ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Εφαρμογή Τεχνικών Ταιριάσματος Οντολογιών για Αυτόματες Οντότητες Ηλεκτρονικών Αγορών (Shopbots)

Καρασμάνογλου Ι. Γεώργιος

Επιβλέποντες:

Ευστάθιος Χατζηευθυμιάδης, Επίκουρος Καθηγητής ΕΚΠΑ

Κωνσταντίνος Κολομβάτσος, Υποψήφιος Διδάκτωρ ΕΚΠΑ

ΑΘΗΝΑ

ΣΕΠΤΕΜΒΡΙΟΣ 2009

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Εφαρμογή Τεχνικών Ταιριάσματος Οντολογιών για Αυτόματες Οντότητες Ηλεκτρονικών Αγορών (Shopbots)

Καρασμάνογλου Ι. Γεώργιος

Α.Μ.:1115200200199

ΕΠΙΒΛΕΠΟΝΤΕΣ:

Ευστάθιος Χατζηευθυμιάδης, Επίκουρος Καθηγητής ΕΚΠΑ

Κωνσταντίνος Κολομβάτσος, Υποψήφιος Διδάκτωρ ΕΚΠΑ

Ημερομηνία εξέτασης 15/09/2005

ΠΕΡΙΛΗΨΗ

Η ανάπτυξη του Σημασιολογικού Ιστού (ΣΙ - Semantic Web), αναμένεται να φέρει επανάσταση στην ζωή των ανθρώπων, με πολλαπλές εφαρμογές στην καθημερινότητα τους. Οι πληροφορίες στο σημασιολογικό ιστό, αποθηκεύονται με την βοήθεια οντολογιών, οι οποίες αποτελούν μια δομή σημασιολογικής αναπαράστασης, κατάλληλη, ώστε να είναι επεξεργάσιμη από τις μηχανές.

Οι ηλεκτρονικές αγορές αποτελούν εικονικούς τόπους αγορών και πωλήσεων αγαθών. Στις αγορές αυτές συμμετέχουν διάφορες οντότητες όπως οι αγοραστές (αυτοί που θέλουν να αποκτήσουν ένα αγαθό), οι πωλητές (που θέλουν να πουλήσουν ένα αγαθό) καθώς και κάποιες ενδιάμεσες βοηθητικές οντότητες. Μια από αυτές, η οποία αποτελεί τη βάση της παρούσας εργασίας, είναι οι οντότητες αγορών (shopbots). Οι οντότητες αυτές, είναι υπεύθυνες για την αναζήτηση, την εύρεση και την παρουσίαση των προϊόντων που ταιριάζουν με τις προτιμήσεις των αγοραστών. Παρά τα πλεονεκτήματα και τις διευκολύνσεις που προσφέρει η ύπαρξή τους στους χρήστες, υπάρχουν κάποια προβλήματα τα οποία μειώνουν την αποτελεσματικότητα τους. Ένα από αυτά τα προβλήματα αποτελεί η ετερογένεια που μπορεί να υπάρξει σχετικά με την αναπαράσταση της πληροφορίας που σχετίζεται με τα χαρακτηριστικά των προιόντων. Τα προβλήματα αυτά, μπορούν να λυθούν με την χρήση των οντολογιών στο περιβάλλον της ηλεκτρονικής αγοράς. Έτσι, ονότητες όπως τα shopbots θα μπορούν να επεξεργαστούν καλύτερα και σε λιγότερο χρόνο τα χαρακτηριστικά των προϊόντων και να καταλήξουν σε ικανοποιητικότερα αποτελέσματα.

Παρόλα αυτά, δεν υπάρχουν κανόνες ως προς τη δημιουργία των οντολογιών, και έτσι, η δομή τους αλλάζει σύμφωνα με τις συμβάσεις που έχει στο μυαλό του ο εκάστοτε δημιουργός. Έτσι, προκύπτει το πρόβλημα της επικοινωνίας μεταξύ τους, ώστε να μπορούν τα διάφορα shopbots να αντιληφθούν τα χαρακτηριστικά των προϊόντων που περιγράφονται σε αυτές. Σκοπός της παρούσας εργασίας είναι η μελέτη των υπαρχουσών μεθόδων και αλγορίθμων ώστε να αντιμετωπιστεί το πρόβλημα αυτό, καθώς και η πρόταση ενός νέου αλγορίθμου που να υπερκαλύπτει τα μειονεκτήματα των ήδη υπαρχόντων μεθοδολογιών.

