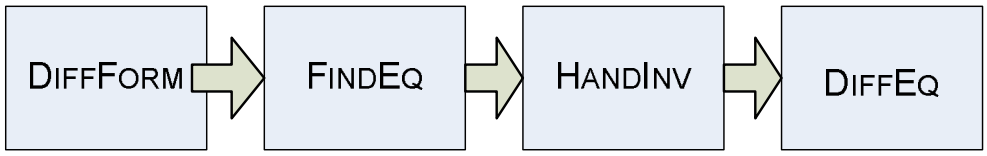


σχήμα 3.2 : Στάδια λειτουργίας αλγόριθμου EXALG

Στο πρώτο στάδιο γίνεται κατασκευή των κλάσεων ισοδυναμίας. Η κλάση ισοδυναμίας περιέχει tokens τα οποία έχουν το ίδιο διάνυσμα εμφάνισης $< f_1,.., f_n >$ όπου f_i είναι το πλήθος εμφανίσεων του token στην σελίδα p_i. Ο λόγος που στην πράξη ενδιαφερόμαστε για την εύρεση των κλάσεων ισοδυναμίας ενός συνόλου ιστοσελίδων $\{p_1,.., p_n\}$ είναι ότι τα tokens που ανήκουν στην ίδια κλάση σχετίζονται με τον ίδιο constructor τ στο σχηματισμό του template.

Στο επόμενο σχήμα βλέπουμε τα επιμέρους τμήματα του πρώτου σταδίου τα οποία και περιγράφονται:





σχήμα 3.3α : 1ο στάδιο λειτουργίας αλγόριθμου EXALG

Στο module **DiffForm** προσδιορίζονται τα dtokens (η ονομασία αυτή προέρχεται από τις λέξεις differentiated tokens που σημαίνει διαφοροποιημένα tokens) δηλαδή tokens που αν και είναι ίδια έχουν διαφορετική σημασιολογία μέσα στην ιστοσελίδα . Για να προσδιοριστεί αν μία εμφάνιση ενός token έχει τον ίδιο ρόλο με μία άλλη εμφάνιση του ίδιου token, κατασκευάζεται το DOM tree της ιστοσελίδας ώστε να προσδιοριστούν τα μονοπάτια από τη ρίζα του δέντρου μέχρι τις δύο εμφανίσεις του token. Αν τα μονοπάτια είναι διαφορετικά συμπεραίνουμε ότι δεν έχουν τον ίδιο ρόλο. Αν τα μονοπάτια είναι ίδια τότε κοιτάμε και τα γειτονικά tokens. Αν είναι διαφορετικά τότε πάλι συμπεραίνουμε ότι δεν έχουν τον ίδιο ρόλο αλλιώς καταλήγουμε στο ότι οι δύο εμφανίσεις έχουν την ίδια σημασιολογία.

Στο module **FindEq** προσδιορίζονται οι κλάσεις ισοδυναμίας και τα διανύσματα εμφάνισης των dtokens. Ο αλγόριθμος διατηρεί όχι όλες τις κλάσεις ισοδυναμίας που προκύπτουν αλλά τις LFEQ δηλαδή αυτές που έχουν μεγάλο πλήθος tokens και που εμφανίζονται σε πολλές σελίδες. Στο module **HandInv**, απορρίπτονται κλάσεις ισοδυναμίας που δεν πληρούν τις ιδιότητες της διάταξης και του φωλιάσματος. Μία κλάση ισοδυναμίας είναι διατεταγμένη αν τα tokens της μπορούν να διαταχθούν σε μία σειρά $< t_1,.., t_m >$ έτσι ώστε για κάθε σελίδα

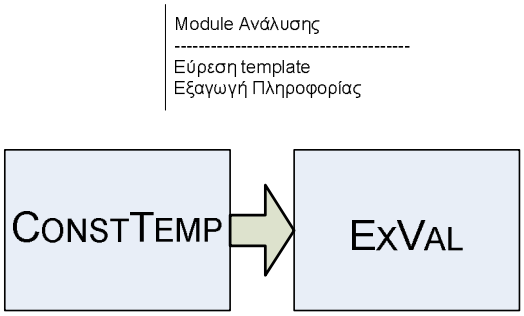
p_i $1 \le i \le n$ και κάθε ζευγάρι tokens t_j, t_k $1 \le j \prec k \le m$ αν το t_j εμφανίζεται τουλάχιστον l φορές στην σελίδα p_i τότε η l-οστή εμφάνιση του t_j στο p_i προηγείται από την l-οστή εμφάνιση του t_k και αν το t_j εμφανίζεται τουλάχιστον $l+1$ φορές στην σελίδα p_i τότε η $(l+1)$-οστή εμφάνιση του t_j στο p_i έπεται της l-οστής εμφάνιση του t_k. Ένα ζεύγος $\varepsilon_i, \varepsilon_j$ διατεταγμένων κλάσεων ισοδυναμίας είναι φωλιασμένο (nested) αν είτε η έκταση[1] (span) της οποιαδήποτε εμφάνισης της ε_i δεν επικαλύπτεται με την έκταση καμίας εμφάνισης της ε_j είτε η έκταση (span) όλων των εμφανίσεων της ε_i είναι εντός της ίδιας θέσης Pos(p)[1] κάποιας εμφάνισης της ε_j (ή το αντίστροφο).

Στο τελευταίο module του πρώτου σταδίου **DiffEq**, τα dtokens που προσδιορίστηκαν κατά το πρώτο βήμα εντάσσονται στην κατάλληλη κλάση ισοδυναμίας με βάση το διάνυσμα εμφάνισης τους.

Στο δεύτερο στάδιο γίνεται η εύρεση του template και του τύπου δεδομένων των σελίδων καθώς και η εξαγωγή των δεδομένων με βάση τις κλάσεις ισοδυναμίας που προέκυψαν από το πρώτο στάδιο. Το module **ConstTemp** παίρνει σαν είσοδο τις κλάσεις ισοδυναμίας που παράχθηκαν κατά το πρώτο στάδιο και τις σελίδες εισόδου, τις οποίες αντιμετωπίζει σαν μία λίστα από strings. Βασίζεται στην αναδρομή ως εξής: Αρχικά προσδιορίζει την κύρια κλάση ισοδυναμίας (root LFEQ) δηλαδή εκείνη την κλάση ισοδυναμίας της οποίας κάθε token εμφανίζεται μία μόνο φορά σε κάθε σελίδα. Στη συνέχεια αναζητά τις μη κενές θέσεις ανάμεσα σε διαδοχικά tokens της κλάσης αυτής. Κενή είναι μία θέση ανάμεσα σε δύο tokens όταν αυτά εμφανίζονται διαδοχικά, το ένα αμέσως μετά από το άλλο. Αν δε συμβαίνει αυτό τότε είναι μη κενή. Αν βρεθούν περισσότερες από μία μη κενές θέσεις ο αλγόριθμος παράγει έναν tuple constructor ενώ αν βρεθεί μόνο μία μη κενή παράγεται ένας set constructor. Για κάθε μη κενή θέση προσδιορίζεται ο κατάλληλος constructor. Αν εντός μίας μη κενής θέσης δεν εμφανίζεται κάποια άλλη κλάση ισοδυναμίας τότε θα εμφανίζεται ο βασικός τύπος δεδομένων. Αλλιώς συνεχίζεται η ίδια διαδικασία αναδρομικά μέχρι να καταλήξουμε σε βασικούς τύπους προσδιορίζοντας έτσι τον τύπο δεδομένων των ιστοσελίδων. Στη συνέχεια με βάση τον ορισμό του template και δοθέντος του τύπου δεδομένων S βρίσκει το template

---

[1]Έστω μία διατεταγμένη κλάση $\varepsilon = < t_1, .., t_m >$ και έστω ότι τα tokens της εμφανίζονται f φορές στη σελίδα p_i. Λέμε τότε ότι η ε εμφανίζεται f στη σελίδα p_i. Η ι-οστή εμφάνιση της ε αναφέρεται συλλογικά στην ι-οστή εμφάνιση των $t_1, .., t_m$. Η έκταση (span) της ι-οστής εμφάνισης της ε στην p_i έχει σαν αρχικό σημείο την ι-οστή εμφάνιση του t_1 και ως τέλος την ι-οστή εμφάνιση του t_m. Έτσι η έκταση (span) της ι-οστής εμφάνισης της ε υποδιαιρείται σε (m-1) θέσεις: Pos(1) μεταξύ t_1, t_2, .., Pos(m-1) μεταξύ t_{m-1}, t_m.



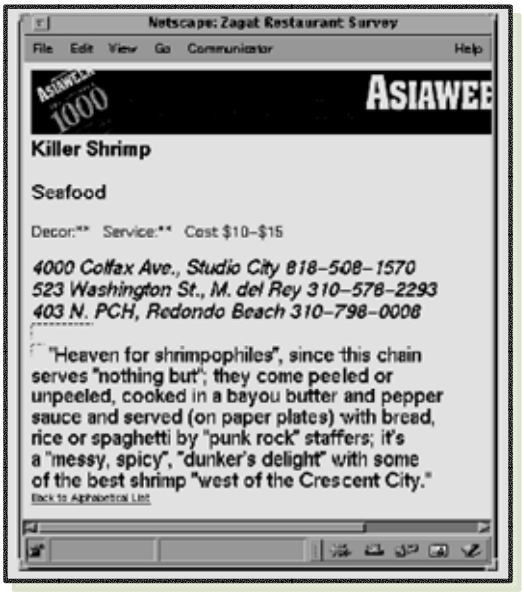
σχήμα 3.3β : 2ο στάδιο λειτουργίας αλγόριθμου EXALG

T^S. Τέλος ο αλγόριθμος μπαίνει στο τελευταίο module **ExVal** στο οποίο με βάση το S και το T^S εξάγει τις τιμές δεδομένων από όλες τις ιστοσελίδες εισόδου.

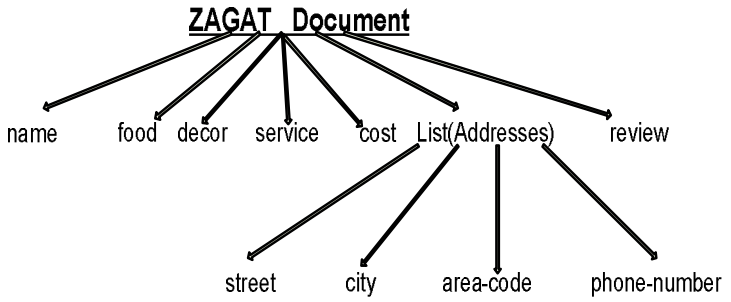
3.1.3 STALKER: Wrapper Induction Αλγόριθμος

Ο STALKER είναι ένας wrapper induction αλγόριθμος ([MMK01]). Οι παραγόμενοι wrappers βασίζονται σε κανόνες εξαγωγής (extraction rules) υψηλής ακρίβειας που παράγονται με βάση παραδείγματα που δίνει ο χρήστης.

Ο STALKER για να περιγράψει τη δομή μίας σελίδας χρησιμοποιεί το Embedded Catalog (EC). Πρόκειται για μια ιεραρχική δομή που μοιάζει με δενδρική. Τα φύλλα είναι τα αντικείμενα τα οποία μπορεί να ενδιαφέρουν τον χρήστη ενώ οι εσωτερικοί κόμβοι αναπαριστούν λίστες όπου κάθε αντικείμενο μέσα στην λίστα μπορεί να είναι είτε φύλλο είτε μία άλλη λίστα. Για να γίνει πιο κατανοητός ο παραπάνω ορισμός παραθέτουμε ένα παράδειγμα. Στο σχήμα 3.4α βλέπουμε μία ιστοσελίδα ενώ στο 3.4β το EC που της αντιστοιχεί.

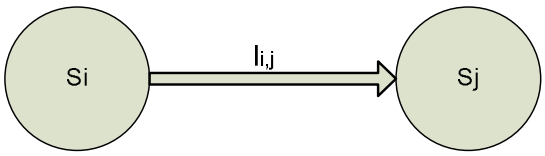


σχήμα 3.4α : Ιστοσελίδα με μία εγγραφή για ένα εστιατόριο



σχήμα 3.4β:Embedded Catalog για τις εγγραφές της ιστοσελίδας του σχ.3.4α

Η εξαγωγή δεδομένων γίνεται με τη χρήση του παραγόμενου wrapper ο οποίος γνωρίζει τη δομή της σελίδας και διαθέτει ένα σύνολο κανόνων. Ο wrapper χρειάζεται έναν κανόνα για την εξαγωγή ενός κόμβου (δηλαδή της πληροφορίας που περιέχει) από τον γονέα του. Επίσης για τους κόμβους που παριστάνουν λίστα χρειάζεται ένας επαναληπτικός κανόνας που να αποσυνθέτει την λίστα στις επιμέρους εγγραφές. Ο καθένας από τους κανόνες που χρησιμοποιεί είναι ένα γραμμικό landmark[1] αυτόματο. Πιο συγκεκριμένα landmark είναι μία ακολουθία από tokens και wildcards[2]. Landmark αυτόματο είναι ένα μη ντετερμινιστικό πεπερασμένο αυτόματο στο οποίο η μετάβαση από μία κατάσταση Si σε μία άλλη κατάσταση Sj χαρακτηρίζεται από ένα landmark $l_{i,j}$.όπως δείχνει και το παρακάτω σχήμα:



σχήμα 3.5:Απλό landmark αυτόματο

Αυτό σημαίνει ότι αν είμαστε στην κατάσταση Si πηγαίνουμε στη κατάσταση Sj αν και μόνο αν το landmark $l_{i,j}$ ταιριάζει με την είσοδο (input).Τα γραμμικά landmark αυτόματα είναι μία κλάση landmark αυτόματων τα οποία είναι χρήσιμα για δύο λόγους : έχουν την απαραίτητη εκφραστική δύναμη για να χειριστούμε το Embedded Catalog και επιπλέον είναι εύκολη η παραγωγή και η τροποποίησή τους όπως και θα δούμε παρακάτω. Χαρακτηρίζονται από τις εξής ιδιότητες :

- Έχουν μόνο μία τελική κατάσταση (accepting state).
- Από κάθε μη τελική κατάσταση υπάρχουν δύο ακριβώς μεταβάσεις. Μία προς την ίδια κατάσταση (loop) και μία προς μία επόμενη. Κάθε μετάβαση προς την επόμενη χαρακτηρίζεται από ένα landmark. Η φυσική σημασία αυτού είναι: «κατανάλωνε tokens εισόδου κάνοντας loops μέχρι να βρεθεί είσοδος που ταιριάζει με το landmark της μετάβασης και τότε πήγαινε στην επόμενη κατάσταση».

---

[1]Μία πιθανή μετάφραση του όρου landmark είναι : ορόσημο από λεκτικές μονάδες

[2]Τα wildcards αντιπροσωπεύουν κλάσεις χαρακτήρων. Π.χ η τελεία, το θαυμαστικό αντιπροσωπεύονται από το wildcard *Punctuation*. Το wildcard που αντιπροσωπεύει όλους τους τύπους είναι το *Anything*.

Στην πραγματικότητα η είσοδος που δέχεται ο STALKER είναι ένα σύνολο από tokens που αντιπροσωπεύουν τα προθέματα που πρέπει να καταναλωθούν από τον κανόνα εξαγωγής πληροφορίας για να φτάσουμε στην αρχή της επιθυμητής πληροφορίας. Αυτό γίνεται με «χειρωνακτικό» τρόπο και όχι αυτόματα, που σημαίνει ότι πρέπει ο χρήστης να σημειώσει με κάποιον τρόπο (π.χ Graphical User Interface,annotation file) την πληροφορία που τον ενδιαφέρει. Έπειτα ο STALKER προσπαθεί σε κάθε βήμα του να παράγει ένα αυτόματο το οποίο σε σχέση με αυτό του προηγούμενου βήματος θα καλύπτει περισσότερα παραδείγματα του χρήστη. Ας θεωρήσουμε ότι ο χρήστης θέλει να εξάγεται αυτόματα η πληροφορία του κωδικού περιοχής (area code) από κάποια ιστοσελίδα. Με βάση τη διαπροσωπεία που του παρέχεται σημειώνει στα τέσσερα παραδείγματα E1-E4 τη πληροφορία που είναι με έντονη και υπογραμμισμένη γραμματοσειρά.

E1 : 513 Pico, Venice,Phone: 1-**800**-555-1515

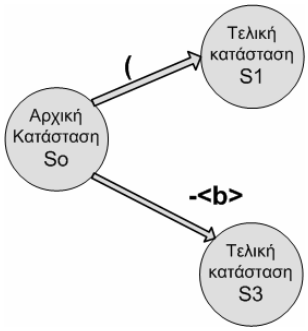
E2 : 90 Colfax Palms,Phone: (**818**)508-1570

E3 : 523 1st St.,LA,Phone: 1-**888**-578-2293

E4 : 403 La Tijera Watts,Phone: (**310**)798-0008

σχήμα 3.6:Παράδειγμα εισόδου με 4 παραδείγματα για τον STALKER

Στη συνέχεια εκτελείται ο STALKER με είσοδο αυτά τα τέσσερα παραδείγματα. Αρχικά κατασκευάζει ένα γραμμικό landmark αυτόματο το οποίο καλύπτει όσο δυνατόν περισσότερα παραδείγματα. Αν μένουν κάποια που δεν ικανοποιούνται τότε παράγει με την ίδια λογική κάποιο άλλο αυτόματο που προσπαθεί να καλύψει τα εναπομείναντα ακάλυπτα παραδείγματα. Όταν έχουν καλυφθεί όλα τα παραδείγματα ο αλγόριθμος επιστρέφει την διάζευξη των αυτομάτων που παρήγαγε. Στο παραπάνω παράδειγμα παράγει ο STALKER τον πρώτο κανόνα $R1 = SkipTo(\ (\ )$ ο οποίος σημαίνει προσπέρασε όλα τα tokens μέχρι να βρεις την πρώτη παρένθεση και καλύπτει τα παραδείγματα E2,E4 αλλά αποτυγχάνει για τα παραδείγματα E1,E3. Στη συνέχεια για να καλύψει και τα παραδείγματα E1,E3 παράγει τον κανόνα $R2 = SkipTo(-<b>)$.Έτσι το αυτόματο που παράγεται είναι το παρακάτω:



σχήμα 3.7:Γραμμικό landmark αυτόματο που επιστρέφει ο STALKER για το παράδειγμα του σχήματος

Πιο τυπικά ας θεωρήσουμε ότι στόχος μας είναι να εξάγουμε ένα αντικείμενο x από το γονέα p.Το input είναι ένα σύνολο από παραδείγματα της μορφής $[T_i, Idx_i]$ όπου T_i είναι ένα στιγμιότυπο του γονέα p και $T_i[Idx_i]$ είναι το token που αναπαριστά την αρχή του x μέσα στο p. Ο αλγόριθμος λαμβάνει κάθε πρόθεμα του p ως προς x $Prefix_x(p)$ σαν θετικό παράδειγμα. Ουσιαστικά το $Prefix_x(p)$ ισοδυναμεί με την ακολουθία $S = T_i[1], T_i[2], ..., T_i[Idx_i - 1]$.Κάθε υποακολουθία ή υπερακολουθία της S θεωρείται αρνητικό παράδειγμα. Ακολουθεί ο ψευδοκώδικας του STALKER καθώς και επεξηγήσεις για το τι κάνει η κάθε μέθοδος:

LearnRule(Examples)
Έστω RetVal ένα κενό σύνολο κανόνων
while $Examples \neq \varnothing$
aDisjunct=**LearnDisjunct**(Examples)
από το *Examples* αφαίρεσε όλα τα παραδείγματα που καλύπτονται από το aDisjunct
RetVal =RetVal $\cup$ aDisjunct
return **OrderDisjuncts**(RetVal)

****

LearnDisjunct(Examples)
Έστω Seed $\in$ Examples το συντομότερο παράδειγμα
Candidates=GetInitialCandidates(Seed)
DO
BestRefiner=GetBestRefiner(Candidates)
BestSolution=GetBestSolution(Candidates $\cup$ {BestSolution})
Candidates= **Refine** (BestRefiner,Seed)
WHILE isNotPerfect(BestSolution) and BestRefiner $\neq \varnothing$
return PostProcess(BestSolution)

****

Refine(C,Seed)
Έστω ότι το C αποτελείται από συνεχόμενα landmarks $l_1, l_2, .., l_n$
Έστω TopologyRefs=LandmarkRefs=$\varnothing$
FOR i=1 TO n DO
Έστω m το πλήθος των tokens στο l_i
FOR EACH token t στο Seed DO
σε ένα αντίγραφο Q του C πρόσθεσε το token-landmark t μεταξύ l_i και l_{i+1}

και φτιάξε και ένα επιπλέον κανόνα βάζοντας στη θέση t το wildcard που το αντιπροσωπεύει

Πρόσθεσε τους κανόνες αυτούς στο TopologyRefs

FOR EACH ακολουθία $s = t_0 t_1 .. t_{m+1}$ στο Seed DO

if matches(l_i,s) then

έστω P=Q=C

αντικατάστησε στο P το l_i με $t_0 l_i$

αντικατάστησε στο Q το l_i με $l_i t_{m+1}$

δημιούργησε κανόνες θέτοντας κατ' αντιστοιχία στις θέσεις των t_0, t_{m+1} τα wildcards που τα αντιπροσωπεύουν

Πρόσθεσε τους κανόνες αυτούς στο LandmarkRefs

return PostProcess(TopologyRefs ∪ LandmarkRefs)

LearnRule()

Ο αλγόριθμος προσπαθεί να βρει μία διάζευξη από γραμμικά landmark αυτόματα έτσι ώστε να καλύπτονται όλα τα παραδείγματα εισόδου. Η **OrderDisjuncts()** διατάσσει τις διάφορες παραγόμενες διαζεύξεις με βάση το κριτήριο πόσες φορές αποτυγχάνει η διάζευξη να αναγνωρίσει σωστά το αντικείμενο x μέσα στον γονέα του p. Επιλέγεται αυτή που αποτυγχάνει τις λιγότερες φορές. Σε περίπτωση ισοπαλίας μεταξύ δύο διαζεύξεων ως προς το πρώτο κριτήριο, επιλέγεται εκείνη που φορές επιτυγχάνει να αναγνωρίσει σωστά το x σε περισσότερα παραδείγματα. Αυτή η διατεταγμένη λίστα από διαζεύξεις επιστρέφεται σαν έξοδος του αλγόριθμου.

GetInitialCandidates()

Η συνάρτηση αυτή παράγει έναν αρχικό υποψήφιο για την τέλεια διάζευξη που θα δώσει ο αλγόριθμος. Σαν όρισμα λαμβάνει το μικρότερο από τα παραδείγματα που δίνονται στον αλγόριθμο σαν είσοδος, δηλαδή εκείνο το παράδειγμα με το μικρότερο πλήθος tokens στο πρόθεμα $\text{Prefix}_x(p)$. Ο υποψήφιος αυτός είναι ένα γραμμικό landmark αυτόματο με δύο καταστάσεις στο οποίο η μετάβαση από την πρώτη στην δεύτερη κατάσταση γίνεται με ένα landmark το οποίο είναι είτε το token που βρίσκεται στο τέλος του προθέματος $\text{Prefix}_x(p)$ του ορίσματος της συνάρτησης είτε ένα wildcard το οποίο αντιπροσωπεύει την κλάση του token αυτού.

GetBestRefiner()

Η συνάρτηση αυτή επιλέγει από το σύνολο των υποψηφίων μία διάζευξη. Η διάζευξη που επιλέγεται είναι αυτή με τις περισσότερες επιτυχίες. Σε περίπτωση ισοβαθμίας δύο διαζεύξεων ως προς τις επιτυχίες, επιλέγεται εκείνη που τις περισσότερες φορές που

αποτυγχάνει προσπερνάει tokens χωρίς να προσπεράσει την επιθυμητή πληροφορία. Αν και πάλι υπάρχει ισοβαθμία τότε επιλέγεται ο πιο συγκεκριμένος-εξειδικευμένος κανόνας.

LearnDisjunct()

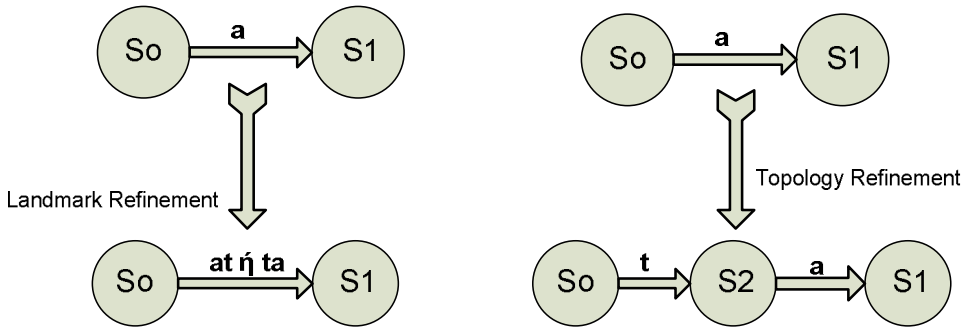
Η συνάρτηση αυτή είναι ένας άπληστος αλγόριθμος που στόχος της είναι να παραγάγει μία τέλεια διάζευξη. Ξεκινάει δημιουργώντας έναν αρχικό υποψήφιο για τέλεια διάζευξη με χρήση της συνάρτησης **GetInitialCandidates()**. Στη συνέχεια επιλέγει τον καλύτερο υποψήφιο με τη χρήση της **GetBestRefiner()** και προσπαθεί να τον βελτιώσει περαιτέρω. Αυτή η διαδικασία επαναλαμβάνεται είτε μέχρι να βρεθεί η τέλεια διάζευξη είτε μέχρι να εξαντληθούν οι υποψήφιοι.

PostProcess()

Αυτή η συνάρτηση βελτιώνει τη ποιότητα ενός κανόνα, προσπαθώντας να διατηρήσει τη διάζευξη γενική ώστε να καλύπτει όσο το δυνατόν περισσότερα σωστά παραδείγματα αλλά και ταυτόχρονα να μην καλύπτει λανθασμένα παραδείγματα. Αυτό το επιτυγχάνει με τρεις τρόπους: αντικαθιστώντας wildcards με tokens, προσθέτοντας περισσότερα tokens σε ένα landmark και συνενώνοντας landmarks.

Refine()

Η συνάρτηση αυτή προσπαθεί να βελτιώσει τις ήδη υπάρχουσες διαζεύξεις που υπάρχουν. Αυτό γίνεται είτε συγκεκριμενοποιώντας landmarks (landmark refinement) είτε προσθέτοντας νέες καταστάσεις (topology refinement). Ένας τρόπος για να βρεθούν τα tokens και τα wildcards τα οποία θα χρησιμοποιηθούν σε αυτή τη διαδικασία είναι να λάβουμε υπ'όψιν τα tokens του ορίσματος της GetInitialCandidates και τα wildcards που τα αντιπροσωπεύουν. Πιο συγκεκριμένα στα παρακάτω σχήματα φαίνονται οι δύο τρόποι :



σχήμα 3.7: Landmark και topology refinements

Στο παράδειγμα του σχήματος 3.6 ο αλγόριθμος επιλέγει ως Seed το παράδειγμα E2 αφού είναι το συντομότερο. Το τελευταίο token που πρέπει να καταναλωθεί είναι η παρένθεση «(» και έτσι παράγονται οι αρχικοί υποψήφιοι:R1=$SkipTo(()$, R2=$SkipTo(Punctuation)$, R3=$SkipTo(Anything)$. Επειδή ο κανόνας R1 είναι τέλεια διάζευξη, στο τέλος της πρώτης επανάληψης η LearnDisjunct() επιστρέφει τον R1. Αφαιρούνται στη συνέχεια από το σύνολο των παραδειγμάτων τα E2,E4 αφού καλύπτονται από τον R1 και έτσι η LearnDisjunct()

καλείται με βάση τα παραδείγματα {E1,E3}. Τώρα σαν Seed επιλέγεται το παράδειγμα E1 και παράγονται οι εξής τρεις υποψήφιοι: R4= $SkipTo()$, R5= $SkipTo(HtmlTag)$, R6= $SkipTo(Anything)$.Και οι τρεις υποψήφιοι αποτυγχάνουν «σταματώντας» πριν τον κωδικό 800. Έτσι επιλέγεται ένας υποψήφιος για να βελτιωθεί. Επειδή και οι τρεις είναι ισοδύναμοι και ως προς τις επιτυχίες αλλά και τις αποτυχίες επιλέγεται ο πρώτος αφού είναι ο πιο συγκεκριμένος από τους άλλους δύο που χρησιμοποιούν wildcards. Οι νέοι κανόνες που παράγονται από τη συνάρτηση Refine() είναι οι εξής: από landmark refinements τους R7= $SkipTo(-)$, R8= $SkipTo(Punctuation)$, R9= $SkipTo(Anything)$ και από topology refinements παράγονται οι κανόνες R10 ως και R24:

$R10 = SkipTo(Venice)SkipTo()$
$R11 = SkipTo()SkipTo(<b>)$
$R12 = SkipTo(:)SkipTo(<b>)$
$R13 = SkipTo(-)SkipTo(<b>)$
$R14 = SkipTo(,)SkipTo(<b>)$
$R15 = SkipTo(Phone)SkipTo(<b>)$
$R16 = SkipTo(1)SkipTo(<b>)$
$R17 = SkipTo(Numeric)SkipTo(<b>)$
$R18 = SkipTo(Punctuation)SkipTo(<b>)$
$R19 = SkipTo(HtmlTag)SkipTo(<b>)$
$R20 = SkipTo(Alphanum)SkipTo(<b>)$
$R21 = SkipTo(Alphabetic)SkipTo(<b>)$
$R22 = SkipTo(Capitalized)SkipTo(<b>)$
$R23 = SkipTo(NonHtml)SkipTo(<b>)$
$R24 = SkipTo(Anything)SkipTo(<b>)$

Ήδη με βάση τους παραπάνω κανόνες έχουν παραχθεί τέλειες διαζεύξεις. Συγκεκριμένα οι κανόνες R7, R11, R12,R13,R15,R16,R19 καλύπτουν σωστά τα E1 και E3 και ταυτόχρονα απορρίπτουν τα E2 και E4. Ανάμεσα σε αυτούς κατευθείαν απορρίπτεται ο R19 γιατί περιέχει wildcard και προτιμάμε τους πιο συγκεκριμένους κανόνες. Ανάμεσα στους υπόλοιπους επιλέγεται ο R7 γιατί έχει το μεγαλύτερο πλήθος tokens στο τελευταίο landmark. Έτσι ο αλγόριθμος τερματίζει δίνοντας σαν αποτέλεσμα την διάζευξη R1 ή R7.

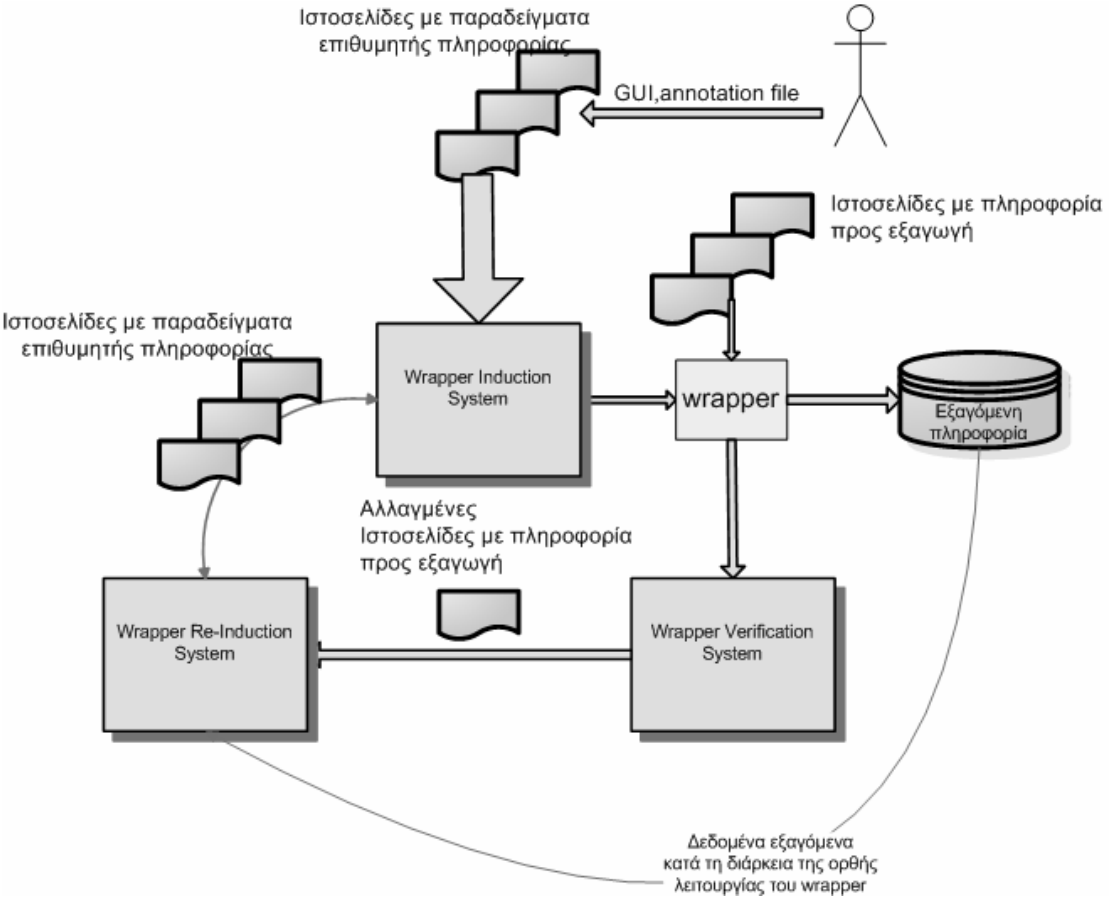
3.2 Wrapper Maintenance

3.2.1 Γενικά

Όπως έγινε φανερό από τα παραπάνω οι wrappers που παράγονται από τα wrapper induction συστήματα βασίζονται σε landmarks τα οποία με τη σειρά τους βασίζονται στην δομή της ιστοσελίδας. Όταν αλλάξει το layout της ιστοσελίδας ή το περιεχόμενο της τότε ο wrapper παύει να λειτουργεί σωστά. Συνεπώς προκύπτει η ανάγκη για την ανάπτυξη ενός αλγόριθμου που όταν εκτελείται θα εντοπίζει τη λανθασμένη λειτουργία του wrapper για μία συγκεκριμένη πηγή καθώς και ενός αλγορίθμου ο οποίος στη συνέχεια θα παράγει έναν καινούριο, σωστό wrapper προσαρμοσμένο στην αλλαγμένη πηγή. Το πρόβλημα αυτό ονομάζεται wrapper maintenance πρόβλημα. Στο σχήμα 3.8 βλέπουμε το τυπικό κύκλο ζωής ενός wrapper. Ο χρήστης αρχικά με κάποιο τρόπο που καθορίζεται από το wrapper induction σύστημα που θα χρησιμοποιήσει δίνει στο σύστημα ένα σύνολο από ιστοσελίδες μαζί με ένα

σύνολο από παραδείγματα της πληροφορίας που τον ενδιαφέρει να εξάγεται αυτόματα. Το σύστημα λαμβάνει τα δύο αυτά σύνολα σαν εισόδους και παράγει έναν wrapper. Ο wrapper έπειτα εφαρμόζεται σε ιστοσελίδες προερχόμενες από την ίδια πηγή για την οποία παράχθηκε και εξάγει τα δεδομένα. Σε τακτά χρονικά διαστήματα γίνεται έλεγχος της ορθής λειτουργίας του wrapper μέσα από ένα verification system. Αν εντοπιστεί κάποιο πρόβλημα στη λειτουργία του, τότε πυροδοτείται το wrapper reinduction system το οποίο λαμβάνει σαν εισόδους δεδομένα που εξάχθηκαν κατά τη διάρκεια της ορθής λειτουργίας του wrapper και τις αλλαγμένες ιστοσελίδες και παράγει στην έξοδο του ένα σύνολο από παραδείγματα της επιθυμητής πληροφορίας στις αλλαγμένες ιστοσελίδες. Και πάλι τα δύο αυτά σύνολα δίνονται σαν είσοδος στο wrapper induction system και παράγεται ο νέος, σωστός πλέον wrapper.

Τα προβλήματα της επαλήθευσης της ορθής λειτουργίας του wrapper και της επαγωγής ενός νέου, σωστού σε περίπτωση που εντοπιστεί η αλλαγή είναι δύο σύνθετα προβλήματα.



σχήμα 3.8 : Τυπικός κύκλος ζωής ενός wrapper

3.2.2 Wrapper Verification (Επαλήθευση ορθής λειτουργίας wrapper)

Υπάρχουν δύο προσεγγίσεις στην αντιμετώπιση του wrapper verification προβλήματος. Η πρώτη βασίζεται στη δομή των ιστοσελίδων (structured based) και η δεύτερη στο

περιεχόμενο τους (content based). Στις πρώτες τρεις παραγράφους 3.2.1.1 έως και 3.2.1.3 περιγράφουμε content based προσεγγίσεις στο πρόβλημα και στην 3.2.1.4 μία structured based προσέγγιση.

3.2.2.1 STRAWMAN αλγόριθμος

Η πρώτη προσέγγιση που έγινε στο πρόβλημα του wrapper verification ήταν αυτή του αλγόριθμου STRAWMAN. Για να διαπιστωθεί αν ο wrapper λειτουργεί σωστά για ένα site εκτελείται ένα query στο site αυτό για το οποίο έχουμε αποθηκεύσει την ιστοσελίδα που είχαμε πάρει σαν απάντηση όταν ο wrapper δούλευε σωστά. Αν η ιστοσελίδα που θα πάρουμε είναι ίδια με αυτή που έχουμε αποθηκεύσει τότε είμαστε βέβαιοι ότι ο wrapper λειτουργεί σωστά. Η προσέγγιση αυτή δεν αποδίδει για πολλά sites αφού βασίζεται στην υπόθεση ότι το site για ένα δοθέν query θα επιστρέψει την ίδια ιστοσελίδα σε κάθε χρονική στιγμή.

3.2.2.2 RAPTURE αλγόριθμος

Ο αλγόριθμος αυτός αναπτύχθηκε από τον Kushmerick και είναι χρονικά ο πρώτος ο οποίος δίνει ικανοποιητικά αποτελέσματα στο wrapper verification πρόβλημα χρησιμοποιώντας μία content based προσέγγιση ([Kus00]). Από μία λίγο διαφορετική οπτική γωνία μπορούμε να δούμε τον wrapper σαν μια συνάρτηση η οποία παίρνει σαν όρισμα μια σελίδα και επιστρέφει μία τιμή. Αυτή η τιμή ονομάζεται label και είναι μία σχέση-πίνακας σε όρους βάσεων δεδομένων, δηλαδή ένα σύνολο από εγγραφές κάθε μία από τις οποίες έχει K attributes. Θεωρούμε ότι υπάρχει ένα σύνολο τέτοιων labels τα οποία ελήφθησαν όταν ο wrapper λειτουργούσε σίγουρα σωστά. Το πρόβλημά μας είναι να διαπιστώσουμε αν ο wrapper εξάγει τα επιθυμητά δεδομένα. Πιο τυπικά έχουμε τα εξής :

*Είσοδος:*Wrapper w, page p, ένα σύνολο από σωστά labels $L=\{l_1,..,l_M\}$

*Έξοδος:*TRUE αν $w(p)=l_{real}$ δηλαδή ο wrapper w δουλεύει σωστά αλλιώς FALSE.