Πιο συγκεκριμένα, μελετάμε την ιστορία και τα βασικά χαρακτηριστικά του ΣΙ, καθώς και των οντολογιών στις οποίες αποθηκεύονται οι πληροφορίες του. Στη συνέχεια

εξετάζουμε το σενάριο των ηλεκτρονικών αγορών και τον τρόπο λειτουργία τους, που μπορεί να επεκταθεί σε πολλές εφαρμογές του ΣΙ. Σε αυτό το σενάριο θα βασίζουμε τη συνέχεια της εργασίας αυτής. Έπειτα, ερευνούμε διάφορες μεθόδους για την αντιστοίχιση πανομοιότυπων στοιχείων σε δύο διαφορετικές οντολογίες (ταίριασμα). Στη συνέχεια, επεκτείνουμε τη μελέτη σε αλγορίθμους που βρίσκουν αντιστοιχίες συνολικά σε δύο οντολογίες, και όχι μεμονωμένα σε δύο στοιχεία όπως στις ήδη προτεινόμενες μεθόδους. Μέσα από την μελέτη των πλεονεκτημάτων και των μειονεκτημάτων τους, καταλήξαμε σε μια δική μας υλοποίηση την οποία και παρουσιάζουμε. Στο κεφάλαιο 6 παρουσιάζεται η λειτουργία του προτεινόμενου αλγορίθμου, ενώ στο κεφάλαιο 7 συγκρίνεται πειραματικά με κάποιους υπάρχοντες αλγορίθμους. Τα πειράματα που διενεργήσαμε, υποδηλώνουν μια ικανοποιητική επίδοση όσον αφορά τη βελτίωση της λειτουργίας των shopbots.

Το δεύτερο σκέλος της παρούσας εργασίας αποτελεί η μεταφορά πληροφοριών από μια οντολογία σε μια άλλη (σημασιολογική μετάφραση). Για τη βέλτιστη λειτουργία των shopbots, θα πρέπει οι πληροφορίες, οι οποίες είναι αποθηκευμένες σε μια οντολογία, να μπορούν να αντιγραφούν σε μια άλλη οντολογία, ώστε να μπορούν έτσι οι αυτόματες οντότητες να τις επεξεργαστούν και να παρουσιάσουν τα αποτελέσματα στο χρήστη. Παρόλα αυτά, η μελέτη έχει επικεντρωθεί κυρίως στη μελέτη της διαδικασίας ταιριάσματος. Ο δεύτερος σκοπός της εργασίας είναι να καλυφθεί αυτό το κενό. Προτείνεται λοιπόν, μια διαδικασία σημασιολογικής μετάφρασης των οντολογιών, η οποία παρουσιάζεται και αυτή στο κεφάλαιο 6.

Τελικές παρατηρήσεις, συμπεράσματα και κατευθύνσεις για μελλοντικές επεκτάσεις δίδονται στο 8ο και τελευταίο κεφάλαιο της παρούσας μελέτης.

Λέξεις Κλειδιά: ταίριασμα οντολογιών, οντολογίες, ηλεκτρονικό εμπόριο, χαρτογράφηση οντολογιών, αυτόματες οντότητες αγορών, μεταφορά πληροφοριών σε οντολογίες, σημασιολογική μετάφραση

ABSTRACT

The development of the Semantic Web (SW - Semantic Web), is expected to revolutionize the people's lives with multiple applications in their daily activities. The information in the SW, is stored by using ontological terms. Ontologies provide a common conceptualization of a domain and define terms for a storage structure suitable to be editable from the machines.

Electronic markets are virtual places where entities not known in advance can negotiate and agree upon the exchange of products. These markets involve various entities such as buyers (who want to acquire goods), sellers (who want to sell a good) and some intermediate entities. One type of the intermediaries are shopbots (shopping robots). Shopbots consist of the basis of this thesis. They are responsible to seek, find and present a set of products which are conformed by the preferences of buyers. Despite of the advantages and facilities that they offer to users, there are some problems that can limit their effectiveness. These problems can be solved by the use of ontologies and especially in an electronic market environment. Thus, shopbots can act more efficiently providing the most appropriate results.

However, there is no common methodology for ontology development. For this reason, their structure depends on their creator view on the specific domain. Thus, shopbots acting in a heterogeneous environment should be capable of dealing with this heterogeneity in order to be able to provide the most efficient results. The purpose of this work is to study existing methods and algorithms to address this problem, and propose a new algorithm.

In particular, we study the history and main features of the SW and the ontologies that are its core technology. We examine the scenario of buying and their function, which can be extended to many applications of the SW. Next, we investigate various methods to match items taken by two different ontologies (ontology matching). Then we expand this study to algorithms that find matches to a total of two ontologies, rather than two individual elements. Through this study, we present the advantages and disadvantages of each methodology. Through this, we present our proposal for this specific shopbots scenario. Chapter 6 presents our algorithm, while in Chapter 7 we provide a comparison of our approach with some other existing algorithms. The experiments conducted, indicate a satisfactory performance in improving the functionality of shopbots.

The second part of this work is to study the transfer of information from one ontology to another (semantic translation). In order to gain the optimum operation of shopbots, there must be a way of copying the data which is stored in an ontology to another ontology, so that the autonomous entities could be able to process them and present their results to users. However, the study has focused mainly on studying the matching process. The second aim of this thesis is to fill this gap. So, a semantic translation of data is introduced in chapter 6.
Final remarks, conclusions and directions for future work are given in the 8th chapter of this study.