Ο αλγόριθμος RAPTURE συγκρίνει τα επαληθευμένα (verified) και τα προς έλεγχο (testing) labels. Θεωρεί ένα σύνολο χαρακτηριστικών της πληροφορίας (πυκνότητα ψηφίων, πυκνότητα γραμμάτων, πυκνότητα σημείων στίξης, πυκνότητα HTML χαρακτήρων, πλήθος λέξεων, μέσο μήκος λέξεων, συνολικό μήκος) και με βάση τις τιμές που παίρνει για τα verified labels κρίνει αν τα testing labels θα μπορούσαν να είναι σωστά. Το κάθε χαρακτηριστικό αντιπροσωπεύεται από μία τυχαία μεταβλητή η οποία θεωρείται ότι ακολουθεί την κανονική κατανομή. Ο αλγόριθμος θα παρουσιαστεί μέσα από ένα ενδεικτικό παράδειγμα. Στα παρακάτω θα χρησιμοποιούνται κεφαλαία γράμματα για τυχαίες μεταβλητές και μικρά γράμματα για τις τιμές τους. Με τους συμβολισμούς μ_R, σ_R εννοούμε την μέση τιμή και την τυπική απόκλιση της τυχαίας μεταβλητής R . $P[r;\mu_R;\sigma_R]$ είναι η πιθανότητα η

τυχαία μεταβλητή να είναι ίση r με ενώ $P[\leq r;\mu_R;\sigma_R]$ είναι η πιθανότητα η να παίρνει τιμή μικρότερη ή το πολύ ίση με r. Θεωρούμε ότι οι όλες οι τυχαίες μεταβλητές ακολουθούν κανονική κατανομή δηλαδή:

$$P[r;\mu_R;\sigma_R]=\frac{1}{\sigma_R\sqrt{2\pi}}e^{-\frac{(r-\mu_R)^2}{2\sigma_R{}^2}}$$

$$P[\leq r;\mu_R;\sigma_R]=\int_{-\infty}^{r}P[x;\mu_R;\sigma_R]dx$$

Ας θεωρήσουμε τις ιστοσελίδες p_a,p_b, p_c,p_d. Στην επόμενη σελίδα βλέπουμε τόσο τον HTML κώδικα των σελίδων όσο και το πώς εμφανίζεται στο χρήστη η κάθε μία ιστοσελίδα. Έστω ότι διαθέτουμε τον wrapper ccwrap ο οποίος γνωρίζουμε με βεβαιότητα ότι εξάγει σωστά δεδομένα από τις σελίδες p_a,p_b και θέλουμε να δούμε αν δουλεύει σωστά και για τις σελίδες p_c,p_d.

```
procedure ccwrap
   while there are more occurrences of <B>
    skip to next <B>
    extract country through next </B>
    skip to next <I>
    extract code through next </I>
return all extracted <country,code> pairs
```

Παρακάτω βλέπουμε τα labels που δίνει ο wrapper για κάθε μία ιστοσελίδα. Για τις verified σελίδες (αυτές από τις οποίες είμαστε βέβαιοι ότι εξάγουμε τη σωστή πληροφορία) τα labels είναι :

label (a)={<Argentina,*54*>,<Bangladesh,*880*>,<Croatia,*385*>,<Denmark,*45*>*}*

label (b)={<El Salvador,*503*>,<France,*33*>,<Greece,*30*>*}*

Αντίθετα για τις testing (προς επαλήθευση) τα labels είναι:

label (c)={<Hong Cong,*852*>,<Ireland,*353*>,<Japan,*81*>*}*

label (d)={ <254,*Libya*>,<218,*Mexico*>,<52,*Stop*>}

Στην σελίδα p_c, πρέπει ο αλγόριθμος να επιστρέψει ότι δεν υπάρχει πρόβλημα στη λειτουργία του wrapper ενώ στη σελίδα p_d πρέπει να εντοπίσει ότι ο wrapper δε δουλεύει σωστά πλέον.Ο αλγόριθμος δουλεύει σε βήματα:

Ιστοσελίδα p_a

```
<HTML>
<BODY>
<B>Argentina</B><I>54</I><BR>
<B>Bangladesh</B><I>880</I><BR>
<B>Croatia</B><I>385</I><BR>
<B>Denmark</B><I>45</I><BR>
<HR><I>Stop</I>
</BODY>
</HTML>
```

Argentina,*54*
Bangladesh,*880*
Croatia,*385*
Denmark,*45*
Stop

Ιστοσελίδα p_b

```
<HTML>
<BODY>
<B>El Salvador</B><I>503</I><BR>
<B>France</B><I>33</I><BR>
<B>Greece</B><I>30</I><BR>
<HR><I>Stop</I>
</BODY>
</HTML>
```

El Salvador,*503*
France,*33*
Greece,*30*
Stop

Ιστοσελίδα p_c

```
<HTML>
<BODY>
<B>Hong Cong</B><I>852</I><BR>
<B>Ireland</B><I>353</I><BR>
<B>Japan</B><I>81</I><BR>
<HR><I>Stop</I>
</BODY>
</HTML>
```

Hong Cong,*852*
Ireland,*353*
Japan,*81*
Stop

Ιστοσελίδα p_d

```
<HTML>
<BODY>
 <B>Kenya</B><I>254</I><B  R>
<B>Libya</B><I>218</I><BR>
<B>Mexico</B><I>52</I><BR>
<HR><I>Stop</I>
</BODY>
</HTML>
```

Kenya,**254**
Libya,**218**
Mexico,**52**
Stop

Βήμα 1ο

Θεωρούμε ότι το πλήθος των εγγραφών σε ένα label είναι μία τυχαία μεταβλητή N και ακολουθεί την κανονική κατανομή. Έτσι από τις σελίδες a,b υπολογίζω τα εξής:

	n	P[n; μ_N; σ_N]
σελίδα a	4	0.44
σελίδα b	3	0.44

από όπου προκύπτει ότι $\mu_N = 3.5, \sigma_N = 0.71$.

Βήμα 2ο

Σε αυτό το βήμα ασχολούμαστε μόνο με τις verified σελίδες. Για κάθε attribute υπολογίζουμε τις τιμές των χαρακτηριστικών (features) της πληροφορίας, τις εκτιμήτριες για την μέση τιμή και την τυπική απόκλιση για το κάθε χαρακτηριστικό καθώς και τις πιθανότητες που προκύπτουν με βάση την κανονική κατανομή με παραμέτρους ίσες με τις εκτιμήτριες να παίρνει η εξαχθείσα πληροφορία τις συγκεκριμένες τιμές για τα διάφορα χαρακτηριστικά. Στο παράδειγμά μας θεωρούμε το πλήθος λέξεων και το μέσο μήκος της λέξης σαν χαρακτηριστικά της πληροφορίας. Τα attributes είναι το όνομα της χώρας και ο κωδικός. Τα αποτελέσματα της παραπάνω διαδικασίας φαίνονται στον παρακάτω πίνακα. Με c συμβολίζουμε το πλήθος λέξεων και με u το μέσο μήκος λέξης..

Attribute 1 : Χώρα

χώρα	c	P[c;μ_{c1},σ_{c1}]	u	P[c;μ_{u1},σ_{u1}]
P_a				
Argentina	1	0.98	9	0.12
Bangladesh	1	0.98	10	0.057
Croatia	1	0.98	5	0.12
Denmark	1	0.98	7	0.22
P_b				
El	2	0.081	6	0.19
Salvador	1	0.98	6	0.19
France	1	0.98	6	0.19
Greece				

μ_{c1}=1.1 και σ_{c1}=0.38 μ_{u1}=7 και σ_{u1}=1.8

Attribute 2 : Κωδικός

κωδικός	c	$P[c;\mu_{c2},\sigma_{c2}]$	u	$P[c;\mu_{u2},\sigma_{u2}]$
P_a				
54	1	1	2	0.54
880	1	1	3	0.42
385	1	1	3	0.42
32	1	1	2	0.54
P_b				
503	1	1	3	0.42
33	1	1	2	0.54
30	1	1	2	0.54

μ_{c2}=1 και σ_{c2}=0 μ_{u2}=2.4 και σ_{u2}=0.53

Βήμα 3ο

Τώρα εκτελούμε τα βήματα 1,2 για τις testing σελίδες, δηλαδή γι αυτές που θέλουμε να ελέγξουμε αν ο wrapper εξάγει σωστά δεδομένα (εν προκειμένω τις σελίδες p_c,p_d).Για το σκοπό αυτό χρησιμοποιούμε τις διάφορες εκτιμήτριες μέσης τιμής και τυπικής απόκλισης για κάθε χαρακτηριστικό ανά attribute που προέκυψαν από τα βήματα 1,2. Έτσι προκύπτουν οι παρακάτω πίνακες:

	n	$P[n;\mu_N;\sigma_N]$
σελίδα c	3	0.44
σελίδα d	3	0.44

Attribute 1 : Χώρα

χώρα	c	$P[c;\mu_{c1},\sigma_{c1}]$	u	$P[c;\mu_{u1},\sigma_{u1}]$
P_c				
Hong	2	0.081	4	0.057
Kong	1	0.98	7	0.22
Ireland	1	0.98	5	0.12
Japan				
P_d				
254	1	0.98	3	0.02
218	1	0.98	3	0.02
52	1	0.98	2	$5.1x10^{-3}$

Attribute 2 : Κωδικός

κωδικός	c	P[c;μ_{c2},σ_{c2}]	u	P[c;μ_{u2},σ_{u2}]
P_c				
852	1	1	3	0.42
353	1	1	3	0.42
81	1	1	2	0.54
P_d				
Libya	1	1	5	$7x10^{-6}$
Mexico	1	1	6	$1.5x10^{-10}$
Stop	1	1	4	$9.9x10^{-4}$

Η φυσική σημασία των παραπάνω τιμών που προκύπτουν για τις διάφορες πιθανότητες είναι προφανής. Π.χ το ότι $P[4;\mu_{U_2};\sigma_{U_2}] = 9.9x10^{-4}$ σημαίνει ότι είναι απίθανο να βρούμε για το attribute κωδικός μία τιμή που το μέσο μήκος λέξης είναι ίσο με 4.

Βήμα 4ο

Στο βήμα αυτό υπολογίζουμε την πιθανότητα επαλήθευσης (verification probability). Για το σκοπό αυτό θεωρούμε την πιθανότητα αυτή σαν μία τυχαία μεταβλητή V που ακολουθεί κανονική κατανομή. Από τις verified σελίδες υπολογίζουμε τις εκτιμήτριες των παραμέτρων της κανονικής κατανομής. Για τον υπολογισμό της πιθανότητα επαλήθευσης για μία verified σελίδα πολλαπλασιάζουμε τις διάφορες πιθανότητες (θεωρώντας ότι είναι μεταξύ τους ανεξάρτητες) που έχουν υπολογιστεί και σχετίζονται με αυτή. Στη συνέχεια, με βάση τις διάφορες πιθανότητες επαλήθευσης που προέκυψαν από τις verified σελίδες υπολογίζουμε τις εκτιμήτριες για την μέση τιμή και την τυπική απόκλιση για την τυχαία μεταβλητή «πιθανότητα επαλήθευσης». Έτσι προκύπτει ο παρακάτω πίνακας :

	τιμές V
p_a .44,.98, .98, .98, .98,.12,.057, .12,.22,1,1,1,1,.54,.42,.42,.54	$3.8x10^{-6}$
p_b .44,.081, .98, .98, .19,.19,.19, 1,1,1,.42,.54,.54	$2.9x10^{-5}$

$\mu_V = 1.6x10^{-5}$ και $\sigma_V = 1.8x10^{-5}$

Τώρα με βάση τις εκτιμήσεις για την μέση τιμή και τη τυπική απόκλιση από τις verified σελίδες κατασκευάζουμε τον παρακάτω πίνακα :

$P[\leq v;\mu_V;\sigma_V]$

p_c .44,.081, .98, .98,.057,.22 .12,1,1,1,.42,.42,.54	$4.9x10^{-6}$	0.26
p_d .44,.98,.98,.98,.02,.02, $5.1x10^{-3}$,1,1,1,$7x10^{-6}$, $1.5x10^{-10}$,$9.9x10^{-4}$	$8.8x10^{-25}$	0.18

Βήμα 5ο

Αυτό είναι το τελευταίο βήμα του αλγόριθμου και σε αυτό γίνεται η επαλήθευση της λειτουργίας του wrapper. Η πιο απλή προσέγγιση είναι να συγκρίνουμε τις πιθανότητες επαλήθευσης για τις σελίδες p_c,p_d με την μέση τιμή πιθανότητας επαλήθευσης που προέκυψε από τις σελίδες p_a,p_b. Αν ΠιθανότηταΕπαλήθευσης($p_{testing}$)$<\mu_V$ τότε κρίνουμε ότι υπάρχει σφάλμα στη λειτουργία του wrapper. Μία πιο καλή προσέγγιση είναι να μπορούμε να ρυθμίζουμε την «αισιοδοξία» του αλγόριθμου για το αν ο wrapper δουλεύει σωστά. Έτσι υπολογίζουμε για τις testing σελίδες τις τιμές $P[\leq u;\mu_V;\sigma_V]$ όπου u είναι η τιμή που προέκυψε για την πιθανότητα επαλήθευσης για κάθε μία από τις testing σελίδες. Έπειτα συγκρίνουμε την τιμή $P[\leq u;\mu_V;\sigma_V]$ που προέκυψε με μία τιμή κατωφλίου (threshold) τ. Αν ισχύει $P[\leq u;\mu_V;\sigma_V] \prec \tau$ τότε συμπεραίνουμε ότι ο wrapper δεν εξάγει σωστά δεδομένα ενώ αν $P[\leq u;\mu_V;\sigma_V] \geq \tau$ τότε θεωρούμε ότι δεν υπάρχει πρόβλημα στην λειτουργία του wrapper. Αν θέσουμε τ=0.25 στο παράδειγμά μας ο αλγόριθμος RAPTURE συμπεραίνει ορθά ότι ο wrapper λειτουργεί σωστά για τη σελίδα c και λάθος για τη σελίδα d.

3.2.2.3 Wrapper Verification: Μία προσέγγιση βασισμένη στην μηχανική μάθηση

Ο αλγόριθμος που αναπτύχθηκε στην εργασία [LMK03] αποτελεί μία βελτίωση του RAPTURE αλγόριθμου αφού θεωρεί τα διάφορα γνωρίσματα που χρησιμοποιεί και ο RAPTURE αλλά επιπλέον εκμεταλλεύεται και κάποια patterns (μοτίβα) που ακολουθεί η πληροφορία. Συγκεκριμένα αναπτύσσεται ένας αλγόριθμος (DATAPROG) για την εξεύρεση patterns που ακολουθεί η εξαγόμενη πληροφορία. Για την verified (πληροφορία που εξάχθηκε κατά τη διάρκεια της ορθής λειτουργίας του wrapper) και την testing (πληροφορία που εξάχθηκε κατά τη διάρκεια της άγνωστης λειτουργίας του wrapper) πληροφορία δημιουργούνται δύο διανύσματα που περιέχουν αριθμητικές τιμές για τα διάφορα γνωρίσματα που θεωρούνται καθώς και τα patterns που ακολουθούν. Στη συνέχεια εκτελείται στατιστικός έλεγχος Pearson για να διαπιστωθεί αν τα δύο διανύσματα που έχουν

σχηματιστεί για την verified και για την testing πληροφορία μπορούν να θεωρηθούν όμοια. Τα γνωρίσματα τα οποία χρησιμοποιεί ο έλεγχος είναι το μέσο πλήθος εγγραφών ανά ιστοσελίδα, το μέσο πλήθος tokens ανά εγγραφή, το μέσο μήκος tokens καθώς και οι πυκνότητες των γραμμάτων, των ψηφίων , των σημείων στίξης και των HTML χαρακτήρων. Επίσης ο έλεγχος κάνει και χρήση των patterns που βρέθηκαν στην επαληθευμένη και την προς έλεγχο πληροφορία ως εξής: προσθέτουμε στην μεταβλητή του Pearson για κάθε κοινό pattern της verified και της testing πληροφορίας έναν όρο που σαν λογική έχει ότι οι εγγραφές που ακολουθούν ένα pattern θα καταλαμβάνουν περίπου το ίδιο ποσοστό επί των συνολικών εγγραφών τόσο στο training όσο και στο testing attribute. Έτσι αν θεωρήσουμε ένα pattern p_i το οποίο το ακολουθούν r_i εγγραφές στο training attribute και έχουμε N εγγραφές συνολικά στο training attribute και n στο testing attribute τότε το αναμενόμενο πλήθος εγγραφών στο testing attribute που ακολουθούν το pattern p_i θα είναι ίσο με nr_i / N .

Έτσι η μεταβλητή του Pearson αυξάνεται για το pattern p_i κατά $\frac{(r_i - nr_i / N)^2}{r_i}$ αυξάνοντας παράλληλα και τους βαθμούς ελευθερίας κατά 1. Αν η τελική προκύπτουσα τιμή για την μεταβλητή του Pearson είναι μεγαλύτερη από την κρίσιμη τιμή που προκύπτει από την κατανομή χι τετράγωνο για τους βαθμούς ελευθερίας που προέκυψαν και ένα δοθέν επίπεδο σημαντικότητας (συνήθως α=0.05) τότε το σύστημα συμπεραίνει ότι ο wrapper δεν εξάγει σωστά την πληροφορία.

3.2.2.4 Wrapper Verification: Μία προσέγγιση βασισμένη στην δομή

Στην εργασία [PLL03] προτείνεται μία μέθοδος που βασίζεται στη δομή των ιστοσελίδων από τις οποίες γίνεται η εξαγωγή της πληροφορίας. Ο λόγος για τον οποίο ο wrapper μπορεί να πάψει να εξάγει σωστά την πληροφορία όπως έχουμε δει, είναι η άμεση εξάρτηση της ορθής λειτουργίας του από το DOM δένδρο της ιστοσελίδας καθώς και η αδυναμία του να αντιληφθεί σημασιολογικές αλλαγές των περιεχομένων της. Η ιδέα του αλγορίθμου είναι η εξής: Όταν είναι γνωστό ότι ο wrapper εξάγει τα επιθυμητά δεδομένα, ο αλγόριθμος αποθηκεύει για κάθε κανόνα εξαγωγής τη «διαδρομή» του κανόνα, καταγράφοντας από τη ρίζα του DOM δένδρου μέχρι το φύλλο στο οποίο υπάρχει η πληροφορία όλα τα tags καθώς και τους δείκτες που τα συνοδεύουν (τα φύλλα συμβολίζονται με τον χαρακτήρα ‘/’), καταγράφει το πλήθος των φύλλων που εξάγονται από τον συγκεκριμένο κανόνα το οποίο συμβολίζουμε με θ(T) όπου T η χρονική στιγμή στην οποία υπολογίζονται τα παραπάνω στοιχεία και επιπλέον καταγράφει για κάθε κόμβο του DOM δένδρου μία τιμή που ισούται με το πλήθος των παιδιών του κόμβου. Αν τη χρονική στιγμή T_1 είναι γνωστή η ορθή λειτουργία του wrapper και τη χρονική στιγμή T_2 γίνεται έλεγχος για επαλήθευση της λειτουργίας του wrapper, εξετάζεται αν τα δεδομένα είναι παρόμοια. Συγκεκριμένα για κάθε

κόμβο που τη χρονική στιγμή T_1 έχει διαφορετικό πλήθος παιδιών από τη χρονική στιγμή T_2 γίνεται ένας έλεγχος των νέων παιδιών που αποκτήθηκαν. Αν η διαδρομή από ένα καινούριο παιδί μέχρι κάποιο φύλλο-απόγονο μοιάζει με τη διαδρομή του κανόνα εξαγωγής από ένα αδερφικό κόμβο που υπήρχε και κατά τη χρονική στιγμή T_1 μέχρι κάποιο φύλλο-απόγονο του τότε το $\theta(T_2)$ αυξάνεται κατά 1. Το σύστημα κρίνει ότι ο wrapper δεν εξάγει σωστά δεδομένα αν $\theta(T_2) \succ \theta(T_1)$ ενώ αν $\theta(T_2) \leq \theta(T_1)$ τότε κρίνει ότι δεν υπάρχει πρόβλημα στην λειτουργία του.

3.2.3 Wrapper Reinduction

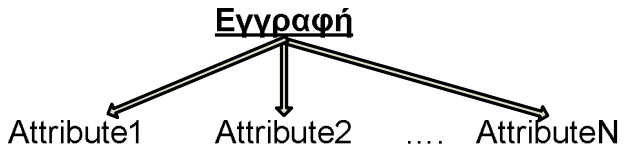
Στην παράγραφο αυτή περιγράφουμε μία κλασσική προσέγγιση για το wrapper reinduction πρόβλημα ([RPV05]). Η ιδέα της προσέγγισης αυτής είναι μέσα από αποτελέσματα που ελήφθησαν κατά την περίοδο της ορθής λειτουργίας του wrapper να προσδιορίσουμε στις αλλαγμένες ιστοσελίδες παραδείγματα, τα οποία αν δοθούν σε ένα wrapper induction system θα παραχθεί ένας νέος wrapper που θα εξάγει την επιθυμητή πληροφορία στις αλλαγμένες σελίδες. Το σύστημα κατά τη διάρκεια της ορθής λειτουργίας του wrapper κάνει queries στο site και αποθηκεύει σε μία βάση δεδομένων ένα πλήθος εγγραφών για κάθε query που πραγματοποιεί. Συγκεκριμένα, για κάθε εξαγόμενη εγγραφή το σύστημα εκτός από την ίδια την εγγραφή αποθηκεύει επιπλέον το query από το οποίο προέκυψε, τον κανόνα εξαγωγής που χρησιμοποίησε ο wrapper για να την εξάγει καθώς και μία ημερομηνία λήξης που σαν φυσική σημασία έχει το έως πότε μπορεί να θεωρείται η εγγραφή αυτή έγκυρη για το σύστημα. Το περιεχόμενο αυτής της βάσης ελέγχεται περιοδικά και συγκεκριμένα ελέγχεται αν το πλήθος των έγκυρων εγγραφών είναι επαρκές για το σύστημα με βάση κάποια κριτήρια που έχει θέσει. Όσες έχουν «λήξει» απομακρύνονται από τη βάση και στη συνέχεια γίνονται queries για να πάρουμε τόσες ακόμα όσες χρειαζόμαστε για να πληρούνται οι απαιτήσεις.
Τα σημαντικότερα κριτήρια που θέλουμε να πληρούνται κάθε χρονική στιγμή στη βάση είναι τα παρακάτω:

- Να υπάρχει ένα πλήθος εγγραφών μεγαλύτερο από μία ελάχιστη τιμή.
- Αυτές οι εγγραφές να προέρχονται από ένα ελάχιστο πλήθος διαφορετικών queries.