Keywords: Ontology matching, ontologies, electronic market, ontology mapping, data copying in ontologies, shopbots, semantic translation

Ευχαριστίες

Θα ήθελα να ευχαριστήσω αρχικά, τον επιβλέποντα καθηγητή μου, κ. Ευστάθιο Χατζηευθυμιάδη, Επίκουρο Καθηγητή του τμήματος Πληροφορικής και Τηλεπικοινωνιών του ΕΚΠΑ, ο οποίος μου έδωσε την ευκαιρία να ασχοληθώ με το συγκεκριμένο θέμα. Ακόμα, θα ήθελα να τον ευχαριστήσω, για τη συμβολή του και τη βοήθεια του κατά το διάστημα που προσπαθούσα να φέρω εις πέρας την εργασία.

Επίσης, θα ήθελα να ευχαριστήσω το μεταπτυχιακό του τμήματος Πληροφορικής και Τηλεπικοινωνιών, κ. Κολομβάτσο Κώστα για τις πολύτιμες συμβουλές και τη βοήθεια του. Οι παρατηρήσεις του, αποδείχθηκαν πολύτιμες για την εκπόνηση της εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου και τα κοντινά μου πρόσωπα για την υποστήριξη τους όλο αυτό το διάστημα.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

1.1 Κίνητρο

Τα τελευταία χρόνια, η ανάπτυξη του Παγκόσμιου Ιστού (World Wide Web – WWW) έχει φέρει επανάσταση στην κοινωνία που ζούμε. Μια πληθώρα από εφαρμογές, πληροφορίες, υπηρεσίες έχουν αναπτυχθεί που στοχεύουν στην βελτίωση της ποιότητας ζωής των ανθρώπων. Η μαζική πρόσβαση και η μεγάλη προσφορά σε υπηρεσίες οδήγησε στην ανάδειξη του WWW ως ένα μέσο παγκόσμιο, ισχυρότερο από τα άλλα συμβατικά μέσα επικοινωνίας όπως την τηλεόραση ή το ραδιόφωνο. Η ανάπτυξη αυτή, οδήγησε και στην αύξηση των πληροφοριών που βρίσκονται στον Ιστό. Ενδεικτικά αναφέρουμε πως βασιζόμενοι στα δεδομένα του Netcraft, ο αριθμός των σελίδων στον Παγκόσμιο Ιστό ξεπερνά τις 29.7 δισεκατομμύρια σελίδες. Με την αύξηση των πληροφοριών επήλθε και αύξηση της ζήτησης από τους χρήστες. Παρόλα αυτά, η μεγάλη πληθώρα πληροφοριών καθιστά την αναζήτηση τους μια πολύ δύσκολη και χρονοβόρα διαδικασία. Αυτό το πρόβλημα σκοπεύει να καλύψει η ανάπτυξη του σημασιολογικού ιστού (Semantic Web - ΣΙ) [3].

Μέσα από τον Παγκόσμιο Ιστό η αναζήτηση δεδομένων γίνεται συνήθως με βάση κάποιες συμβολοσειρές χωρίς να γίνεται επεξεργασία δεδομένων, η οποία επαφίεται στο χρήστη. Αυτό το κενό στοχεύει να καλύψει ο ΣΙ. Ως ΣΙ ορίζουμε την οργάνωση των πληροφοριών(που στο Παγκόσμιο Ιστό βρίσκονται ελεύθερες) σε ένα πλέγμα, συνδεδεμένες με τέτοιο τρόπο ώστε να είναι επεξεργάσιμες από τις μηχανές. Ο ΣΙ αποτελεί έμπνευση του Tim Berners-Lee που ουσιαστικά είναι ο εφευρέτης και του σημερινού Παγκόσμιου Ιστού. Ο ορισμός που έδωσε γι’ αυτόν είναι:

“The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation”

[Tim Berners-Lee, James Hedler, Ora Lassila, “The Semantic Web”, Scientific American, May 2001].

Σύμφωνα με τον παραπάνω ορισμό, ο ΣΙ δεν αποτελεί κάτι νέο αλλά μια επέκταση του Παγκόσμιου Ιστού. Αποτελεί ένα εργαλείο, κατά το οποίο οι μηχανές επεξεργάζονται τα δεδομένα και τις πληροφορίες αναλόγως με τις προτιμήσεις του χρήστη, και του παρουσιάζουν τα αποτελέσματα. Τα μέσα στο οποίο αποθηκεύονται οι πληροφορίες ώστε να είναι επεξεργάσιμες από τις οντότητες λογισμικού (bots) που περιέχει ο ΣΙ, ονομάζονται οντολογίες. Οι οντολογίες [2] είναι ένας τρόπος οργάνωσης των δεδομένων, κατά τρόπο τέτοιο ώστε να γίνεται εύκολα αντιληπτή η σχέση μεταξύ τους από τα bots.

Μια άλλη έννοια που πρέπει να εξετάσουμε αποτελούν οι ηλεκτρονικές αγορές. Με τον όρο ηλεκτρονική αγορά εννοούμε ένα μέρος όπου γίνονται ηλεκτρονικές ανταλλαγές διαφόρων αγαθών. Υπάρχουν οι πελάτες, όπου εκφράζουν ζήτηση για κάποια αγαθά, οι πωλητές, οι οποίοι έχουν τα αγαθά και τα προσφέρουν με κάποιο αντίτιμο, καθώς και οι ενδιάμεσες οντότητες που προσφέρουν βοήθεια και αξιοπιστία στις υπόλοιπες. Οι οντότητες αυτές έχουν αντικρουόμενα συμφέροντα(οι πελάτες ενδιαφέρονται να αγοράσουν στη χαμηλότερη δυνατή τιμή, ενώ οι πωλητές να πουλήσουν στην υψηλότερη). Έτσι, σε αυτές τις αγορές μπορεί να υφίσταται μια διαδικασία διαπραγμάτευσης, κατά την οποία γίνονται διάφορες προτάσεις αντιτίμου και από τις δύο μεριές για να καταλήξουν σε μια τιμή που ικανοποιεί και τις δύο μεριές ή να υφίσταται μια διαδικασία δημοπρασίας σχετικά με την αγορά ενός προιόντος. Μια αγοραπωλησία θεωρείται πετυχημένη, αν καταλήξει σε μια τιμή που συμφέρει και τις δύο πλευρές και επιτευχθεί συμφωνία, ενώ θεωρείται αποτυχημένη στην αντίθετη περίπτωση.

Όλες οι οντότητες που περιγράψαμε μπορούν να αντικατασταθούν από αυτόνομο λογισμικό το οποίο είναι υπεύθυνο για την τήρηση των παραπάνω στόχων, ώστε να διευκολυνθεί η λειτουργία από την πλευρά των χρηστών. Με τη χρήση λογισμικού, αυξάνεται η επιτυχία και αυτονομία των συναλλαγών αλλά και μειώνεται σημαντικά ο χρόνος που σπαταλά ο χρήστης για τη διαδικασία αυτή

Μια κατηγορία τέτοιων οντοτήτων λογισμικού αποτελούν οι οντότητες αγοράς (shopping bots ή shopbots) [76] Τα shopbots, όπως δηλώνει και το όνομα τους, αποτελούν αυτόματες οντότητες λογισμικού, οι οποίες κάνουν αναζήτηση σε διάφορους πωλητές, για την εύρεση προϊόντων που ανταποκρίνονται στις απαιτήσεις του καταναλωτή. Μπορούν μέσα από την αναζήτηση αυτή, να του παρουσιάσουν σχεδόν όλες τις προσφορές που σχετίζονται με το προϊόν που αναζητά.

Για να εκπληρωθούν πετυχημένα οι διαδικασίες αναζήτησης αγαθών σε διάφορους πωλητές, η γρήγορη εύρεση των σχετικών προϊόντων, και η προσαρμοσμένη παρουσίαση των αποτελεσμάτων στους αγοραστές, θα πρέπει τα shopbots να είναι αρκετά ευέλικτα στις λειτουργίες τους. Με τον όρο ευέλικτα εννοούμε να μπορούν να αντιληφθούν τις προτιμήσεις των χρηστών και να μπορούν να τις συγκρίνουν με τα προϊόντα για να αποφανθούν αν πληρούν τις προϋποθέσεις και ενδιαφέρουν το αγοραστή ή όχι. Να μπορούν με άλλα λόγια να επεξεργαστούν τα δεδομένα. Αυτό πετυχαίνεται με τον συνδυασμό του περιβάλλοντος της ηλεκτρονικής αγοράς με την τεχνολογία του ΣΙ. Μέσα από την αναπαράσταση των δεδομένων με οντολογίες, έχουμε πολλά πλεονεκτήματα όπως:

- Δυνατότητα επεξεργασίας δεδομένων.
- Δυνατότητα ομαδοποίησης τους.
- Τα δεδομένα θα μπορούν να παρουσιαστούν με διάφορους τρόπους, κατατάξεις, μορφές αναλόγως με τις προτιμήσεις των χρηστών, όπως και στις βάσεις δεδομένων.
- Άμεση και εύκολη ανανέωση των δεδομένων.
- Δυνατότητα σύγκρισης με άλλα παρόμοια προϊόντα.