Ας θεωρήσουμε ότι το σύνολο $Q = \{q_1, .., q_n\}$ είναι τα queries των οποίων τα αποτελέσματα συλλέγονται κατά τη διάρκεια της ορθής της λειτουργίας του wrapper. Το αποτέλεσμα για το query q_i είναι το σύνολο $R_i = \{r_{i1}, r_{i2}, .., r_{im}\}$ (σύνολο από εγγραφές) και προέρχεται από το σύνολο σελίδων P_i. Η λίστα τιμών (value list) $V_{ij} = \{v_{ij(1)}, .., v_{ij(f)}\}$ μίας αποθηκευμένης εγγραφής r_{ij} είναι η λίστα όλων των ατομικών τιμών της r_{ij} . Στόχος μας είναι για κάθε αποθηκευμένη εγγραφή r_{ij} να βρούμε στις αλλαγμένες ιστοσελίδες μία εμφάνισή της. Αρχικά προσδιορίζουμε όλες τις υποψήφιες εμφανίσεις. Μία υποψήφια εμφάνιση σε μία

σελίδα $p \in P_i$ για την εγγραφή $r_{ij} \in R_i$ του query q_i είναι της μορφής $c = \{(v_{ij(1)}, pos_1), .., (v_{ij(f)}, pos_f)\}$ όπου $(v_{ij(k)}, pos_k)$, $k \in \{1..f\}$ είναι η pos_k εμφάνιση της ατομικής τιμής $v_{ij(k)} \in V_{ij}$. Κάθε $(v_{ij(k)}, pos_k)$ $k \in \{1..f\}$ ονομάζεται και υποψήφια εμφάνιση πεδίου (candidate field occurrence). Από τις υποψήφιες εμφανίσεις προσδιορίζονται οι σωστές , δηλαδή εκείνες που αντιστοιχούν σε πραγματικές εγγραφές και που θα ονομάζονται ορθά παραδείγματα (labeled examples). Αυτό επιτυγχάνεται εξετάζοντας αν οι διάφορες τιμές $v_{ij(k)}$ $k \in \{1..f\}$ είναι μέρος της ίδιας εγγραφής στη σελίδα p. Αν για κάποια εγγραφή δεν υπάρχει κάποιο ορθό παράδειγμα τότε η εγγραφή αυτή ονομάζεται άχρηστη. Συνεπώς το πρόβλημα που έχει να αντιμετωπίσει το σύστημα είναι το εξής: Έστω $Q = \{q_1, .., q_n\}$ το σύνολο των queries σε μία πηγή s (site-διαδικτυακός τόπος) των οποίων τα αποτελέσματα συλλέγονται κατά τη διάρκεια της ορθής λειτουργίας του wrapper. Έστω $R_i = \{r_{i1}, r_{i2}, .., r_{im}\}$ το σύνολο των αποθηκευμένων αποτελεσμάτων για το query q_i και έστω P_i το σύνολο των σελίδων από τις οποίες προέρχονται τα αποτελέσματα. Για κάθε r_{ij} $i = 1..n, j = 1..m$ βρες c_{ij} το οποίο να είναι ένα ορθό παράδειγμα στις σελίδες $p \in P_i$. Αν υπάρχουν περισσότερα από ένα ορθά παραδείγματα τότε ευνοείται εκείνο το παράδειγμα που βρίσκεται στη σελίδα με τα περισσότερα ορθά παραδείγματα συνολικά για όλα τα αποτελέσματα ($r_{ik}, k \neq j$). Το πρόβλημα αυτό ονομάζεται *labelling problem.*

Θα περιγράψουμε την λειτουργία του αλγόριθμου στη περίπτωση που οι εγγραφές που θέλουμε να εξάγουμε αποτελούνται μόνο από ατομικά attributes (επίπεδες εγγραφές). Σε ορολογία Embedded Catalog σημαίνει ότι η εγγραφή θα χει την δομή του σχήματος 3.9:



σχήμα 3.9 : Επίπεδες εγγραφές (flat tuples)

Ο αλγόριθμος παίρνει σαν είσοδο μία σελίδα P[1], έναν τύπο δεδομένων T $< attribute_1, .. attribute_f >$, ένα σύνολο αποτελεσμάτων $R = \{r_1, r_2, .., r_n\}$ όπου το κάθε r_i είναι τύπου T και είναι της μορφής $\{(attribute_1, value_{i1}), .. (attribute_f, value_{if})\}$ και μία παράμετρο maxCandidates η οποία καθορίζει το μέγιστο πλήθος υποψήφιων αποτελεσμάτων που θα διατηρεί ο αλγόριθμος. Η έξοδος του αλγόριθμου είναι ένα σύνολο συνόλων

---

[1]εν γένει μπορεί να έχουμε και πολλές σελίδες σαν είσοδο αλλά τις θεωρούμε σαν ένα γενικευμένο DOM δένδρο που προκύπτει από την ένωση των επιμέρους DOM δένδρων

$E = \{E_1, ..E_n\}$ όπου το είναι E_i ένα σύνολο που είτε είναι κενό αν το r_i είναι άχρηστο είτε περιέχει τις «καλύτερες» δυνατές εμφανίσεις του r_i στο P. Ο αλγόριθμος για να κρίνει ποιες είναι οι «καλύτερες» εμφανίσεις βασίζεται σε τρεις ευριστικές: την ευριστική των αδερφικών κόμβων, την ευριστική των patterns και την ευριστική της γειτνίασης.

Ευριστική αδερφικών κόμβων (The Sibling nodes Heuristic)

Η ευριστική των αδερφικών κόμβων βασίζεται σε διάφορες παρατηρήσεις. Δοθέντος του DOM δένδρου μίας ιστοσελίδας που περιέχει ένα σύνολο εγγραφών, κάθε εγγραφή βρίσκεται συνήθως εξ ολοκλήρου μέσα σε ένα υπόδενδρο το οποίο έχει σαν αδέρφια άλλα υπόδενδρα τα οποία περιέχουν τις άλλες εγγραφές. Η δεύτερη παρατήρηση είναι πως αν η υποψήφια εμφάνιση c_1 για την εγγραφή r_i δεν είναι ορθό παράδειγμα τότε το υπόδενδρο της πιθανότατα θα περιέχει δεδομένα από περισσότερες από μία εγγραφές. Έτσι είναι αρκετά πιθανό μέσα στο ίδιο ή σε κάποιο αδερφικό υπόδενδρο[1] να περιέχονται και υποψήφιες εμφανίσεις για κάποιο άλλο αποτέλεσμα r_j. Η παρατήρηση αυτή αξιοποιείται προσδιορίζοντας για την υποψήφια εμφάνιση $c = \{(v_{i1}, pos_1), .., (v_{if}, pos_f)\}$ το ελάχιστο υπόδενδρο της, δηλαδή το υπόδενδρο με το ελάχιστο ύψος που περιέχει όλες τις τιμές $v_{i1}, .., v_{if}$. Έστω s_1 αυτό και $S = \{s_1, ..s_t\}$ το σύνολο των αδερφικών υπόδενδρων του s_1 (συμπεριλαμβανομένου και του s_1). Αν κάποιο s_i περιέχει υποψήφιες εμφανίσεις για αποτελέσματα από δύο ή και περισσότερα queries τότε το c_1 συνήθως δεν είναι ορθό παράδειγμα για την εγγραφή r_i. Η τελευταία παρατήρηση βασίζεται στην έννοια των συμμετρικών υποψήφιων εμφανίσεων, η οποία εκφράζει με πιο τυπικό τρόπο τη παρατήρηση ότι τα αποτελέσματα που παίρνουμε σαν απάντηση από ένα site σε ένα query οργανώνονται συνήθως σε λίστα και εμφανίζονται κατά τον ίδιο τρόπο στον χρήστη και το ότι η σειρά με την οποία εμφανίζονται τα attributes στις εγγραφές είναι κοινή. Εκφράζεται ως εξής: Για δύο αποτελέσματα $r_i, r_j \in R$ τα ορθά παραδείγματα τους είναι συμμετρικά. Δύο υποψήφιες εμφανίσεις $c_i = \{(v_{i1}, pos_{i1}), .., (v_{if}, pos_{if})\}$ για το r_i και $c_j = \{(v_{j1}, pos_{j1}), .., (v_{jf}, pos_{jf})\}$ για το r_j, είναι συμμετρικές όταν ισχύουν τα εξής :

- Για κάθε $k \in \{1..f\}$ η XPATH διαδρομή στο DOM δένδρο του (v_{ik}, pos_{ik}) είναι ίδια με την XPATH διαδρομή του (v_{jk}, pos_{jk}).
- Για κάθε $k \in \{1..f\}$ $(v_{ik}, pos_{ik}) \neq (v_{jk}, pos_{jk})$.

---

[1] Δύο υπόδενδρα ονομάζονται αδερφικά αν οι ρίζες τους έχουν κοινό γονέα.

- Αν το v_{ik} εμφανίζεται πριν το v_{im} στο c_i $k,m \in \{1..f\}$ τότε και το v_{jk} εμφανίζεται πριν v_{jm} στο c_j .

Με βάση όλες τις παραπάνω παρατηρήσεις μπορούμε να σχηματίσουμε την ευριστική για να αξιολογήσουμε την υποψήφια εμφάνιση c για το αποτέλεσμα $r_i \in R$:

1)Προσδιόρισε το ελάχιστο υπόδενδρο s που περιέχει το c.

2)Έστω $S = \{s_1, ..s_t\}$ το σύνολο των αδερφικών υπόδενδρων του s.Σε καθένα από αυτά ψάξε για υποψήφιες εμφανίσεις άλλων αποτελεσμάτων $r_j \in R$ με $j \neq i$ που να είναι συμμετρικές με το c. Το πλήθος των συμμετρικών εμφανίσεων του c στα αδερφικά υπόδενδρα θα είναι ο βαθμός αξιολόγησης του.

3)Αν βρεθούν υποψήφιες εμφανίσεις για δύο ή και περισσότερα queries στο s ή σε κάποιο $s_k \in S$ τότε το c παίρνει βαθμό αξιολόγησης 0.

Η ευριστική αυτή δεν είναι πάντα εφαρμόσιμη. Σε τέτοιες περιπτώσεις ακόμα και τα ορθά παραδείγματα θα πάρουν βαθμό αξιολόγησης 0.

Ευριστική Pattern (Pattern Heuristic)

Αυτή η ευριστική βασίζεται επίσης στη παρατήρηση ότι τα αποτελέσματα που παίρνουμε σαν απάντηση από ένα site σε ένα query οργανώνονται συνήθως σε λίστα και εμφανίζονται κατά τον ίδιο τρόπο στον χρήστη. Η ιδέα της είναι ότι για να αξιολογήσουμε την υποψήφια εμφάνιση c για ένα query αποτέλεσμα $r_i \in R$, βρίσκουμε από το c μία κανονική έκφραση-pattern. Έπειτα εφαρμόζεται το pattern στη σελίδα και λαμβάνεται έτσι ένα σύνολο από συμβολοσειρές (strings) τα οποία ταιριάζουν με το pattern. Το πλήθος των συμβολοσειρών οι οποίες εμπεριέχουν κάποια εγγραφή που είναι αποτέλεσμα κάποιου άλλου query αποτελέσματος είναι ο βαθμός αξιολόγησης του c.Ο προσδιορισμός του pattern γίνεται με κατά βάθος διάσχιση του DOM δένδρου. Αντικαθιστούμε τις διάφορες ατομικές τιμές του c με τα ονόματα των attributes που αντιπροσωπεύουν και στη θέση κάθε άλλου token που δεν είναι HTML χαρακτήρας μέσα στο pattern θέτουμε έναν wildcard χαρακτήρα ($ANY). Τέλος τις υποψήφιες εμφανίσεις που έχουν προκύψει με ψηλό βαθμό αξιολόγησης από τις παραπάνω ευριστικές αλλά ισοβαθμούν τις ταξινομούμε με βάση την ευριστική της γειτνίασης .

Ευριστική Γειτνίασης (Proximity Heuristic)

Αυτή η ευριστική βασίζεται στην παρατήρηση ότι οι τιμές των διαφόρων attributes μίας εγγραφής τείνουν να βρίσκονται κοντά μέσα στη σελίδα. Αν η απόσταση μεταξύ δύο τιμών v_i, v_j είναι ίση με το (ελάχιστο δυνατό) πλήθος των DOM κόμβων ανάμεσα τους, τότε ορίζουμε την απόσταση μίας υποψήφιας εμφάνισης $c = \{(v_{i1}, pos_1), .., (v_{if}, pos_f)\}$ σαν το

άθροισμα των αποστάσεων μεταξύ των διαδοχικών ατομικών τιμών. Η υποψήφια εμφάνιση με την μικρότερη απόσταση θεωρείται καλύτερη.

Ο παραπάνω αλγόριθμος για να έχει καλά πρακτικά αποτελέσματα εκτελείται συχνά ώστε να υπάρχει βεβαιότητα για το ότι θα βρει ορθά παραδείγματα. Αν και αυτή η πρακτική σε πολλές πηγές αποδίδει, υπάρχουν πηγές που από τη φύση τους μεταβάλλονται τόσο ριζικά ώστε οι αποθηκευμένες εγγραφές να είναι άχρηστες. Ενδεικτικά αναφέρουμε τα χρηματιστηριακά και τα ειδησεογραφικά sites. Σε τέτοιες περιπτώσεις διακρίνονται στις εγγραφές σταθερά και μεταβλητά attributes. Για κάθε σταθερό attribute ενεργούμε όπως περιγράφηκε παραπάνω ενώ για τα μεταβλητά προσδιορίζονται κάποιες κανονικές εκφράσεις που εκφράζουν την συνήθη δομή τους. Αυτές οι κανονικές εκφράσεις θα χρησιμοποιηθούν για τον εντοπισμό τιμών για τα μεταβλητά attributes στις αλλαγμένες ιστοσελίδες της πηγής. Τέλος, ο αλγόριθμος που περιγράψαμε μπορεί να εφαρμοστεί και σε τύπους που έχουν και σύνθετα πεδία . Η ιδέα για να το πετύχουμε βασίζεται στην αναδρομή και είναι η εξής : Αν ο τύπος αναλυθεί σε μία δενδρική δομή με βάθος k σύμφωνα με το φορμαλισμό του Embedded Catalog τότε στο (k-1) επίπεδο όλα τα σύνθετα πεδία αποτελούνται από ατομικά πεδία και μόνο. Οπότε ο αλγόριθμος για τους απλούς τύπους εφαρμόζεται αναδρομικά για τα σύνθετα πεδία.

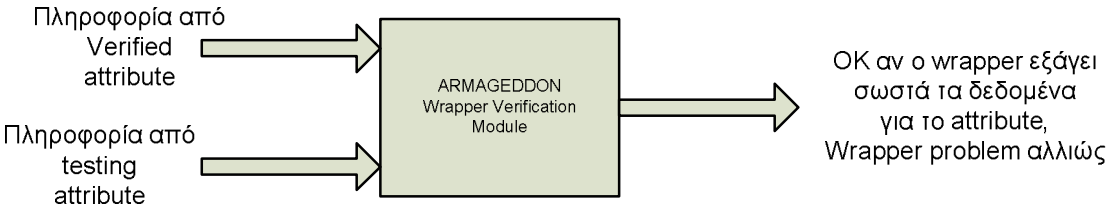
4

ARMAGEDDON[1] Wrapper Maintenance System

Στο κεφάλαιο αυτό γίνεται η περιγραφή του συστήματος wrapper maintenance που αναπτύχθηκε στα πλαίσια αυτής της εργασίας. Στην πρώτη παράγραφο αναπτύσσεται το wrapper verification system ενώ στη δεύτερη το wrapper reinduction system.

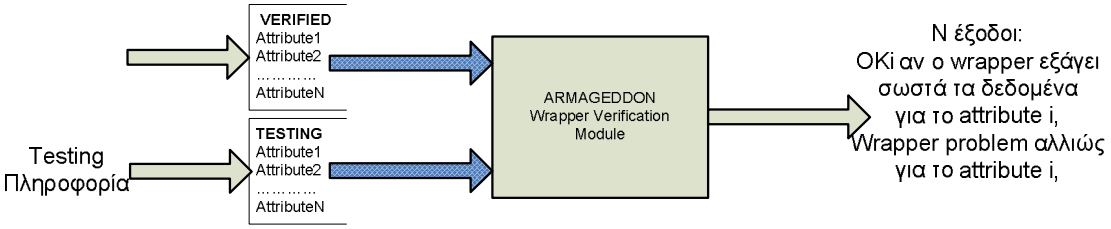
4.1 Σύστημα επαλήθευσης ορθής λειτουργίας wrapper (Wrapper Verification system)

Το σύστημα που αναπτύχθηκε βασίζεται σε ένα σύνθετο αλγόριθμο ο οποίος προσπαθεί να κατασκευάσει ένα διάνυσμα μεταπληροφορίας για την εξαγόμενη πληροφορία. Στα σχήματα 4.1α και 4.1β βλέπουμε τι μπορεί να δεχθεί ο αλγόριθμος μας ως είσοδο:



σχήμα 4.1α: ARMAGEDDON wrapper maintenance system με είσοδο ένα attribute

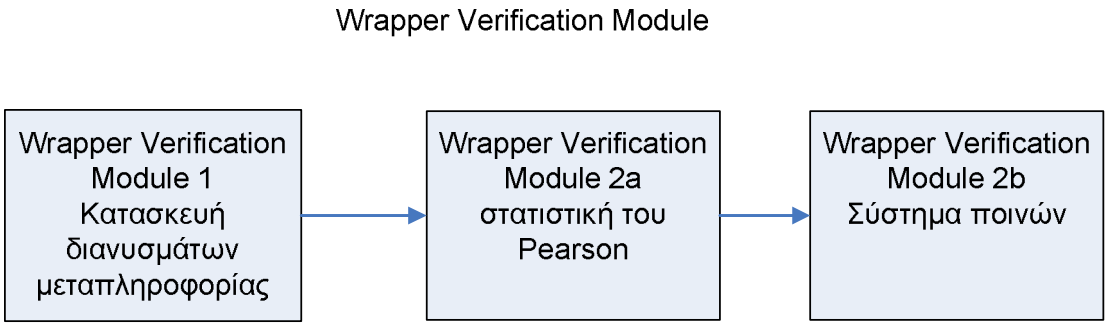
[1]Το όνομα του συστήματος προέκυψε σαν λογοπαίγνιο: rapture που είναι το όνομα της πιο γνωστής εργασίας πάνω στο wrapper verification πρόβλημα, σημαίνει αναρπαγή των πιστών πριν από τον Αρμαγεδόνα



σχήμα 4.1β: ARMAGEDDON wrapper maintenance system με είσοδο πολλά attributes

Στην πρώτη περίπτωση (σχ.4.1α) ο αλγόριθμος λαμβάνει σαν είσοδο δύο φακέλους που περιέχουν πληροφορία που αφορά ένα συγκεκριμένο attribute. Ο πρώτος φάκελος περιέχει την πληροφορία που γνωρίζουμε ότι εξήχθη κατά τη διάρκεια της ορθής λειτουργίας του wrapper ενώ ο δεύτερος φάκελος περιέχει την πληροφορία που εξήχθη κατά τη διάρκεια της άγνωστης λειτουργίας του wrapper. Στη συνέχεια αποφασίζεται αν ο wrapper έχει «χαλάσει» ή παραμένει σωστός. Στη δεύτερη περίπτωση (σχ.4.1β), δώσαμε την δυνατότητα στο χρήστη να δίνει σαν είσοδο ταυτόχρονα πολλά attributes. Ουσιαστικά ανάγεται στη πρώτη περίπτωση και σκοπεύει στη διευκόλυνση του χρήστη και στην απαλλαγή του από την επαναληπτική εκτέλεση της πρώτης περίπτωσης .

Το system verification module έχει δύο κύρια μέρη. Το πρώτο συνίσταται στη κατασκευή των διανυσμάτων μεταπληροφορίας για την επικυρωμένη και την προς έλεγχο πληροφορία. Το δεύτερο συνίσταται στον έλεγχο της ορθής λειτουργίας του wrapper με βάση τα δύο διανύσματα που προέκυψαν. Αυτό το μέρος έχει δύο διακριτά υπομέρη. Το πρώτο είναι ο έλεγχος Pearson και το δεύτερο ένα σύστημα ποινών που δίνει ιδιαίτερη έμφαση στη σημασιολογία των εξαγόμενων δεδομένων. Αυτό το σύστημα βασίζεται σαν ιδέα στην ευριστική ότι ίδια σημασιολογία δίνει παρόμοια patterns. Στο παρακάτω σχήμα φαίνεται η δομή του verification module:



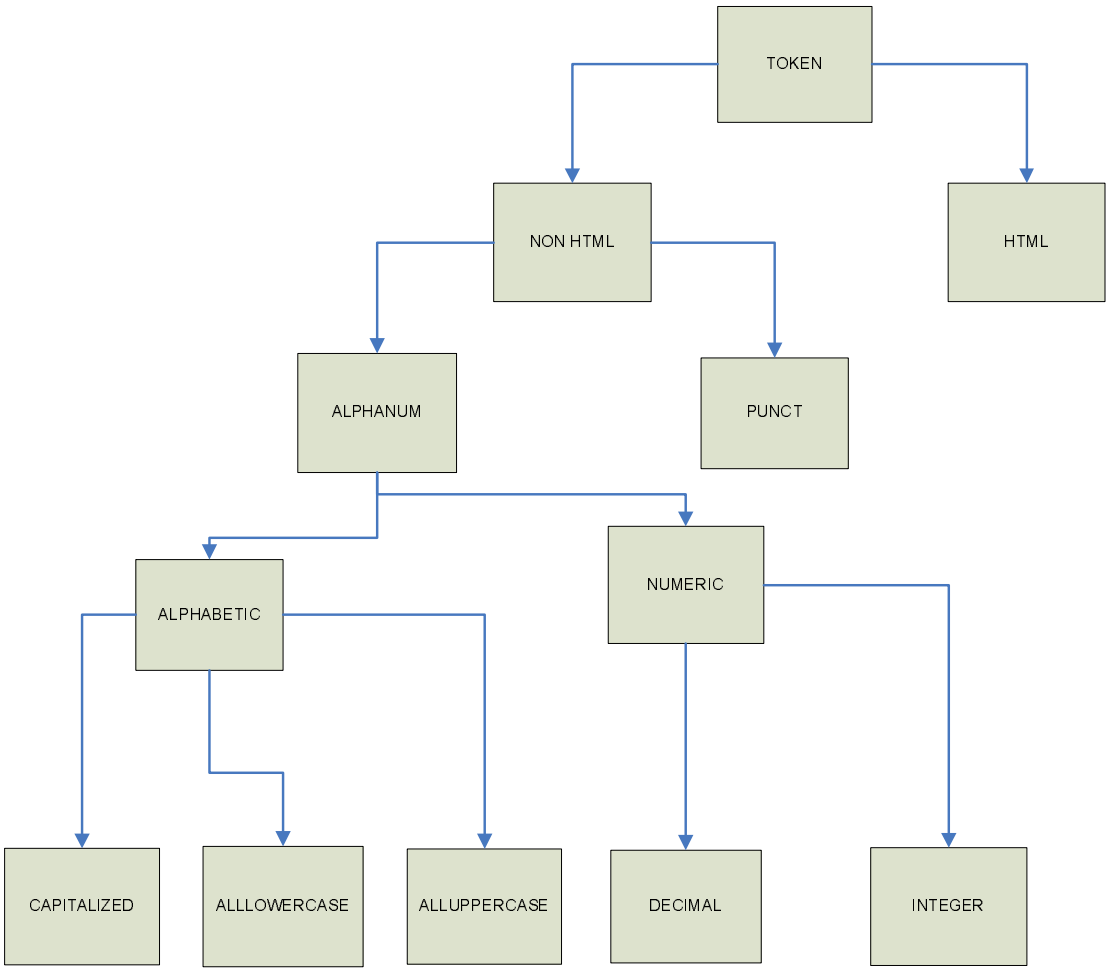
σχήμα 4.2 : ARMAGEDDON wrapper verification module

Παρακάτω αναλύεται η λειτουργία του κάθε μέρους για την πρώτη μορφή εισόδου.

Βήμα 1^{ο} :Κατασκευή διανυσμάτων μεταπληροφορίας

Ζητείται από το χρήστη αρχικά να δώσει τη διεύθυνση ενός φακέλου που περιέχει ένα σύνολο από αρχεία με την εξαγόμενη πληροφορία που θεωρείται ότι εξάχθηκε κατά τη διάρκεια της ορθής λειτουργίας του wrapper. Αυτός ο φάκελος ονομάζεται training attribute γιατί με βάση τα δεδομένα αυτού του φακέλου το σύστημα ARMAGEDDON εκπαιδεύεται

ώστε να μπορεί να αντιληφθεί αν ο wrapper εξάγει την επιθυμητή πληροφορία ή όχι. Κατά την υλοποίηση του συστήματος το όνομά του φακέλου θεωρείται ότι είναι ίδιο με αυτό του attribute που αντιπροσωπεύει. Αυτό έγινε κυρίως λόγω της δεύτερης μορφής εισόδου στην οποία ο χρήστης δίνει πολλά attributes ταυτόχρονα. Έτσι αυτή η σύμβαση επιτρέπει να γίνεται εύκολα ο εντοπισμός των αντιστοίχων attributes ανάμεσα στις δύο εισόδους. Μετά την εισαγωγή του training attribute το verification module διασπά το κείμενο σε tokens και απονέμει σε κάθε token ένα τύπο (token type). Συγκεκριμένα, το σύστημα έχει κατασκευάσει μία ιεραρχία τύπων και απονέμει σε κάθε token τον πιο εξειδικευμένο τύπο που του ταιριάζει. Η γενικότητα των τύπων μειώνεται από πάνω προς τα κάτω. Στο παρακάτω σχήμα βλέπουμε την ιεραρχία και ορισμένα παραδείγματα απονομής τύπων σε tokens για να γίνει αντιληπτή η σημασία τους:



σχήμα 4.3:Ιεραρχία τύπων tokens (token types)

Το token ”CS123” έχει τύπο ALPHANUM γιατί αποτελείται τόσο από γράμματα όσο και από ψηφία, το “12” INTEGER γιατί είναι ακέραιος, το 12.3 DECIMAL γιατί είναι δεκαδικός, το “DATABASE” ALLUPPERCASE γιατί αποτελείται μόνο από κεφαλαία γράμματα, το “course” ALLLOWERCASE γιατί αποτελείται μόνο από πεζά γράμματα, το “!” PUNCT γιατί αποτελεί σημείο στίξης, το “Alice” CAPITALIZED γιατί αποτελείται μόνο από

γράμματα εκ των οποίων μόνο το πρώτο είναι κεφαλαίο και το “TheBook” ALPHABETIC αποτελείται μόνο από γράμματα αλλά δεν είναι ούτε CAPITALIZED ούτε ALLLOWERCASE ούτε ALLUPPERCASE.

Το verification module θεωρεί ότι κάθε εξαγόμενη εγγραφή καταλαμβάνει μία διαφορετική γραμμή στα αρχεία τα οποία περιέχουν την εξαγόμενη πληροφορία. Έτσι το πλήθος των μη κενών γραμμών των αρχείων μέσα στο φάκελο που αντιπροσωπεύει ένα attribute είναι το πλήθος των εγγραφών που εξάχθηκαν για το συγκεκριμένο attribute. Αρχικά υπολογίζεται για το σύνολο όλων των εγγραφών του attribute η μέση πυκνότητα πεζών γραμμάτων, κεφαλαίων γραμμάτων, χαρακτήρων στίξης, το μέσο μήκος tokens και το μέσο πλήθος tokens ανά εγγραφή. Σε αντίθεση όμως με τον RAPTURE αλλά και με τον αλγόριθμο της εργασίας [LMK03] δεν κάνει καθόλου χρήση του γνωρίσματος πυκνότητα HTML χαρακτήρων. Έτσι λαμβάνεται υπ’ όψιν το γεγονός πως πρακτικά ακόμα και ένας σωστός wrapper μπορεί να εξάγει κάποιους, έστω και λίγους HTML χαρακτήρες. Αυτό έχει σαν συνέπεια το σύστημα μας να μην απαιτεί καθαρισμό των δεδομένων (data cleansing) καθιστώντας το έτσι εύρωστο.

Το σύνολο των τύπων tokens που χρησιμοποιούνται στην πράξη είναι το εξής: TT={ALPHANUM, ALPHABETIC, CAPITALIZED, ALLLOWERCASE, ALLUPPERCASE, INTEGER, DECIMAL, PUNCT, “*”} όπου ο χαρακτήρας * παίζει τον ρόλο του wildcard χαρακτήρα και αντιπροσωπεύει οποιονδήποτε τύπο token που δεν είναι κάποιος από τους υπόλοιπους του συνόλου TT. Ο τύπος που δίνει την μέση πυκνότητα για τους χαρακτήρες τύπου x δίνεται από την παρακάτω σχέση:

$$\text{μέση πυκνότητα x}=\frac{\text{πλήθος χαρακτήρων τύπου x}}{\text{συνολικό πλήθος χαρακτήρων}}$$

(όπου στο παρονομαστή συμπεριλαμβάνονται και οι κενοί χαρακτήρες).

Το μέσο πλήθος των tokens ανά εγγραφή προκύπτει από τη σχέση :

$$\text{μέσο πλήθος tokens ανά εγγραφή}=\frac{\text{συνολικό πλήθος tokens}}{\text{συνολικό πλήθος εγγραφών}}$$

ενώ το μέσο μήκος των tokens δίνεται από τη σχέση:

$$\text{μέσο μήκος tokens}=\frac{\text{συνολικό μήκος tokens}}{\text{πλήθος tokens}}.$$

Στη συνέχεια το σύστημα προσπαθεί να ανακαλύψει τα patterns τα οποία περιγράφουν τη συνήθη δομή που έχουν τα πρώτα tokens μίας εγγραφής ενός attribute. Τα patterns αυτά θα ονομάζονται starting patterns. Με αυτό τον τρόπο προσπαθούμε να εκμεταλλευτούμε το γεγονός ότι ίδια σημασιολογία μεταξύ δύο attributes συνεπάγεται και παρόμοια δομή της εξαγόμενης πληροφορίας. Π.χ για το attribute ΔΙΕΥΘΥΝΣΗ εξετάζοντας τις παρακάτω εγγραφές

37 Χαριλάου Τρικούπη
23 Λεωφόρος Παπάγου
34 Αιγινήτου
12 Λαοδίκειας

μπορούμε να δούμε ότι εμφανίζονται τα patterns INTEGER CAPITALIZED και INTEGER CAPITALIZED CAPITALIZED.Ο αλγόριθμος που υλοποιήθηκε για την εξεύρεση των patterns αποτελεί μία παραλλαγή του αλγόριθμου DATAPROG ([LMK03]). Ο αλγόριθμος αυτός δέχεται σαν είσοδο ένα σύνολο από ακολουθίες tokens και παράγει στην έξοδο του μία ή και περισσότερες ακολουθίες από τύπους tokens οι οποίες περιγράφουν το πώς ξεκινάει συνήθως μία από τις ακολουθίες τις εισόδους. Ισοδύναμα, η έξοδος του αλγόριθμου είναι το σύνολο των starting patterns.

Ο αλγόριθμος βασίζεται στον έλεγχο των μηδενικών υποθέσεων (null hypothesis testing). Πιο συγκεκριμένα ο αλγόριθμος αρχικά υπολογίζει την μέση τιμή της τυχαίας μεταβλητής «πλήθος tokens ανά εγγραφή» και με βάση αυτή την τιμή προκύπτει το πλήθος των θέσεων που θεωρούνται αρχή του attribute (starting tokens) ή ισοδύναμα το μήκος των starting patterns. Στη συνέχεια κατασκευάζει τη ρίζα ενός δένδρου, από το οποίο θα παραχθούν τα starting patterns. Έπειτα υποθέτει: «Στη θέση 1 δεν είναι στατιστικά σημαντικός ο τύπος x». Ουσιαστικά κάνει 8 υποθέσεις αφού υπάρχουν 8 τύποι tokens. Ας θεωρήσουμε λοιπόν την μία εκ των 8 υποθέσεων, ότι ο τύπος CAPITALIZED δεν είναι σημαντικός στην θέση 1. Αν έχουμε n εγγραφές η τυχαία μεταβλητή «πλήθος εμφανίσεων του CAPITALIZED στη θέση 1» ακολουθεί τη διωνυμική κατανομή. Όπως προκύπτει από το κεντρικό οριακό θεώρημα μπορούμε να προσεγγίσουμε τη διωνυμική κατανομή με την κανονική. Στη συνέχεια με βάση το πλήθος των εγγραφών n,το πλήθος k των εμφανίσεων του τύπου CAPITALIZED στη θέση 1 και την πιθανότητα p=$\frac{1}{8}$ κάνουμε z-test με παραμέτρους $\mu = np = \frac{n}{8}$ και $\sigma^2 = np(1-p) = \frac{7n}{64}$.

Αν προκύψει ότι ο τύπος CAPITALIZED είναι στατιστικά σημαντικός τότε προστίθεται ένα παιδί στη ρίζα του δένδρου που περιέχει τον τύπο CAPITALIZED. Ομοίως γίνονται και οι υπόλοιποι έλεγχοι των άλλων μηδενικών υποθέσεων. Για κάθε στατιστικά σημαντικό τύπο προστίθεται ένα παιδί στη ρίζα. Έτσι σε βάθος 1 στο δένδρο υπάρχουν όλοι οι στατιστικά σημαντικοί τύποι της θέσης 1. Αν δεν προκύψει κανένας στατιστικά σημαντικός τύπος στη θέση 1 ή αν βρεθεί ότι στη θέση 1 οι στατιστικά σημαντικοί τύποι tokens είναι HTML τύποι (HTML tags, HTML attributes, HTML attribute values) κάτι που μπορεί να εμφανιστεί στην περίπτωση που έχει χαλάσει ο wrapper προστίθεται στη ρίζα ένα παιδί που περιέχει τον wildcard χαρακτήρα δηλώνοντας ότι δεν υπάρχει κάποιος από τους τύπους που μας ενδιαφέρουν στατιστικά σημαντικός στη θέση 1. Ο λόγος που δε σταματάμε την προσπάθεια

εύρεσης starting pattern σε αντίθεση με τον DATAPROG όταν δεν βρεθεί κάτι στατιστικά σημαντικό είναι ότι θέλουμε να δούμε τι στατιστικά σημαντικό υπάρχει στο πλήθος των θέσεων που θεωρείται αρχή του attribute. Έστω λοιπόν ότι το σύνολο των τύπων που βρέθηκαν να είναι σημαντικοί στη θέση 1 είναι το σύνολο $T = \{t_i\}\ 1 \le i \le n, n \le 8$.Στη συνέχεια αναζητούμε τους τύπους x για τους οποίους το μοτίβο $t_i x, 1 \le i \le n$ είναι στατιστικά σημαντικό. Οι μηδενικές υποθέσεις τώρα διαμορφώνονται ως εξής :«Το pattern $t_i x$ δεν είναι στατιστικά σημαντικό» $1 \le i \le n, x \in TT$. Έτσι ουσιαστικά γίνονται $n|TT|$ έλεγχοι μηδενικών υποθέσεων. Ο έλεγχος για το $t_i\ x_j$ γίνεται με βάση τη πληθικότητα του συνόλου των εγγραφών που ξεκινούν με τον τύπο t_i, το πλήθος των εγγραφών που ξεκινούν με τους τύπους $t_i x_j$ και την πιθανότητα εμφάνισης του τύπου x_j μετά τον τύπο t_i. Επειδή η εμφάνιση του x_j θεωρείται ανεξάρτητη από οτιδήποτε άλλο, η πιθανότητα εμφάνισής του θα είναι πάντα ίση με p=1/8. Για κάθε τύπο x_j που θα διαπιστωθεί ότι είναι σημαντικός μετά από το τύπο t_i ο αλγόριθμος δημιουργεί ένα παιδί για τον κόμβο του t_i με περιεχόμενο τον τύπο x_j.Έτσι δημιουργείται ένα δένδρο βάθους 2 στο οποίο αν κάνουμε μία κατά βάθος διάσχιση παίρνουμε όλα τα σημαντικά patterns μήκους 2. Αυτή η διαδικασία συνεχίζεται μέχρι να κατασκευαστεί ένα δένδρο βάθους ίσο με το πλήθος των starting tokens. Έπειτα κάνουμε εξαγωγή αυτών των patterns από το δένδρο έχοντας προσδιορίσει έτσι τα starting patterns. Στη συνέχεια ο αλγόριθμος κατασκευάζει ένα διάνυσμα μεταπληροφορίας για το attribute ATTR. Ας θεωρήσουμε ότι η δομή του attribute ATTR περιγράφεται από τα n patterns του συνόλου $P = \{p_1, p_2, .., p_n\}$ με $p_i \ne p_j \forall i, j$. Έστω k το συνολικό πλήθος tuples για το ATTR και έστω k_i το πλήθος των tuples που ακολουθούν το pattern p_i. Προφανώς ισχύει $\sum_{i=1}^{n} k_i \le k$. Τότε το διάνυσμα μεταπληροφορίας θα έχει την εξής μορφή:

< ATTR, k, n, $p_1, k_1, p_2, k_2, ..., p_i, k_i, p_n, k_n$, *digDen, upperDen , lowerDen, punctDen, averTokenLength, averNumOfTokensPerLine*>

όπου

digDen: μέση πυκνότητα ψηφίων,

upperDen: μέση πυκνότητα κεφαλαίων γραμμάτων,

lowerDen: μέση πυκνότητα πεζών γραμμάτων,

punctDen: μέση πυκνότητα χαρακτήρων στίξης,

averTokenLength: μέσο μήκος λεκτικών μονάδων,

averNumOfTokensPerTuple:μέσο πλήθος λεκτικών μονάδων ανά εγγραφή.

Παρακάτω βλέπουμε τον ψευδοκώδικα του αλγόριθμου που υλοποιήθηκε. Αυτή η διαδικασία κατασκευής διανυσμάτων μεταπληροφορίας γίνεται για το training attribute αλλά και για το testing attribute. Αυτά τα δύο διανύσματα τα οποία θα τα ονομάζουμε trainingVector και testingVector αντίστοιχα είναι η είσοδος για το δεύτερο μέρος του wrapper verification module.

```
PatternLearningAlgorithm(Attribute ATTR)
Για το  Attribute ATTR
 Έστω η τυχαία μεταβλητή X ← πλήθος tokens ανά εγγραφή
 μ0 ← μέση τιμή(X)
 n ← FIRSTTOKENS( μ0 )
 Δημιούργησε τη ρίζα του δένδρου
 for i ←1 to n
    {s1,s2,..,sm(i)} ← getNodesInDepth(i-1)//αν i=1 τότε επίστρεψε τη ρίζα
     for j ←1 to m(i)
          boolean somethingImportantExists ← false
          for each token type t ∈ TT
             t.pattern ← concatenate(sj, t)
             t.examples ← από το sj.examples εκείνες οι εγγραφές που μετά το  sj ακολουθεί τύπος t
             if(statisticallySignificant(|sj.examples|,|t.examples|,πιθανότητα εμφάνισης του t))
                   προσθέσε ένα παιδί στο κόμβο του sj για τον τύπο t
                   somethingImportantExists ← true
    if(somethingImportantExists=false)
            προσθέσε ένα παιδί στο κόμβο του sj με τον wildcard τύπο *
  Κάνε εξαγωγή των patterns από το δένδρο που κατασκεύασες
```

Βήμα 2ο :Έλεγχος Pearson (goodness of fit method)

Το δεύτερο μέρος βασίζεται στον έλεγχο καλής προσαρμογής (goodness of fit test) του Pearson. Θεωρούμε λοιπόν την μεταβλητή του Pearson q που αρχικοποιείται στην τιμή μηδέν και την μεταβλητή των βαθμών ελευθερίας του ελέγχου που επίσης αρχικοποιείται στο μηδέν. Για ευκολία το σύνολο τιμών {digDen, upperDen, lowerDen, punctDen, averTokenLength, averNumOfTokensPerLine} θα συμβολίζεται ως $\{x_1,..,x_6\}$. Έπειτα εκτελείται το παρακάτω τμήμα ψευδοκώδικα:

for i ←1 to 6

$$q \leftarrow q + \frac{(x_i(ver)-x_i(test))^2}{x_i(ver)}$$

freedomDegrees ← freedomDegrees+1

όπου $x_i(ver)$ η τιμή x_i για το trainingVector και $x_i(test)$ η τιμή x_i για το testingVector. Υπολογίζεται από την κατανομή χι τετράγωνο η κρίσιμη τιμή X του ελέγχου Pearson με βάση τους βαθμούς ελευθερίας που προέκυψαν μείον 1 και το επίπεδο σημαντικότητας

(significance level) το οποίο τίθεται ίσο με 5% (α=0.05). Αν η μεταβλητή q είναι μεγαλύτερη από αυτή την τιμή τότε το σύστημα δείχνει ένα μήνυμα στο χρήστη ότι ο wrapper δεν εξάγει σωστά την πληροφορία γι αυτό το attribute. Αν όμως η μεταβλητή q δεν είναι μεγαλύτερη από την κρίσιμη τιμή τότε το σύστημα εκτελεί το $3^{ο}$ βήμα.

Βήμα $3^{ο}$: Σύστημα ποινών (penalty system)

Αυτό το σύστημα βασίζεται στα patterns που έχουν παραχθεί για το training και για το testing attribute αντίστοιχα. Η λογική του είναι ότι τα patterns του training και του testing πρέπει να σχετίζονται. Αυτό γιατί με βάση την υπόθεση που κάνει το σύστημα, ότι ο wrapper εξάγει σωστά την πληροφορία για το attribute που ελέγχουμε, συνεπάγεται όπως έχουμε προαναφέρει ότι το training και το testing attribute έχουν ίδια σημασιολογία που σημαίνει και παρόμοια patterns. Ο λόγος που το σύστημα αυτό ονομάζεται σύστημα ποινών είναι ότι η μεταβλητή του Pearson αυξάνεται χωρίς όμως να αυξάνονται και οι βαθμοί ελευθερίας. Πριν γίνει παρουσίαση των κανόνων του συστήματος ποινών περιγράφονται διάφορες έννοιες τις οποίες χρησιμοποιεί το σύστημα για να μπορεί να επιβάλλει όσο το δυνατόν λογικότερες ποινές.