Παρόλα αυτά, υπάρχουν ορισμένες προκλήσεις που χρειάζεται να επιλυθούν για την βέλτιστη λειτουργία του παραπάνω σεναρίου. Θα πρέπει να υπάρξει ένας τρόπος επικοινωνίας μεταξύ των οντοτήτων. Ένας τρόπος, κατά τον οποίο θα αποστέλλονται οι πληροφορίες σχετικά με τα αγαθά από την μια οντότητα στην άλλη, και θα μπορούν να είναι επεξεργάσιμες. Οι οντολογίες, παρά τις διευκολύνσεις που προσφέρουν όσον αφορά την οργάνωση, την επεξεργασιμότητα τους από τις μηχανές και την εύκολη επανάκληση των πληροφοριών έχουν και κάποια προβλήματα: Δεν υπάρχει κάποιο πρότυπο να την δημιουργία τους, και έτσι η δομή και ο τρόπος οργάνωσης των πληροφοριών είναι αποκλειστικά θέμα του δημιουργού τους. Έτσι, θα υπάρχουν πολλές διαφορές από οντολογία σε οντολογία, αναλόγως με τις πεποιθήσεις του δημιουργού, τις συμβάσεις που κάνει κ.λ.π. ακόμα και αν αναφέρονται στις ίδιες πληροφορίες. Αλλά και για την μετάβαση από τον Παγκόσμιο Ιστό, όπου οι πληροφορίες είναι αποθηκευμένες κυρίως σε βάσεις δεδομένων, στο ΣΙ που χρησιμοποιεί οντολογίες, χρειάζεται ένα είδος επικοινωνίας για την μεταφορά και επεξεργασία των δεδομένων. Ο στόχος αυτής της επικοινωνίας είναι η ενοποίηση διαφόρων ετερογενών στοιχείων από διάφορες πηγές(πωλητές) σε ένα σύνολο, ώστε να μην χρειάζεται αναζήτηση από το τελικό χρήστη, αλλά και για ευκολότερη σύγκριση μεταξύ τους.

Συνοψίζοντας λοιπόν, είναι ανάγκη να βρεθεί ένας τρόπος, κατά τον οποίο θα είναι πιθανό, οι πληροφορίες από μια οντότητα (οντολογία ή βάση δεδομένων) να διαβαστούν, να αντιγραφούν και να αποθηκευτούν σε μια άλλη οντότητα (οντολογία), με όσο το δυνατόν καλύτερη οργάνωση και λογική, ώστε να είναι εύκολα επεξεργάσιμες από το λογισμικό.

1.2 Επιδιωκόμενοι Στόχοι

Έχουν υπάρξει πολλές μελέτες κατά καιρούς σχετικά με την επικοινωνία μεταξύ οντολογιών. Οι μελέτες αυτές είναι κυρίως προσανατολισμένες στην εύρεση των πανομοιότυπων στοιχείων μεταξύ τους, ώστε να υπάρξει μια αντιστοίχιση. Στη παρούσα εργασία, οι στόχοι στους οποίους προσβλέπουμε είναι οι εξής:

- Αρχικός στόχος αυτής της εργασίας είναι η μελέτη των ήδη υπαρχόντων αλγορίθμων για εύρεση ομοιοτήτων ανάμεσα σε οντολογίες. Η μελέτη αυτή γίνεται υπό το πρίσμα του σεναρίου μιας ηλεκτρονικής αγοράς στην οποία γίνεται διακίνηση πληροφοριών για προϊόντα. Μελετώνται τα πλεονεκτήματα και μειονεκτήματα αυτών για την εξαγωγή ορισμένων συμπερασμάτων. Για την μελέτη αυτή γίνεται εκτενής αναζήτηση σε βιβλιογραφία, ώστε να φανούν οι όποιες λειτουργίες τους. Στη παρούσα μελέτη παρουσιάζονται οι βασικότεροι αλγόριθμοι. Η επιλογή τους έγινε με βάση την προσέγγιση που χρησιμοποιούν για να υπολογίσουν την ομοιότητα, ώστε να καλυφθούν όσο το δυνατόν περισσότερες τεχνικές.
- Επόμενος στόχος είναι να καταλήξουμε σε ένα αλγόριθμο κατάλληλο για το σενάριο του ηλεκτρονικού εμπορίου. Η επιλογή αυτή θα βασιστεί στα θετικά και αρνητικά των αλγορίθμων που μελετήσαμε. Θα προσπαθήσουμε ο αλγόριθμος αυτός να έχει ικανοποιητική συμπεριφορά, υπό την έννοια ότι θα έχει υψηλό ποσοστό επιτυχίας στο ταίριασμα οντολογιών προϊόντων, χωρίς να περιέχει μειονεκτήματα σε σχέση με τους υπόλοιπους αλγόριθμους και τεχνικές που μελετήσαμε.
- Τέλος, θα μελετηθεί και θα υλοποιηθεί μια λειτουργία κατά την οποία θα έχουμε αντιγραφή των πληροφοριών από μια οντολογία και εισαγωγή τους σε μια άλλη. Η λειτουργία αυτή θα γίνει με βάση τις ομοιότητες που θα βρεθούν από τον αλγόριθμο στον οποίο καταλήξαμε.