Το σύστημα ποινών θεωρεί τρεις ομάδες τύπων tokens. Κάθε ομάδα περιέχει εκείνους τους τύπους που θεωρεί συσχετιζόμενους. Αυτό σημαίνει ότι στη θέση ενός τύπου θα μπορούσε να εμφανιστεί ένας άλλος τύπος από την ίδια ομάδα με μεγαλύτερη πιθανότητα σε σχέση με έναν άλλο τύπο από κάποια άλλη ομάδα. Οι ομάδες είναι οι εξής:

Ομάδα1={“*”,”ALPHANUM”},

Ομάδα2={“ALPHABETIC”,”ALLUPPERCASE”,”ALLLOWERCASE”,”CAPITALIZED”, ”*”} και Ομάδα3={“INTEGER”,”DECIMAL”,”*”}

Αν εξαιρέσουμε τον wildcard χαρακτήρα που όπως είναι φυσικό υπάρχει και στις τρεις ομάδες, οι ομάδες ανά δύο έχουν το κενό σύνολο σαν τομή. Ο τύπος της στίξης δεν ανήκει σε καμία ομάδα. Δύο token τύποι θα ονομάζονται συσχετιζόμενοι αν και μόνο αν ανήκουν στην ίδια ομάδα. Δύο patterns θεωρούνται ότι είναι συσχετιζόμενα όταν το πλήθος των token τύπων σε αντίστοιχες θέσεις των patterns που είναι συσχετιζόμενοι είναι μεγαλύτερο από ένα συγκεκριμένο αριθμό ο οποίος είναι συνάρτηση του πλήθους των tokens του pattern με το μικρότερο πλήθος tokens. Όσο αυξάνεται το πλήθος αυτό, τόσο μειώνεται (αναλογικά) το απαιτούμενο πλήθος συσχετιζόμενων τύπων σε αντίστοιχες θέσεις. Με αυτό τον τρόπο αποκτάμε τη δυνατότητα μιας «σημασιολογικής» σύγκρισης patterns. Επίσης ας θεωρήσουμε δύο σύνολα από patterns $P_1 = \{p_{11},..,p_{1m}\}, P_2 = \{p_{21},..,p_{2n}\}$. Τα P_1, P_2 θα θεωρούνται συσχετιζόμενα αν κάθε pattern στο P_1 είναι συσχετιζόμενο με ένα τουλάχιστον pattern στο P_2 και επίσης κάθε pattern στο P_2 είναι συσχετιζόμενο με ένα τουλάχιστον pattern στο P_1. Η ιδέα από την οποία «γεννήθηκαν» τα συσχετιζόμενα patterns και τα συσχετιζόμενα σύνολα patterns είναι πως όταν βλέπουμε κάποια διαφορετικά patterns στο testing attribute από αυτά

του training attribute και είναι συσχετιζόμενα μεταξύ τους τότε η ποινή θα είναι μικρότερη από αυτή που θα επιβάλλαμε αν δεν ήταν συσχετιζόμενα. Κατά την υλοποίηση, η περίπτωση που το πλήθος tokens των training και των testing patterns δεν είναι ίδιο αντιμετωπίστηκε περικόπτοντας από εκείνο το σύνολο με το μεγαλύτερο πλήθος tokens για κάθε pattern τόσα tokens μέχρι να φτάσουμε το πλήθος tokens που είχαν τα patterns του άλλου συνόλου. Έτσι και πετυχαίνουμε την 1-1 σύγκριση τύπων αλλά και δεν χάνουμε πληροφορία αφού υπάρχει το γνώρισμα μέσο πλήθος tokens ανά εγγραφή. Τέλος, το σύστημα κάνει μία εκτίμηση του απαιτούμενου πλήθους εγγραφών (tuples) που πρέπει να έχει το training attribute ώστε να είναι σχεδόν βέβαιο ότι όσα patterns μπορεί να υπάρξουν γι αυτό το attribute έχουν γίνει αντιληπτά κατά τη διάρκεια της κατασκευής του διανύσματος μεταπληροφορίας. Αν και υπάρχουν 9 τύποι tokens και άρα για ένα πλήθος n tokens θα μπορούσαμε να έχουμε 9^n διαφορετικά patterns θεωρούμε ότι το απαιτούμενο πλήθος tuples είναι ίσο με $8*5^n$. Αν και η παράμετρος αυτή διαδραματίζει πρακτικά κάποιο ρόλο για σχετικά μικρά n, η ιδέα της ικανοποιεί την διαίσθηση μας ότι όσες περισσότερες εγγραφές βλέπουμε κατά την εκπαίδευση του συστήματος τόσο αυξάνεται η πεποίθηση μας ότι έχουμε δει τα περισσότερα patterns που θα μπορούσαν να υπάρχουν για το attribute. Άρα αν έχουμε αυξημένη πεποίθηση ότι έχουμε δει τα περισσότερα patterns κατά την εκπαίδευση τότε μπορούμε να τιμωρήσουμε περισσότερο τον wrapper αν στο testing attribute δούμε διαφορετικά patterns από ότι αν δεν ήμασταν σίγουροι. Τώρα μπορούμε να παρουσιάσουμε το σύνολο των κανόνων του συστήματος ποινών. Θεωρούμε τις διάφορες περιπτώσεις που μπορούμε να λάβουμε συγκρίνοντας τα σύνολα patterns P_{ver}, P_{test} που προκύπτουν από την επαληθευμένη για το attribute πληροφορία (verified) και την προς έλεγχο πληροφορία (testing) αντίστοιχα. Συγκεκριμένα το σύνολο των κανόνων που συνιστούν το σύστημα ποινών φαίνεται στο παρακάτω πλαίσιο:

1. P_{ver}, P_{test} ταυτίζονται ή $P_{test} \subseteq P_{ver}$

   Τότε δε δίνεται καμία ποινή στη μεταβλητή του Pearson. Αυτό γιατί νιώθουμε βέβαιοι ότι το testing attribute έχει την ίδια σημασιολογία με το training attribute.

2. $P_{ver} \subseteq P_{test}$

   Το σύστημα εμφανίζει ένα προειδοποιητικό σύστημα στο χρήστη ζητώντας του να δώσει είσοδο για το αν θέλει να αλλάξει τους ρόλους των δύο διανυσμάτων. Αυτό γίνεται γιατί υπάρχει η περίπτωση το σύνολο παραδειγμάτων εκπαίδευσης να μην είναι αρκετά μεγάλο ώστε να εμφανιστούν όλα τα πιθανά patterns του attribute. Αν ο χρήστης επιλέξει την εναλλαγή των ρόλων τότε αναγόμαστε στην περίπτωση 1. Αν όμως επιλέξει να διατηρήσει τους ρόλους των δύο διανυσμάτων τότε υπολογίζονται αρχικά οι παρακάτω παράμετροι:

$a = 1 - \frac{|P_{ver}|}{|P_{test}|}$ που εκφράζει το ποσοστό των patterns του P_{test} που δεν είναι κοινά με τα patterns του P_{ver},

$b = \frac{\#tuples\ testing - \#tuples\ testing\ \text{που καλύπτονται από το}\ P_{ver}}{\#tuples\ testing}$ που εκφράζει το ποσοστό των tuples του testing attribute που δεν εκφράζονται από τα patterns του P_{ver}, δηλαδή τα κοινά patterns. Διακρίνονται οι εξής υποπεριπτώσεις :

α)Αν $a \succ b$ και τα P_{ver}, P_{test} είναι συσχετιζόμενα τότε η μεταβλητή q αυξάνεται κατά 0.05X (υπενθυμίζουμε ότι X είναι η κρίσιμη τιμή που αν υπερβεί η μεταβλητή του Pearson το σύστημα αποφαίνεται ότι ο wrapper έχει πρόβλημα).

β)Αν $a \prec b$ και τα P_{ver}, P_{test} είναι συσχετιζόμενα τότε η μεταβλητή q αυξάνεται κατά 0.2X.

γ)Αν τα P_{ver}, P_{test} δεν είναι συσχετιζόμενα τότε υπολογίζουμε αρχικά το ποσοστό των patterns του P_{test} τα οποία δε σχετίζονται με κανένα από τα patterns του P_{ver}.Έστω ότι αυτή η τιμή είναι ίση με r.Τότε η μεταβλητή q αυξάνεται κατά (r+0.4)X.

3. $P_{ver} \cap P_{test} = \varnothing$

Διακρίνουμε τις ακόλουθες περιπτώσεις:

α)Αν τα P_{ver}, P_{test} συσχετιζόμενα τότε αν κάποιο από τα δύο έχει πληθικότητα ίση με 1 τότε η μεταβλητή q αυξάνεται κατά 0.75X αλλιώς αυξάνεται κατά 0.9X.

β) Αν τα P_{ver}, P_{test} δεν είναι συσχετιζόμενα τότε η μεταβλητή q αυξάνεται κατά X θεωρώντας ότι ο wrapper δεν δουλεύει σωστά πλέον.

4. $P_{ver} \cap P_{test} = P_{common}$

Έστω ότι τα σύνολα P_{ver}, P_{test} έχουν μη κενή τομή το σύνολο P_{common}. Υπολογίζουμε την παράμετρο $a = \frac{|P_{common}|}{|P_{test}|}$ που εκφράζει το ποσοστό των patterns του P_{test} που είναι κοινά με το P_{ver} και την παράμετρο

$b = \frac{\#tuples\ testing - \#tuples\ testing\ \text{που καλύπτονται από το}\ P_{common}}{\#tuples\ testing}$ που εκφράζει το ποσοστό των εγγραφών του P_{test} που δεν καλύπτονται από τα patterns του P_{common}. Η περίπτωση αυτή διασπάται σε πολλές υποπεριπτώσεις για την καλύτερη αντιμετώπισή της. Οι διάφορες υποπεριπτώσεις εξαρτώνται από τις πληθικότητες των συνόλων $P_{ver}, P_{test}, P_{common}$, τις συσχετίσεις μεταξύ των patterns, τις παραπάνω

παραμέτρους καθώς και τη βεβαιότητα μας για το αν έχουμε δει τα patterns κατά την εκπαίδευση του συστήματος. Για συντομία, αν το πλήθος των εγγραφών του training attribute είναι τέτοιο ώστε να μας εξασφαλίζει με βάση τις υποθέσεις μας ότι έχουμε δει τα περισσότερα patterns θα γράφουμε *vevaioi* ενώ αν δεν είμαστε σίγουροι $\neg$*vevaioi*

α)$|P_{common}|=1, |P_{ver}|=|P_{test}|=2$

Αν τα P_{ver}, P_{test} είναι συσχετιζόμενα τότε διακρίνουμε τις εξής υποπεριπτώσεις:

i)Αν $b \le a$ και $\neg$*vevaioi* τότε η μεταβλητή q αυξάνεται κατά 0.1X.

ii) Αν $b \le a$ και *vevaioi* τότε η μεταβλητή q αυξάνεται κατά 0.5X.

iii) Αν $a \prec b$ και $\neg$*vevaioi* τότε η μεταβλητή q αυξάνεται κατά 0.3X.

iv) Αν $a \prec b$ και *vevaioi* τότε η μεταβλητή q αυξάνεται κατά 0.6X.

Αν τα P_{ver}, P_{test} δεν είναι συσχετιζόμενα τότε η μεταβλητή q αυξάνεται κατά 0.8X.

β)$|P_{common}|=1, |P_{ver}|=3, |P_{test}|=2$

Αν τα P_{ver}, P_{test} είναι συσχετιζόμενα τότε διακρίνουμε τις εξής υποπεριπτώσεις:

i)Αν $b \le a$ και $\neg$*vevaioi* τότε η μεταβλητή q αυξάνεται κατά 0.25X.

ii) Αν $b \le a$ και *vevaioi* τότε η μεταβλητή q αυξάνεται κατά 0.6X.

iii) Αν $a \prec b$ και $\neg$*vevaioi* τότε η μεταβλητή q αυξάνεται κατά 0.4X.

iv) Αν $a \prec b$ και *vevaioi* τότε η μεταβλητή q αυξάνεται κατά 0.8X.

Αν τα P_{ver}, P_{test} δεν είναι συσχετιζόμενα τότε η μεταβλητή q αυξάνεται κατά 0.9X.

γ) $|P_{common}|=1, |P_{ver}|=2, |P_{test}|=3$

Αν τα P_{ver}, P_{test} είναι συσχετιζόμενα τότε διακρίνουμε τις εξής υποπεριπτώσεις:

i)Αν $b \le a$ και $\neg$*vevaioi* τότε η μεταβλητή q αυξάνεται κατά 0.25X.

ii) Αν $b \le a$ και *vevaioi* τότε η μεταβλητή q αυξάνεται κατά 0.65X.

iii) Αν $a \prec b$ και $\neg$*vevaioi* τότε η μεταβλητή q αυξάνεται κατά 0.4X.

iv) Αν $a \prec b$ και *vevaioi* τότε η μεταβλητή q αυξάνεται κατά 0.85X.

Αν τα P_{ver}, P_{test} δεν είναι συσχετιζόμενα τότε η μεταβλητή q αυξάνεται κατά 0.9X.

δ) $|P_{common}|=1, |P_{ver}|\ge 3, |P_{test}|\ge 3$

Αν τα P_{ver}, P_{test} είναι συσχετιζόμενα τότε διακρίνουμε τις εξής υποπεριπτώσεις:

i)Αν $b \le a$ και $\neg$*vevaioi* τότε η μεταβλητή q αυξάνεται κατά 0.25X.

ii) Αν $b \le a$ και *vevaioi* τότε η μεταβλητή q αυξάνεται κατά 0.6X.

iii) Αν $a \prec b$ και $\neg$*vevaioi* τότε η μεταβλητή q αυξάνεται κατά 0.6X.

iv) Αν $a \prec b$ και *vevaioi* τότε η μεταβλητή q αυξάνεται κατά 0.85X.

Αν τα P_{ver}, P_{test} δεν είναι συσχετιζόμενα τότε υπολογίζουμε το ποσοστό των patterns του συνόλου με την μικρότερη πληθικότητα το οποίο δε σχετίζεται με κανένα pattern από το άλλο σύνολο. Έστω r αυτή η τιμή. Τότε η μεταβλητή q αυξάνεται κατά (0.5+r)X.

ε) $|P_{common}|=2, |P_{ver}|=3, |P_{test}|=3$

Αν τα P_{ver}, P_{test} είναι συσχετιζόμενα τότε διακρίνουμε τις εξής υποπεριπτώσεις:

i)Αν $b \le a$ και $\neg vevaioi$ τότε η μεταβλητή q αυξάνεται κατά 0.25X.

ii) Αν $b \le a$ και $vevaioi$ τότε η μεταβλητή q αυξάνεται κατά 0.6X.

iii) Αν $a \prec b$ και $\neg vevaioi$ τότε η μεταβλητή q αυξάνεται κατά 0.6X.

iv) Αν $a \prec b$ και $vevaioi$ τότε η μεταβλητή q αυξάνεται κατά 0.85X.

Αν τα P_{ver}, P_{test} δεν είναι συσχετιζόμενα τότε υπολογίζουμε το ποσοστό των patterns του συνόλου με την μικρότερη πληθικότητα το οποίο δε σχετίζεται με κανένα pattern από το άλλο σύνολο. Έστω r αυτή η τιμή. Τότε η μεταβλητή q αυξάνεται κατά (0.5+r)X.

στ) $|P_{common}|=2, |P_{ver}|=4, |P_{test}|=3$

Αν τα P_{ver}, P_{test} είναι συσχετιζόμενα τότε διακρίνουμε τις εξής υποπεριπτώσεις:

i)Αν $b \le a$ και $\neg vevaioi$ τότε η μεταβλητή q αυξάνεται κατά 0.2X.

ii) Αν $b \le a$ και $vevaioi$ τότε η μεταβλητή q αυξάνεται κατά 0.5X.

iii) Αν $a \prec b$ και $\neg vevaioi$ τότε η μεταβλητή q αυξάνεται κατά 0.5X.

iv) Αν $a \prec b$ και $vevaioi$ τότε η μεταβλητή q αυξάνεται κατά 0.75X.

Αν τα P_{ver}, P_{test} δεν είναι συσχετιζόμενα τότε υπολογίζουμε το ποσοστό των patterns του συνόλου με την μικρότερη πληθικότητα το οποίο δε σχετίζεται με κανένα pattern από το άλλο σύνολο. Έστω r αυτή η τιμή. Τότε η μεταβλητή q αυξάνεται κατά (0.6+r)X.

ζ) $|P_{common}|=2, |P_{ver}|=3, |P_{test}|=4$

Αν τα P_{ver}, P_{test} είναι συσχετιζόμενα τότε διακρίνουμε τις εξής υποπεριπτώσεις:

i)Αν $b \le a$ και $\neg vevaioi$ τότε η μεταβλητή q αυξάνεται κατά 0.25X.

ii) Αν $b \le a$ και $vevaioi$ τότε η μεταβλητή q αυξάνεται κατά 0.55X.

iii) Αν $a \prec b$ και $\neg vevaioi$ τότε η μεταβλητή q αυξάνεται κατά 0.6X.

iv) Αν $a \prec b$ και $vevaioi$ τότε η μεταβλητή q αυξάνεται κατά 0.8X.

Αν τα P_{ver}, P_{test} δεν είναι συσχετιζόμενα τότε υπολογίζουμε το ποσοστό των patterns του συνόλου με την μικρότερη πληθικότητα το οποίο δε σχετίζεται με κανένα pattern από το άλλο σύνολο. Έστω r αυτή η τιμή. Τότε η μεταβλητή q αυξάνεται κατά

(0.65+r)X.

η) $|P_{common}| = 2, |P_{ver}| \succ 4, |P_{test}| \succ 4$

Ακριβώς το ίδιο σύστημα κανόνων με την περίπτωση ζ.

θ)Οποιαδήποτε περίπτωση που δεν καλύπτεται από τις περιπτώσεις α) έως και η)

Υπολογίζουμε τις εξής τιμές :

$c_1 = \frac{\max(|P_{ver}|,|P_{test}|) - |P_{common}|}{\max(|P_{ver}|,|P_{test}|)}$ και $c_2 = \frac{\min(|P_{ver}|,|P_{test}|) - |P_{common}|}{\min(|P_{ver}|,|P_{test}|)}$.

Αν τα P_{ver}, P_{test} είναι συσχετιζόμενα τότε διακρίνουμε τις εξής υποπεριπτώσεις:

i)Αν $b \le a$ και $\neg vevaioi$ τότε η μεταβλητή q αυξάνεται κατά $\min(c_1, c_2)$X.

ii) Αν $b \le a$ και $vevaioi$ τότε η μεταβλητή q αυξάνεται κατά $\max(c_1, c_2)$X.

iii)Αν $a \prec b$ και $\neg vevaioi$ τότε η μεταβλητή q αυξάνεται κατά $(\min(c_1, c_2)+0.1)$X

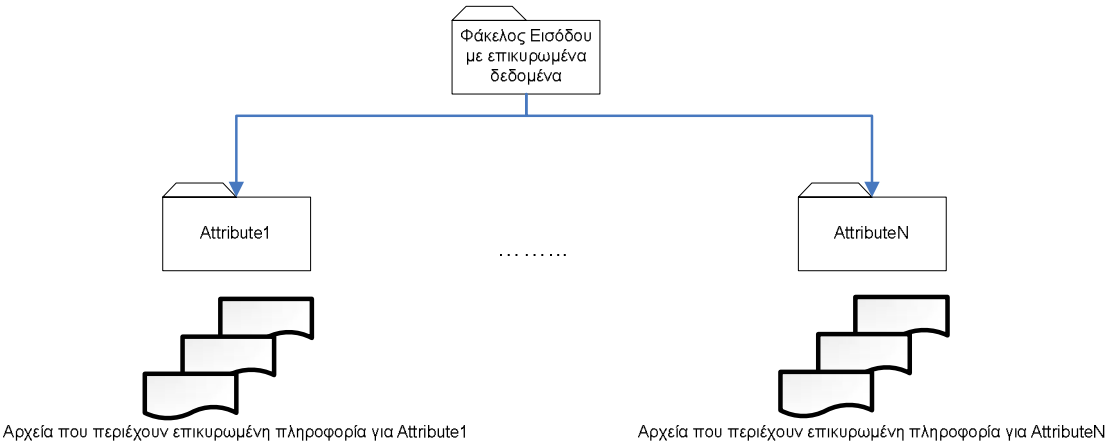
iv) Αν $a \prec b$ και $vevaioi$ τότε η μεταβλητή q αυξάνεται κατά $(\max(c_1, c_2)+0.1)$X.

Αν τα P_{ver}, P_{test} δεν είναι συσχετιζόμενα τότε η μεταβλητή q αυξάνεται κατά $(c_1 + c_2)$X.