Αποσκοπούμε έτσι να πετύχουμε την μεταφορά πληροφοριών από διάφορες οντολογίες, σε μια γενική, με τέτοιο τρόπο, ώστε να πετύχουμε μια καλή οργάνωση

των δεδομένων, αλλά και να μπορούν οι πληροφορίες αυτές να γίνουν εύκολα αντιληπτές από τα διάφορες οντότητες λογισμικού. Πιστεύουμε ότι με την παρούσα μελέτη καλύπτουμε ένα μεγάλο ποσοστό ετερογένειας στις πληροφορίες και πετυχαίνουμε έτσι τον γενικό στόχο της ομογενοποιήσης των πληροφοριών, όσον αφορά το ηλεκτρονικό εμπόριο και τα προϊόντα.

1.3 Περιγραφή

Η παρούσα εργασία έχει οργανωθεί σε 8 κεφάλαια. Το παρόν κεφάλαιο αποτελεί το αρχικό, μια εισαγωγή στις έννοιες που θα συναντήσουμε και θα αναπτύξουμε λεπτομερέστατα στη συνέχεια της εργασίας.

Στο κεφάλαιο 2 μελετάμε τον ΣΙ. Αναφέρουμε τα σημαντικότερα χαρακτηριστικά του καθώς και χαρακτηριστικά των οντολογιών που αποτελούν την βασικότερη τεχνολογία του ΣΙ. Αναπτύσσουμε επίσης τη δομή των οντολογιών, και αναφέρουμε απαραίτητες πληροφορίες για κατανόηση της λειτουργίας της μεθοδολογίας μας.

Στο κεφάλαιο 3, πραγματοποιούμε μια επισκόπηση του ηλεκτρονικού εμπορίου και πιο συγκεκριμένα των ηλεκτρονικών αγορών ενώ ταυτόχρονα αναλύεται το σενάριο πάνω στο οποίο βασίζεται η εργασία αυτή. Μελετάμε τις αρχές λειτουργίας τους, καθώς και τις οντότητες που συμμετέχουν σε αυτές. Ακόμα, παρουσιάζονται τα πλεονεκτήματα τους σε σχέση με το πραγματικό εμπόριο. Τέλος, μελετώνται οι αυτόματες οντότητες που μπορεί να περιέχουν και ειδικότερα οι αυτόματες οντότητες αγορών, που μας απασχολούν στη παρούσα εργασία. Εμφανίζονται οι λειτουργίες και η ανάγκη ύπαρξης τους, καθώς και οι βελτιώσεις στις λειτουργίες τους που μπορούν να γίνουν με την χρήση των οντολογιών. Εδώ, γίνεται πιο εμφανής η ανάγκη για επικοινωνία μεταξύ των οντοτήτων και για ομογενοποιημένα δεδομένα.

Στο κεφάλαιο 4 εμφανίζουμε τις κυριότερες τεχνικές με βάση τις οποίες γίνεται η αντιστοίχιση των οντολογιών. Έχουμε χωρίσει το κεφάλαιο σε δύο τμήματα, για να αναπτύξουμε τις κυριότερες κατηγορίες ελέγχου. Στο πρώτο κομμάτι εξετάζουμε τη λεξικογραφική ομοιότητα, κατά την οποία ελέγχονται τα ονόματα των στοιχείων που αποτελούν τις οντολογίες, και εξετάζεται η ομοιότητα ή μη αναλόγως με τα στοιχεία/σύμβολα που τις αποτελούν. Στο επόμενο κομμάτι εξετάζεται η σημασιολογική ομοιότητα η οποία εξετάζει αν δύο οντότητες που ανήκουν σε οντολογίες έχουν κοινά χαρακτηριστικά π.χ. συνώνυμα.

Στο επόμενο κεφάλαιο, το 5ο, εξετάζονται οι αλγόριθμοι που έχουν προταθεί κατά καιρούς για την αντιστοίχιση των οντολογιών, οι οποίοι βασίζονται στις τεχνικές του

κεφαλαίου 4. Εξετάζεται μια πληθώρα αλγορίθμων, με διαφορετικές προσεγγίσεις στο πρόβλημα. Πιο συγκεκριμένα παρουσιάζουμε τον τρόπο λειτουργίας τους, τις μεθόδους στις οποίες βασίζονται καθώς και τα μειονεκτήματα τους.