4.2 Σύστημα επαγωγής ορθού wrapper (Wrapper Reinduction system)

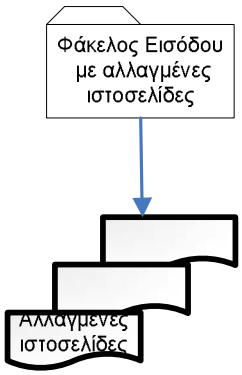
Όταν το wrapper verification module ειδοποιήσει το χρήστη ότι υπάρχει πρόβλημα στην εξαγωγή της πληροφορίας για ένα ή και περισσότερα attributes πρέπει να βρεθούν παραδείγματα πληροφορίας για κάθε attribute στις αλλαγμένες σελίδες. Η ιδέα του συστήματος που υλοποιήθηκε είναι ίδια με αυτήν που περιγράφηκε στην παράγραφο 3.2.2. Η υλοποίηση ήταν αρκετά πιο απλή από αυτή που περιγράφηκε. Ο κύριος λόγος γι αυτό ήταν η άμεση εξάρτηση του module αυτού (σε αντίθεση με το verification module) από το σύστημα επαγωγής wrappers που είχαμε στη διάθεσή μας. Το συγκεκριμένο Wrapper Induction σύστημα που είχαμε στη διάθεση μας υλοποιήθηκε στα πλαίσια των ερευνητικών δραστηριοτήτων του ερευνητικού ιδρύματος «ΔΗΜΟΚΡΙΤΟΣ» και βασίζεται στον αλγόριθμο STALKER. Όμως σε αντίθεση με τον αλγόριθμο STALKER που περιγράψαμε στην παράγραφο 3.1.3 η υλοποίησή που διαθέταμε αποτελούσε μια παραλλαγή του που δεν μπορούσε να κάνει χρήση της ιεραρχικής δομής του εγγράφου (embedded catalog) με συνέπεια αντί να είναι ένας multi-slot extractor να είναι ένας single-slot extractor. Αυτό πρακτικά σημαίνει ότι ο τύπος των δεδομένων σε δενδρική δομή θα ήταν ένα δένδρο όπως αυτό του σχήματος 3.9. Αυτό απλοποίησε τις απαιτήσεις του συστήματος που θα έπρεπε να

υλοποιήσουμε. Το reinduction module παίρνει σαν είσοδο ένα σύνολο από εγγραφές που εξάχθηκαν κατά τη διάρκεια της ορθής λειτουργίας του wrapper.Το σύστημα στην πραγματικότητα δέχεται σαν είσοδο δύο διευθύνσεις φακέλων. Η πρώτη είναι η διεύθυνση ενός φακέλου ο οποίος περιέχει διάφορους υποφακέλους που αντιπροσωπεύουν attributes.Κάθε τέτοιος υποφάκελος περιέχει ένα σύνολο εγγραφών για το attribute που αντιπροσωπεύει που ελήφθησαν κατά τη διάρκεια της ορθής λειτουργίας του wrapper. Και πάλι θεωρούμε ότι κάθε γραμμή των αρχείων βρίσκεται μία εγγραφή. Στο σχήμα 4.4 βλέπουμε τη δομή του φακέλου με τα επικυρωμένα δεδομένα που δίνεται σαν είσοδος στο σύστημα.



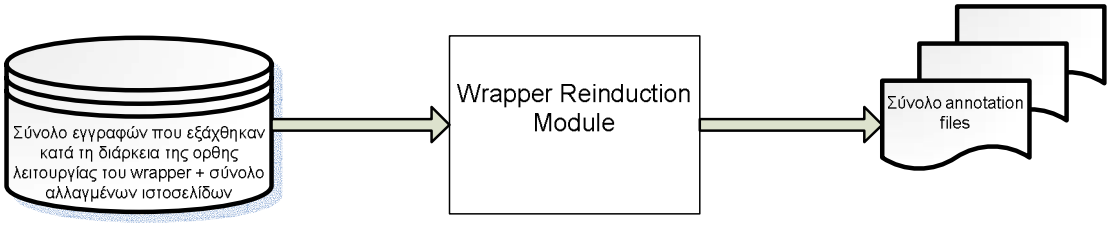
σχήμα 4.4: Δομή φακέλου εισόδου για το reinduction module με την επαληθευμένη πληροφορία

Ο δεύτερος φάκελος περιέχει τις αλλαγμένες πλέον ιστοσελίδες (αρχεία html,htm) στα οποία θα γίνει αναζήτηση της πληροφορίας.



σχήμα 4.5: Δομή φακέλου εισόδου για το reinduction module με τις αλλαγμένες ιστοσελίδες

Στη συνέχεια το σύστημα κάνει αναζήτηση των εγγραφών που δόθηκαν μέσα στις αλλαγμένες ιστοσελίδες. Η είσοδος και η έξοδος του συστήματος φαίνεται στο παρακάτω σχήμα.



σχήμα 4.5: Μακροσκοπική όψη του reinduction module

Στην έξοδο παίρνουμε ένα σύνολο αρχείων κειμένου (txt αρχεία) τα οποία δημιουργούνται μέσα στο φάκελο που περιέχει τις ιστοσελίδες και τα οποία ονομάζονται annotation files. Για κάθε ιστοσελίδα μέσα στο φάκελο δημιουργείται και ένα τέτοιο αρχείο το οποίο έχει το ίδιο όνομα με την αντίστοιχη ιστοσελίδα. Η μορφή αυτού του αρχείου είναι προκαθορισμένη και είναι η ζητούμενη μορφή της εισόδου για το wrapper induction σύστημα που διαθέταμε. Στην πρώτη γραμμή του αρχείου βρίσκεται η διεύθυνση της ιστοσελίδας στην οποία αντιστοιχεί το αρχείο και στη δεύτερη γραμμή το συνολικό πλήθος των χαρακτήρων μέσα στην ιστοσελίδα. Στις υπόλοιπες γραμμές βρίσκεται η πληροφορία στην εξής μορφή:

start end {data} {attribute name} {attribute name} {}{}

όπου *start* και *end* είναι δύο ακέραιοι που δηλώνουν τη θέση του πρώτου και του τελευταίου χαρακτήρα της πληροφορίας *data* μέσα στην ιστοσελίδα, *data* είναι η πληροφορία που εξάγεται ή που θέλουμε να εξάγουμε (ανάλογα με το αν βρισκόμαστε στη φάση εξαγωγής ή εκπαίδευσης αντίστοιχα) και *attribute name* το όνομα του attribute που εκφράζει η πληροφορία *data*. Αντιλαμβανόμαστε ότι τα annotation files είναι μια μορφή δομημένης πληροφορίας.

Ο αλγόριθμος που υλοποιήθηκε είναι ένας brute force αλγόριθμος αναζήτησης tokens μέσα σε HTML κείμενο. Αν και η υλοποίηση αναζητεί single slotted δεδομένα όπως απαιτεί και η υλοποίηση του STALKER υπάρχει κίνδυνος να υπάρχει η ίδια εγγραφή δύο ή και περισσότερες φορές σε διαφορετικό layout. Βλέπουμε ένα παράδειγμα:

……

2.***Apple PowerMac 9500/120 Memory Upgrade Products***

Apple PowerMac 9500/120 Memory Upgrade Products now cheaper than ever….

Το reinduction σύστημα στην παραπάνω ιστοσελίδα που θέλουμε να εξάγουμε τίτλους που μορφοποιούνται με έντονα και πλάγια γράμματα αν είχε σαν παράδειγμα αναζήτησης το “Apple PowerMac 9500/120 Memory Upgrade Products” θα έβρισκε δύο εμφανίσεις και έτσι αν το annotation file που προέκυπτε δινόταν σαν είσοδος στο wrapper induction system θα υπήρχε πρόβλημα στην μάθηση ενός σωστού wrapper. Γι αυτό το λόγο δημιουργήσαμε στο interface με τον χρήστη τη δυνατότητα να διασπά την ιστοσελίδα σε tokens βλέποντας

την αρχή και το τέλος τους και να ελέγχει αν τα παραγόμενα annotation files από το reinduction σύστημα είναι σωστά.

5

Πειραματικά Αποτελέσματα-Αξιολόγηση συστήματος ARMAGEDDON

5.1 Αξιολόγηση Wrapper Verification Module

Στην ενότητα αυτή γίνεται παρουσίαση των αποτελεσμάτων που λάβαμε κατά την αξιολόγηση του verification module.

5.1.1 Μετρικές για την αξιολόγηση

Κατά τον έλεγχο της λειτουργίας του wrapper μπορούν να συμβούν τέσσερα διαφορετικά ενδεχόμενα. Για να περιγράψουμε με συντομία τα ενδεχόμενα αυτά, θα χρησιμοποιήσουμε την πρόταση a:« Το σύστημα συμπεραίνει ότι ο wrapper λειτουργεί σωστά» και την πρόταση b:«Στην πραγματικότητα ο wrapper λειτουργεί σωστά». Τα ενδεχόμενα είναι τα εξής :

1) $a \wedge b$. Αυτό το ενδεχόμενο είναι επιθυμητό γιατί το σύστημα προβλέπει σωστά ότι ο wrapper δεν έχει πρόβλημα.
2) $a \wedge \neg b$. Αυτό το ενδεχόμενο αντιστοιχεί σε λάθος της λειτουργίας του συστήματος αφού το σύστημα δεν αντιλαμβάνεται ότι ο wrapper εξάγει λάθος δεδομένα.
3) $\neg a \wedge b$. Και αυτή η περίπτωση αντιστοιχεί σε σφάλμα της λειτουργίας του συστήματος. Εδώ το σύστημα συμπεραίνει λανθασμένα ότι ο wrapper δεν δουλεύει σωστά.
4) $\neg a \wedge \neg b$. Αυτό το ενδεχόμενο αντιστοιχεί στην περίπτωση που το σύστημα αντιλαμβάνεται σωστά την λανθασμένη λειτουργία του wrapper.

Από το δεύτερο και το τρίτο ενδεχόμενο τα οποία είναι ανεπιθύμητα το πιο ανεπιθύμητο είναι το δεύτερο. Αυτό γιατί στο τρίτο ενδεχόμενο όταν εκτελεστεί ο αλγόριθμος του reinduction system θα παραχθεί ένας νέος wrapper ο οποίος θα είναι ίδιος με τον αρχικό. Αν και αυτό κοστίζει σε πόρους (resources) δεν αποτελεί θεμελιώδες πρόβλημα για την λειτουργία του συστήματος.Αντίθετα, στη δεύτερη περίπτωση ο wrapper θα συνεχίσει να εξάγει λανθασμένα δεδομένα από την αλλαγμένη πηγή.

Ας θεωρήσουμε ότι εκτελούμε συνολικά Α+Β+Γ+Δ δοκιμές. Παρακάτω βλέπουμε τα ενδεχόμενα που μπορεί να προκύψουν σε μορφή πίνακα.

	b	$\neg b$
a	Α	Β
$\neg a$	Γ	Δ

Τα Α,Β,Γ,Δ αντιστοιχούν στο πλήθος των εμφανίσεων των ενδεχομένων που προκύπτουν ως τομή των ενδεχομένων που αντιστοιχούν στη γραμμή και στη στήλη τους αντίστοιχα.

Το σύστημα θα έχει τέλεια απόδοση έχουμε όταν Β=Γ=0. Με βάση τις τιμές Α,Β,Γ,Δ ορίζουμε διάφορα μεγέθη τα οποία θα αποτελέσουν τα κριτήρια αξιολόγησης την επίδοσης του συστήματος μας. Αυτά είναι τα παρακάτω:

$$ac = accuracy = \frac{\Α + \Delta}{\Α + \Β + \Gamma + \Delta}$$

$$up = unchanged\ \ precision = \frac{\Α}{\Α + \Β}$$

$$cp = changed\ \ precision = \frac{\Delta}{\Delta + \Gamma}$$

$$ur = unchanged\ \ recall = \frac{\Α}{\Α + \Gamma}$$

$$cr = changed\ \ recall = \frac{\Delta}{\Delta + \Β}$$

$$F_{CHANGED} = \frac{2*cr*cp}{cr + cp}$$

$$F_{UNCHANGED} = \frac{2*ur*up}{ur + up}$$

5.1.2 Αποτελέσματα ARMAGEDDON

Μία αξιολόγηση του συστήματος μας ήταν αρκετά δύσκολη σε πραγματικά δεδομένα. Ο λόγος γι αυτό είναι πως οι αλλαγές που γίνονται στους διαδικτυακούς τόπους μπορεί να γίνονται ανά μεγάλα χρονικά διαστήματα. Κάτι τέτοιο δεν ήταν χρονικά εφικτό στα πλαίσια της διπλωματικής εργασίας αφού ήταν αδύνατον να γίνει επί μακρόν παρακολούθηση

διαφόρων sites. Έτσι τα δεδομένα πάνω στα οποία δοκιμάστηκε το σύστημα προήλθαν από τον Nicholas Kushmerick και είναι αυτά τα οποία χρησιμοποίησε για την αξιολόγηση του RAPTURE αλγόριθμου που περιγράφηκε στο κεφάλαιο 3. Ο Kushmerick ανά τακτά χρονικά διαστήματα για ένα περίπου εξάμηνο συγκέντρωνε από κάποια sites ένα σύνολο ιστοσελίδων οι οποίες προέκυπταν από queries που πραγματοποιούσε στα sites. Παρακάτω αναλύονται τα πειράματα που έγιναν για 3 web sites για να γίνει φανερή η λογική της εκτέλεσης των πειραμάτων και στη συνέχεια παρουσιάζονται τα αποτελέσματα που λάβαμε συγκεντρωτικά.

ALTAVISTA

Τα attributes που επιλέξαμε να εξάγουμε από τις ιστοσελίδες της πηγής altavista ήταν ο τίτλος και ο υπερσύνδεσμος (hyperlink).Τα 4 queries για τα οποία πραγματοποιήσαμε τα πειράματα μας είχαν ως λέξη αναζήτησης τις λέξεις apple,Guatemala,Chile,paprika. Στο site αυτό υπήρξαν δύο αλλαγές στο layout των ιστοσελίδων. Παρακάτω αναλύονται τα αποτελέσματα που λάβαμε για κάθε query:

1)apple

Δημιουργήσαμε έναν wrapper με βάση την πρώτη χρονικά ιστοσελίδα. Αυτός ο wrapper χρησιμοποιήθηκε για να εξαχθεί η πληροφορία από τις ιστοσελίδες που λάβαμε πριν γίνει η πρώτη αλλαγή. Το σύστημα μας πέτυχε στο να αναγνωρίσει την ορθή λειτουργία του wrapper και για τα δύο attributes. Στη συνέχεια ελέγξαμε αν ο wrapper λειτουργούσε σωστά για τις αλλαγμένες ιστοσελίδες μετά την πρώτη αλλαγή. Και πάλι το σύστημα μας κατέληξε στο συμπέρασμα ότι ο wrapper έχει χαλάσει. Μάλιστα πέτυχε στο αναγνωρίσει ότι η εξαγόμενη πληροφορία για το attribute του τίτλου ήταν σωστή ενώ το πρόβλημα βρισκόταν στο attribute του υπερσυνδέσμου. Στη συνέχεια παράχθηκε ένας νέος wrapper για τις αλλαγμένες ιστοσελίδες. Ο wrapper αυτός χρησιμοποιήθηκε για να εξάγει την επιθυμητή πληροφορία από τις ιστοσελίδες πριν την δεύτερη αλλαγή. Και πάλι το σύστημα μας αναγνώρισε σωστά την ορθή λειτουργία του wrapper. Όταν άλλαξαν οι ιστοσελίδες για δεύτερη φορά και πάλι το verification σύστημα κατάφερε να αναγνωρίσει με επιτυχία ότι ο wrapper είχε χαλάσει για το attribute του υπερσυνδέσμου ενώ λειτουργούσε σωστά για το attribute του τίτλου. Τέλος, για να κάνουμε περισσότερες δοκιμές, δοκιμάσαμε τον αρχικό wrapper στις ιστοσελίδες μετά την δεύτερη αλλαγή. Το σύστημα μας αναγνώρισε και πάλι ορθά την λάθος εξαγωγή της πληροφορίας για τον υπερσύνδεσμο και την ορθή λειτουργία για τον τίτλο. Συνεπώς σε επίπεδο αναγνώρισης λειτουργίας του wrapper ανά attribute (επίπεδο attribute) έχουμε συνοπτικά τα παρακάτω αποτελέσματα

	b	$\neg b$
a	7	0
$\neg a$	0	3

ενώ σε επίπεδο αναγνώρισης λειτουργίας του wrapper για site (επίπεδο site) έχουμε τα παρακάτω αποτελέσματα:

	b	¬*b*
a	2	0
¬*a*	0	3

2)paprika, Guatemala, Chile

Και για τα παραπάνω queries έχουμε πολύ ικανοποιητικά αποτελέσματα. Συγκεκριμένα σε όλες τις δοκιμές σε επίπεδο site είχαμε σωστές προβλέψεις και σε επίπεδο attribute είχαμε μόνο μία αποτυχία του συστήματος. Συγκεκριμένα το σύστημα προέβλεψε ότι υπάρχει πρόβλημα στην λειτουργία του wrapper για την εξαγωγή της πληροφορίας του τίτλου ενώ στην πραγματικότητα δεν υπήρχε τέτοιο πρόβλημα. Σε επίπεδο wrapper δεν δημιουργήθηκε πρόβλημα γιατί το σύστημα προέβλεψε ορθά ότι ο wrapper δεν δούλευε σωστά για τον υπερσύνδεσμο. Ο λόγος που έγινε αυτό το λάθος ήταν ότι για το συγκεκριμένο δείγμα ιστοσελίδων στο οποίο γινόταν ο έλεγχος «ξέφευγε» από όσα είχε μάθει το σύστημα κατά τη διάρκεια της εκπαίδευσης. Έτσι ενώ το σύστημα κατά την εκπαίδευση του είχε δει τίτλους όπως «Mama Goldman's Paprika Chicken, Rice Pilaf, and Green Peas» και «Empfehlungen: Paprika» στο δείγμα ελέγχου της λειτουργίας εμφανίστηκαν σελίδες με πολύ μεγαλύτερη πυκνότητα σημείων στίξης και μέσου πλήθους tokens ανά εγγραφή (π.χ εξάχθηκαν πολλοί τίτλοι όπως οι «paprika---vi ser saker som du missar----------#---------- » και «..... -Paprika; infoPepper Internet Service-»). Έτσι πριν καν εισέλθει το verification module στο σύστημα ποινών είχε προκύψει ότι ο wrapper είχε χαλάσει μιας και το γνώρισμα του μέσου πλήθους tokens ανά εγγραφή για το training attribute είχε τιμή 4.2439 ενώ για το testing attribute τιμή 10.579 με συνέπεια η αύξηση στην μεταβλητή Pearson να είναι 9.45. Σε συνδυασμό με την μεγάλη αύξηση (5.3) λόγω της διαφοράς στην πυκνότητα των σημείων στίξης προκύπτει τελικά η τιμή 16.46 για την μεταβλητή του Pearson που είναι μεγαλύτερη από την κρίσιμη τιμή 11.07.

Συνεπώς συγκεντρωτικά τα αποτελέσματα για το site altavista σε επίπεδο attribute φαίνονται στον παρακάτω πίνακα

	b	¬*b*
a	27	0
¬*a*	1	12

ενώ σε επίπεδο site :

	b	¬*b*
a	2	0
¬*a*	0	3

BIBL

Οι σελίδες αυτές προήλθαν από την ψηφιακή βιβλιοθήκη http://etext.virginia.edu/ebooks και αφορούσαν το κείμενο της Βίβλου. Το site αυτό δεν υπέστη κάποια αλλαγή. Τα attributes γι αυτό το site ήταν η πηγή (source) και το απόσπασμα (text).Τα queries είχαν σαν λέξεις αναζήτησης τα blood και faith. Τα αποτελέσματα σε επίπεδο attribute φαίνονται στον παρακάτω πίνακα:

	b	¬*b*
a	4	0
¬*a*	0	0

ενώ σε επίπεδο site:

	b	¬*b*
a	1	0
¬*a*	0	0

CDPL

Οι σελίδες προήλθαν από το site με url http://www.cd-plus.com. Τα attributes τα οποία αποτέλεσαν την επιθυμητή πληροφορία ήταν ο τίτλος του cd (cd title), η ημερομηνία (date), η κατηγορία στην οποία ανήκει το cd (category), ο διανομέας (distributor), ο κωδικός του CD (CD Plus Code) και η τιμή (price). Το site δεν άλλαξε καθόλου και έτσι θα περιμέναμε το σύστημά μας να μην βρει κανέναν πρόβλημα στην λειτουργία του wrapper. Έτσι και έγινε. Τα αποτελέσματα φαίνονται στον παρακάτω πίνακα

	b	¬*b*
a	18	0
¬*a*	0	0

ενώ σε επίπεδο λειτουργίας wrapper :

	b	¬*b*
a	1	0
¬*a*	0	0

Συνεχίζοντας με την ίδια λογική όπως περιγράφηκε για τα τρία παραπάνω sites προκύπτουν τα αποτελέσματα που φαίνονται στον παρακάτω πίνακα. Με τον δείκτη ATTRIBUTE δηλώνουμε την επίδοση του συστήματος σε επίπεδο αναγνώρισης λειτουργίας του wrapper

για εξαγωγή πληροφορίας από ένα attribute και με το δείκτη SITE την επίδοση του συστήματος σε επίπεδο αναγνώρισης λειτουργίας του wrapper για το site για το οποίο σχεδιάστηκε.

<url>,< queries >,<attributes>, $< a \wedge b , a \wedge \neg b ,$ $\neg a \wedge b , \neg a \wedge \neg b >_{\text{ATTRIBUTE}}$, $< a \wedge b , a \wedge \neg b ,$ $\neg a \wedge b , \neg a \wedge \neg b >_{\text{SITE}}$
<www.altavista.com>, <apple, Chile, Guatemala, paprika>, <τίτλος,υπερσύνδεσμος>, <27,0,1,12>,<2,0,0,3>
<etext.virginia.edu/ebooks>, <faith,blood>, <πηγή,απόσπασμα>, <4,0,0,0>,<1,0,0,0>
<www.cd-plus.com>, < Madonna, Cranberries, Beatles>, <τίτλος cd,ημερομηνία, κατηγορία, διανομέας, κωδικός cd, τιμή>, <18,0,0,0>,<1,0,0,0>
<www.shopper.com>, < CD>,< τίτλος cd, τιμή, εταιρεία>, <3,0,0,0>,<1,0,0,0>
<www.uk.lycos.de>, <apple, potato, Guatemala, happy>, <τίτλος, υπερσύνδεσμος, ποσοστό σχετικότητας αποτελέσματος με το query>, <12,0,0,12>,<2,0,0,1>
<www.news.com>, <California, IBM, Intel, Ireland, Java, jobs, Microsoft>,<τίτλος είδησης, ημερομηνία>, <14,0,0,0>,<1,0,0,0>
<www.thriveonline.com>, <diet, energy, stress >,<τίτλος>, <3,0,0,0>,<1,0,0,0>

<www.fourmilab.ch >,<Death, Income, tax>, <τίτλος>,<3,0,0,0>,<1,0,0,0>
<http://cgi.pathfinder.com/cgi-bin/Money /col.cgi?calced=calced&host=pathfinder&from= U.S.+average&to=Decatur,+Alabama& amountfromsub=>, <10001>,< Μισθός11,Μισθός12,Ποσοστιαία διαφορά 1,Τύπος διαφοράς1,Μισθός21, Μισθός22,Ποσοστιαία διαφορά 2,Τύπος διαφοράς2>,<8,0,0,0>,<1,0,0,0>
<cgi.pathfinder.com/cgi-bin/time/cg/pshell ?venue=time&pg=q&date=all&q=",> <Clinton,election,health>, <τίτλος,υπερσύνδεσμος,κείμενο>, <9,0,0,0>,<1,0,0,0>
<patents.uspto.gov> <air, glass, iron >,<τίτλος πατέντας, κωδικός πατέντας, ποσοστό σχετικότητας αποτελέσματος με το query>, <6,0,0,0>,<1,0,0,0>
<www.usnews.com>,< education >, <τίτλος>,<1,0,0,0>,<1,0,0,0>
<www.law.emory.edu>,<war, freedom, president, state>, <τίτλος, σκορ σχετικότητας αποτελέσματος με το query, μέγεθος αρχείου, είδος αρχείου>, <8,0,0,0>,<1,0,0,0>
<www.cinemachine.com>, < attack, blood, new>, < τίτλος ταινίας>,<3,0,0,3>,<2,0,0,1>
<www.webcrawler.com>, <apple,coriander, paprika>, < τίτλος, ποσοστό σχετικότητας αποτελέσματος με το query>,<6,0,0,6>,<2,0,0,1,>
<people.yahoo.com>, <Etzioni, Kushmerick, Perkowitz, Franklin>

<ονοματεπώνυμο,internet domain>, <4,0,0,4>,<2,0,0,1>

Συγκεντρωτικά λοιπόν πραγματοποιήθηκαν δοκιμές σε 16 διαδικτυακούς τόπους. Σε πέντε από αυτούς πραγματοποιήθηκε τουλάχιστον μία αλλαγή. Τα συνολικά αποτελέσματα που λάβαμε ως προς την αναγνώριση της ορθής ή μη λειτουργίας του wrapper για ένα site φαίνονται στον παρακάτω πίνακα:

	b	$\neg b$
a	21	0
$\neg a$	0	7

Έτσι οι τιμές που προέκυψαν για τις μετρικές που προαναφέρθηκαν είναι οι παρακάτω: $ac = 100\%$, $up = 100\%$, $cp = 100\%$, $ur = 100\%$, $cr = 100\%$, $F_{CHANGED} = 1$ και $F_{UNCHANGED} = 1$. Οι τιμές αυτές είναι οι ιδανικές. Το σύστημά μας στα δεδομένα είχε τέλεια επίδοση! Τα συνολικά αποτελέσματα που λάβαμε ως προς την αναγνώριση της ορθής ή μη λειτουργίας του wrapper για ένα attribute φαίνονται στον παρακάτω πίνακα:

	b	$\neg b$
a	129	0
$\neg a$	1	37

Οι τιμές που παίρνουμε για τις μετρικές είναι οι εξής:

$ac = 99.4\%$, $up = 100\%$, $cp = 97.37\%$, $ur = 99.23\%$, $cr = 100\%$, $F_{CHANGED} = 0.9867$ και $F_{UNCHANGED} = 0.9961$. Και πάλι η επίδοση του συστήματός μας είναι σχεδόν τέλεια.

5.1.3 Επιλογές Παραμέτρων

Το σύνολο των χαρακτηριστικών γνωρισμάτων των οποίων κάνει χρήση το σύστημα ARMAGEDDON είναι το ελάχιστο δυνατό για την επίτευξη των παραπάνω αποτελεσμάτων. Συγκεκριμένα, αν και αρχικά εντός των διαφόρων χαρακτηριστικών γνωρισμάτων της εξαγόμενης πληροφορία υπήρχε και η πυκνότητα των γραμμάτων διαπιστώθηκε από τα πρώτα κιόλας πειράματα ότι το γνώρισμα αυτό δεν ωφελούσε την επίδοση του συστήματος. Ο λόγος που συνέβαινε αυτό, είναι ότι το χαρακτηριστικό αυτό ήταν περιττό αφού δεν είναι ανεξάρτητο από τα υπόλοιπα. Συγκεκριμένα παράγεται σαν άθροισμα των πυκνοτήτων κεφαλαίων και πεζών γραμμάτων. Επίσης δοκιμάσαμε να προσθέτουμε όρους στην μεταβλητή του Pearson όπως περιγράψαμε στην παράγραφο 3.2.1.3 για τα κοινά patterns των training και testing attribute.Περιμέναμε ότι η ιδέα αυτή δεν θα απέδιδε ιδιαίτερα καλά αφού

θεωρούμε ότι δεν υπάρχει κάποιος λόγος να διατηρείται η αναλογία μεταξύ των ποσοστών των εγγραφών στο training και στο testing attribute που ακολουθούν κάποιο pattern. Όντως τα αποτελέσματα ενθάρρυναν την άποψή μας. Συγκεκριμένα προέκυψαν αρκετά λάθη τα οποία ήταν τύπου $\neg a \wedge b$, δηλαδή ενώ ο wrapper λειτουργούσε σωστά το σύστημα αντιλαμβανόταν το αντίθετο. Τέλος, μία σημαντική μέθοδος από την οποία εξαρτάται η επίδοση του συστήματος είναι η FIRSTTOKENS η οποία επιστρέφει το πλήθος των tokens που θεωρούνται αρχή ενός attribute. Η μέθοδος φαίνεται στον ακόλουθο πίνακα. Παίρνει σαν είσοδο την μέση τιμή των tokens ανά εγγραφή και ανάλογα με την τιμή αυτή επιστρέφει το ζητούμενο πλήθος.

```
FIRSTTOKENS(μ)
 if(μ ≠ 0)
   if(μ ≤ 1)
     return 1
   else if(μ ≤ 2)
     return 2
   else if(μ ≤ 4)
        return 3
    else
      return (μ/3)+1
else return 0
```

5.2 Αξιολόγηση Wrapper Reinduction Module

Όπως είδαμε κατά την ανάλυση του αλγόριθμου για το reinduction μέρος, η ιδέα είναι πολύ απλή. Κατά συνέπεια δεν περιμέναμε κάποια εξαιρετική απόδοση στο μέρος αυτό εκτός από κάποιες περιπτώσεις που το περιεχόμενο των σελίδων είναι στατικό. Στα sites στα οποία διαπιστώθηκε πρόβλημα στην λειτουργία του wrapper εκτελέστηκε ο αλγόριθμος αναζήτησης παραδειγμάτων ορθής πληροφορίας για κάθε attribute. Τα αποτελέσματα φαίνονται στον παρακάτω πίνακα:

url	Query	attributes	Attributes για τα οποία βρέθηκαν σωστά παραδείγματα
people.yahoo.com	Etzioni	ονοματεπώνυμο, internet domain	ονοματεπώνυμο
www.webcrawler.com	apple	τίτλος, ποσοστό σχετικότητας αποτελέσματος	*κανένα*

		με το query	
www.cinemachine.com	attack	τίτλος ταινίας	τίτλος ταινίας
www.uk.lycos.de	apple	τίτλος, υπερσύνδεσμος, ποσοστό σχετικότητας αποτελέσματος με το query	τίτλος, υπερσύνδεσμος
www.altavista.com	apple	τίτλος, υπερσύνδεσμος	τίτλος, υπερσύνδεσμος

ΑΝΑΛΥΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Στο people.yahoo.com δεν βρέθηκε κάποιο σωστό παράδειγμα για το internet domain αφού στις αλλαγμένες ιστοσελίδες δεν υπήρχε καν αυτό το attribute. Το σύστημα βρήκε λανθασμένα παραδείγματα. Ο λόγος γι αυτό ήταν ότι τα παραδείγματα για το internet domain σαν συμβολοσειρές εμπεριέχονταν στις συμβολοσειρές των e-mails που υπήρχαν στις αλλαγμένες ιστοσελίδες. Έτσι το σύστημα καθώς διάβαζε τη συμβολοσειρά detzioni@yahoo.com στην αλλαγμένη ιστοσελίδα έβρισκε το yahoo.com σαν παράδειγμα για το attribute internet domain. Αντίθετα για το ονοματεπώνυμο βρέθηκαν αρκετά παραδείγματα ώστε να παραχθεί ένας νέος wrapper. Αυτό ήταν αναμενόμενο δεδομένης της στατικότητας της πληροφορίας που διέπει το attribute αυτό. Στο www.webcrawler.com δεν βρέθηκαν για κανένα attribute παραδείγματα. Για τον τίτλο, επειδή οι νέες σελίδες δεν περιείχαν κανένα παράδειγμα από αυτά που είχαν εξαχθεί πριν την αλλαγή του site και για το ποσοστό σχετικότητας αποτελέσματος επειδή δεν εμφανιζόταν καν στις νέες ιστοσελίδες. Για το site www.cinemachine.com βρέθηκαν παραδείγματα για τον τίτλο της ταινίας. Για το www.uk.lycos.de δεν βρέθηκαν παραδείγματα μόνο για το ποσοστό σχετικότητας αποτελέσματος και αυτό γιατί δεν υπήρχε στις αλλαγμένες ιστοσελίδες. Τέλος για το site της www.altavista.com βρέθηκαν παραδείγματα και για τα δύο attributes.

Αν και τα αποτελέσματα είναι αρκετά ικανοποιητικά δεδομένης της απλότητας του αλγορίθμου για το reinduction μέρος, το σύστημα που υλοποιήσαμε μπορεί να χρησιμοποιηθεί αποδοτικά και σαν ένα βοηθητικό εργαλείο για τον χρήστη ως εξής: Όταν ειδοποιείται ο χρήστης από το verification module ότι ο wrapper δεν εξάγει σωστά πληροφορία, εντοπίζει ο ίδιος κάποια παραδείγματα για την πληροφορία που θέλει στις νέες ιστοσελίδες, τα σημειώνει σε ένα αρχείο που βρίσκεται σε έναν φάκελο με το όνομα του attribute που αντιπροσωπεύουν και έπειτα προκαλεί την εκτέλεση του reinduction module. Έτσι παράγονται τα annotation αρχεία με τα οποία τροφοδοτείται ο STALKER. Αν αναλογιστούμε ότι ο STALKER απαιτεί λίγα παραδείγματα για την πλειονότητα των

ιστοσελίδων για να παραγάγει έναν σωστό wrapper, αντιλαμβανόμαστε ότι με το reinduction σύστημα μπορεί ο χρήστης πολύ γρήγορα να δημιουργήσει annotation αρχεία.

6

Επίλογος

6.1 Σύνοψη και Συμπεράσματα

Όπως είδαμε στα εισαγωγικά κεφάλαια η πλειονότητα των ιστοσελίδων είναι γραμμένες σε HTML γλώσσα ενώ ταυτόχρονα οι προσπάθειες για την ανάπτυξη του σημασιολογικού ιστού (semantic web) βρίσκονται σε αρχικό στάδιο. Όλα αυτά σε συνδυασμό με την αναμενόμενη απροθυμία[1] αρκετών διαδικτυακών τόπων να αναπτύξουν τις ιστοσελίδες τους με τρόπο «προγραμματοκεντρικό» και όχι μόνο ανθρωποκεντρικό καθιστούν βάσιμη την πρόβλεψη ότι η ανάπτυξη των web wrappers για ημιδομημένες πηγές πληροφορίας δεν θα καταργηθεί άμεσα. Συνεπώς το wrapper maintenance πρόβλημα είναι ένα επίκαιρο και αρκετά σύνθετο πρόβλημα. Κύριο αντικείμενο της εργασίας αυτής ήταν η ανάπτυξη ενός ολοκληρωμένου συστήματος wrapper maintenance. Στα πρώτα τρία κεφάλαια έγινε παρουσίαση κάποιων θεωρητικών γνώσεων καθώς και ορισμένων από τις πιο γνωστές εργασίες πάνω στα θέματα wrapper induction, wrapper verification και wrapper reinduction. Στο τέταρτο κεφάλαιο περιγράφηκε το σύστημα ARMAGEDDON που αναπτύχθηκε στα πλαίσια της εργασίας αυτής και στο πέμπτο κεφάλαιο τα πειραματικά αποτελέσματα .

Η κύρια συνεισφορά της εργασίας έγκειται στην ανάπτυξη ενός σύνθετου verification συστήματος. Οι μέχρι τώρα προσεγγίσεις στο πρόβλημα του wrapper verification είναι δύο κατηγοριών. Η πρώτη προσέγγιση βασίζεται στη δομή των ιστοσελίδων από τις οποίες εξάγεται η πληροφορία (DOM trees) και ονομάζεται structure based ενώ η δεύτερη ονομάζεται content based και βασίζεται στην εξαγόμενη πληροφορία. Οι υπάρχουσες μέθοδοι της δεύτερης κατηγορίας βασίζουν σε αρκετά μεγάλο έως και αποκλειστικό βαθμό όπως

[1]π.χ πολλά sites βασίζονται στις διαφημίσεις οπότε δεν θέλουν να είναι «προγραμματοκεντρικά»

στην περίπτωση του RAPTURE -της πρώτης σημαντικής content based μεθόδου- την επίδοσή τους στο χαρακτηριστικό γνώρισμα της πυκνότητας των HTML χαρακτήρων. Αυτό είναι αναμενόμενο γιατί η εξαγόμενη πληροφορία όταν είναι η επιθυμητή δεν περιέχει καθόλου HTML χαρακτήρες αλλά και το αντίθετο: συνήθως όταν χαλάει ο wrapper η πληροφορία που εξάγεται περιέχει HTML χαρακτήρες. Συνεπώς οι παραπάνω μέθοδοι δεν λαμβάνουν υπ'όψιν τους ότι ακόμα και ένας σωστός wrapper μπορεί να εξάγει εκτός από τα επιθυμητά δεδομένα κάποια άλλα δεδομένα που περιέχουν HTML χαρακτήρες. Αυτό το διαπιστώσαμε εξάλλου στην πράξη χρησιμοποιώντας το wrapper induction σύστημα που διαθέταμε. Αυτό μπορεί να συμβεί ακόμα και στα καλύτερα wrapper induction συστήματα. Το σύστημα μας είναι content based αλλά δεν χρησιμοποιεί καθόλου τη πυκνότητα των HTML χαρακτήρων. Έτσι ο αλγόριθμός μας είναι εύρωστος αφού δεν προϋποθέτει σε καμία περίπτωση καθαρισμό των εξαγόμενων δεδομένων (data cleansing) λαμβάνοντας υπ'όψιν μικρές «ατέλειες» του wrapper αλλά ταυτόχρονα είναι και γενικός δίνοντας την δυνατότητα να χρησιμοποιηθεί και προσαρμοσμένα και σε άλλες εφαρμογές.

Στη συνέχεια αναπτύχθηκε και το reinduction σύστημα το οποίο τίθεται σε λειτουργία από το χρήστη όταν εντοπιστεί πρόβλημα στην λειτουργία του wrapper. Η ιδέα του συστήματος αυτού είναι ο προσδιορισμός παραδειγμάτων επιθυμητής πληροφορίας στις αλλαγμένες ιστοσελίδες με βάση την πληροφορία που γνωρίζουμε ότι εξάχθηκε κατά τη διάρκεια της ορθής λειτουργίας του wrapper. Σε αντίθεση με το verification σύστημα, το reinduction σύστημα αναπτύχθηκε με βάση τις δυνατότητες που διέθετε το wrapper induction σύστημα που είχαμε στη διάθεση μας.

Με βάση τα πειραματικά αποτελέσματα συμπεράναμε ότι ο verification αλγόριθμος αποδίδει άριστα στην πράξη αλλά και το reinduction σύστημα έχει ικανοποιητική απόδοση. Έτσι, καταλήγουμε στο ότι το σύστημα ARMAGEDDON είναι ένα αξιόπιστο wrapper maintenance σύστημα.

6.2 Μελλοντικές Επεκτάσεις

Όσον αφορά το verification μέρος της εργασίας τα αποτελέσματα που λάβαμε κατά την αξιολόγηση του verification αλγόριθμου είναι ενθαρρυντικά για τη προσέγγιση μας στο πρόβλημα . Για την καλύτερη αξιολόγηση του αλγορίθμου θα γίνουν στο άμεσο μέλλον εκτενέστερα πειράματα. Επιπλέον, θα προσπαθήσουμε να βελτιώσουμε περαιτέρω το σύστημα ποινών είτε κάνοντας το πολυπλοκότερο είτε απλούστερο προσθέτοντας ή συγχωνεύοντας κάποιες περιπτώσεις αντίστοιχα αλλά και προσδιορίζοντας με εμπειρικό τρόπο τις βέλτιστες τιμές για τις διάφορες ποινές που επιβάλλονται. Ένα άλλο ζήτημα έρευνας είναι η εύρεση και άλλων χρήσιμων γνωρισμάτων για τον έλεγχο Pearson τα οποία θα βελτιώνουν την απόδοση του αλγόριθμου. Τέλος, άλλη μία ερευνητική προέκταση είναι η

εύρεση άλλων προβλημάτων στα οποία θα μπορούσε να εφαρμοστεί η μέθοδος που αναπτύχθηκε και η προσαρμοσμένη εφαρμογή της σε αυτά.

Σχετικά με το reinduction μέρος ο προκλητικός στόχος είναι η εξεύρεση παραδειγμάτων της επιθυμητής πληροφορίας ακόμα και όταν δεν υπάρχουν παραδείγματα τα οποία γνωρίζουμε ότι είναι τα επιθυμητά. Η ιδέα αυτή υπάρχει στην εργασία [LMK03] και εκμεταλλεύεται την ύπαρξη των patterns. Μελλοντικός στόχος είναι η υλοποίηση ενός συστήματος το οποίο θα εκμεταλλεύεται τα patterns καθώς και το DOM δένδρο των ιστοσελίδων με βάση τις διάφορες ευριστικές που αναπτύχθηκαν στην παράγραφο 3.3.2.

7

Βιβλιογραφία

[IGS06]	Internet World Statistics, available at: http://www.internetworldstats.com/emarketing.htm
[Ber01]	Bergman, M., *The 'Deep' Web: Surfacing Hidden Value*, available at: http://www.brightplanet.com/images/stories/pdf/deepwebwhitepaper.pdf
[SA99]	Sahuguet,A.,Azavant,F. *Web Ecology :Recycling HTML pages as XML using W4F*, available at: http://db.cis.upenn.edu/DL/webdb99.pdf
[WI06]	Wikipedia, "Wikipedia ,the free encyclopaedia",available at: en.wikipedia.org
[AI06]	Alberta Ingenuity Centre for Machine Learning ,available at: http://www.aicml.cs.ualberta.ca/research/machineLearning/index.php
[Mit97]	Mitchell,T., *Machine Learning*,McGraw-Hill,1997
[AG02]	Arasu, A., Garcia-Molina, H., *Extracting Structured Data from Web Pages*, available at: http://dbpubs.stanford.edu:8090/pub/2002-40
[MMK01]	Muslea,I.,Minton,S.,Knoblock,C., *Hierarchical Wrapper Induction for Semistructured Information Sources* , available at: http://citeseer.ist.psu.edu/muslea01hierarchical.html
[LMK03]	Lerman,K., Minton,S., Knoblock,C. **:**Wrapper Maintenance: A Machine Learning Approach , available at:http://citeseer.ist.psu.edu/581032.html
[Kus00]	Kushmerick,N., Wrapper Verification, available at: http://citeseer.ist.psu.edu/383908.html
[PLL03]	Pek,E.,Li,X.,Liu,Y., *Web Wrapper Validation*, available at: http://www.springerlink.com/content/dr5r02w2raqmw1wd/fulltext.pdf
[RPV05]	Raposo, J., Pan, A., Viña, Á., Álvarez,M., *Automatic Wrapper Maintenance for Semi-Structured Web Sources Using Results from previous queries* ,available at: http://portal.acm.org/citation.cfm?id=1066677.1066826