Στο 6ο κεφάλαιο, που αποτελεί και τη βάση της πτυχιακής, παρουσιάζουμε τον αλγόριθμο που προτείνουμε. Σε αυτό, εμφανίζουμε εκτενέστατα τον τρόπο λειτουργίας του. Αναλύουμε επακριβώς τα βήματα του και επεξηγούμε τους λόγους που χρησιμοποιήσαμε αυτά τα βήματα. Εξετάζουμε τους τρόπους με τους οποίους πετυχαίνει την αντιστοίχιση των οντολογιών. Η αντιστοίχιση αυτή, γίνεται μεταξύ δύο οντολογιών κάθε φορά: την οντολογία πηγής και την οντολογία στόχου. Ως οντολογία πηγής θεωρούμε την οντολογία που περιέχει τις πληροφορίες για τα διάφορα προϊόντα, την οντολογία των πωλητών. Η οντολογία στόχου είναι η οντολογία του χρήστη, αυτή στην οποία θα εισάγουμε τα δεδομένα για τα προϊόντα τα οποία ενδιαφέρεται και βρίσκονται στην οντολογία πηγής. Θα μπορούσαμε να πούμε πως η οντολογία πηγής είναι η οντολογία από την οποία αντιγράφονται τα δεδομένα, ενώ οντολογία στόχου η οντολογία στην οποία θα προστεθούν τα δεδομένα αυτά. Στη συνέχεια, εμφανίζουμε τον τρόπο που γίνεται η συλλογή των δεδομένων από την οντολογία πηγής αλλά και η εισαγωγή τους στην οντολογία στόχου, χρησιμοποιώντας την αντιστοίχιση που διαμορφώθηκε στο προηγούμενο βήμα. Τέλος, αναλύουμε και κάποιες λειτουργίες που έχουμε αναπτύξει για διευκόλυνση των υπολογισμών και ελαχιστοποίηση του χρόνου εκτέλεσης του αλγορίθμου (δημιουργία πίνακα πιθανοτήτων ομοιότητας μεταξύ των στοιχείων ώστε να γίνεται εύκολη ανάκληση και επεξεργασία των τιμών του, υπολογισμός μέσου όρου τιμών του πίνακα αυτού με βάρη και χωρίς, υπολογισμός πιθανότερου ταιριάσματος καθώς και έλεγχος και επεξεργασία των βαρών).

Στο 7ο κεφάλαιο, γίνεται η αποτίμηση της λειτουργίας του αλγορίθμου με πραγματικά δεδομένα. Εμφανίζουμε τα αποτελέσματα του σε πραγματικά σενάρια, και τα αντιπαραθέτουμε με άλλους αλγορίθμους για σύγκριση.

Στο 8ο και τελευταίο κεφάλαιο δίνουμε τα συμπεράσματα μας όσον αφορά τη λειτουργία του αλγόριθμου και κάνουμε κάποιες προτάσεις για μελλοντική κατεύθυνση.

1.4 Επίλογος

Το παρόν κεφάλαιο αποτελεί την εισαγωγή της εργασίας μας. Περιγράφηκε επιγραμματικά το πρόβλημα, και η ανάγκη για ομοιογένεια και συλλογή των δεδομένων σε ένα σενάριο ηλεκτρονικής αγοράς, αλλά και γενικότερα στο ΣΙ. Το πρόβλημα αυτό ανάγεται σε δύο στάδια: αυτό της εύρεσης των ομοιοτήτων μεταξύ οντολογιών, δηλαδή

τα στοιχεία που είναι παρόμοια μεταξύ δύο οντολογίες, και αυτό της αντιγραφής των δεδομένων από τη μια οντολογία στην άλλη σε κατάλληλες θέσεις με βάση αυτές τις ομοιότητες. Η μέχρι τώρα έρευνα πάνω στο πρόβλημα αυτό, προσανατολίζεται κυρίως στο πρώτο στάδιο.

Οι στόχοι μας αφορούν και τα δύο στάδια. Όσον αφορά το πρώτο, θα μελετήσουμε τους αλγόριθμους που ήδη υπάρχουν, και θα τους παρουσιάσουμε καθώς και μια νέα τεχνική που προσανατολίζεται ειδικά για το ηλεκτρονικό εμπόριο. Για το δεύτερο στάδιο, έχουμε ως στόχο την υλοποίηση μιας λειτουργίας που αποπερατώνει πλήρως τη μεταφορά των πληροφοριών στις σωστές θέσεις.

ΚΕΦΑΛΑΙΟ 2

ΣΗΜΑΣΙΟΛΟΓΙΚΟΣ ΙΣΤΟΣ - ΟΝΤΟΛΟΓΙΕΣ

2.1 Γενική Περιγραφή

Ο Παγκόσμιος Ιστός αποτελεί ένα παγκόσμιο μέσο δεδομένων στο οποίο μπορούν οι χρήστες να ανακτήσουν ή/και να τοποθετήσουν νέα δεδομένα. Μέσω ενός προγράμματος περιηγητή (browser), μπορούν να προβάλουν ιστοσελίδες με κείμενο, ήχο, εικόνα και άλλα πολυμέσα. Λόγω της ευκολίας του, του πλήθους εφαρμογών που υποστηρίζει και την ευκολία πρόσβασης έχει επηρεάσει πολλές πτυχές της ζωής μας. Από τα επαγγελματικά και τη διασκέδαση μέχρι τις κοινωνικές σχέσεις. Παρόλα αυτά, υπάρχουν κάποιες εργασίες που δεν μπορούν να πραγματοποιηθούν ή είναι πολύ δύσκολο να πραγματοποιηθούν στον Παγκόσμιο Ιστό. Η αναζήτηση των πιο κατάλληλων αγαθών μέσα σε ένα τεράστιο σύνολο πηγών είναι ένα από αυτά. Για παράδειγμα, ένας χρήστης μπορεί να βρει την αντίστοιχη ρώσικη για την ελληνική λέξη ‘αυτοκίνητο’ ή να αγοράσουν ένα DVD. Ένας υπολογιστής ή πρόγραμμα λογισμικού δεν είναι σε θέση να το κάνει αυτό. Έτσι ο χρήστης, για να πραγματοποιήσει το σκοπό του θα πρέπει να αναλωθεί σε μια χρονοβόρα (λόγω τεραστίου όγκου πληροφοριών), κουραστική (λόγω του ότι θα πρέπει από μόνος του να ξεκαθαρίσει τις πληροφορίες που τον ενδιαφέρουν από τις άχρηστες γι’ αυτόν) αναζήτηση. Αυτά τα προβλήματα μπορούν να λυθούν με την χρήση και την επέκταση του ΣΙ.

Ένας γενικός ορισμός του ΣΙ είναι η απόδοση νοήματος στις πληροφορίες, ώστε να γίνεται αντιληπτή από τις μηχανές. Ο απώτερος σκοπός του, είναι η ύπαρξη λογισμικού το οποίο να αντιλαμβάνεται τους σκοπούς και τις επιθυμίες του χρήστη, και να τους εκτελεί αυτόνομα δίχως καμία παρέμβαση. Ουσιαστικά η αναζήτηση αγαθών και πληροφοριών σε διάφορες τοποθεσίες του Ιστού, που γίνεται μέχρι σήμερα από το χρήστη, θα γίνεται μέσω τεχνολογιών και οντοτήτων λογισμικού. Έτσι ο χρήστης θα έχει το αποτέλεσμα που θέλει ταχύτερα και δίχως κόπο.

Ο Tim Berners-Lee, που πρώτος εμπνεύστηκε το ΣΙ είπε χαρακτηριστικά:

“*I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A ‘Semantic Web’, which should make this possible, has yet to emerge, but when it*

does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The 'intelligent agents' people have touted for ages will finally materialize."
– Tim Berners-Lee, 1999

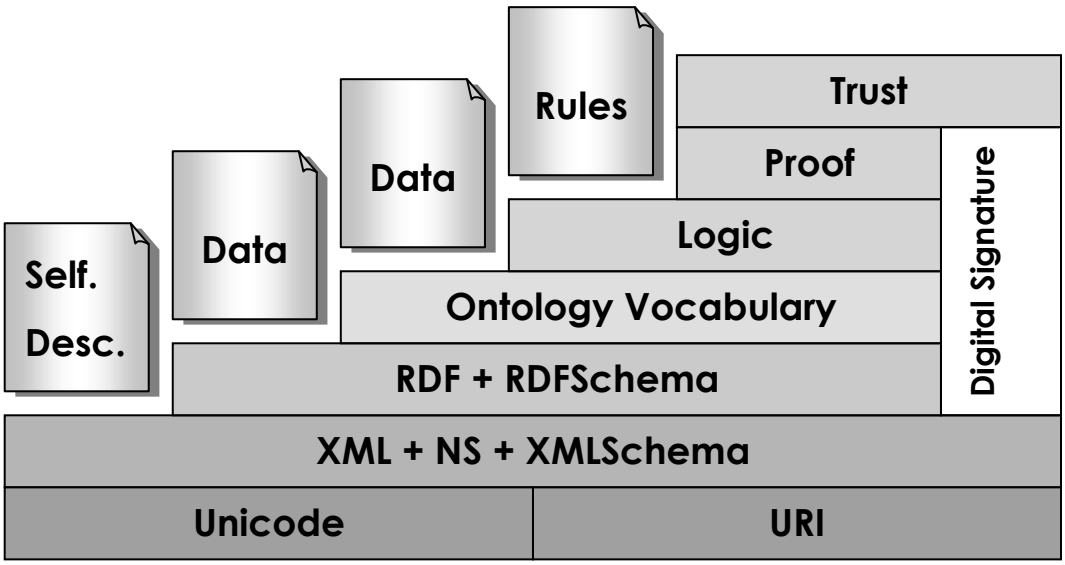
Μέσα από τα λόγια του Berners-Lee, γίνονται προφανείς οι εφαρμογές που μπορεί να έχει, καθώς και η επίδραση του στην καθημερινότητα μας. Ακόμα, ο ΣΙ, που περιγράφεται από τον Berners-Lee ως 'ιστός δεδομένων', σε αντίθεση με τον σημερινό 'ιστό εγγράφων', έχει προοπτικές να δώσει στο χρήστη τη δυνατότητα να επιδράσει στα δεδομένα (να επιλέξει ποια δεδομένα του εμφανίζονται, σε τι μορφή κ.λ.π.). Αναφέρεται σε εφαρμογές φαρμακευτικές και ιατρικές, ακόμα και τετριμμένες όπως το να ξέρεις που είναι οι φίλοι σου σε σχέση με το κοντινότερο μαγαζί. Ακόμα, αναμένεται να φέρει επανάσταση στα επιστημονικά έγγραφα και εργασίες, καθώς και στο διαμοιρασμό και κοινή χρήση πειραματικών δεδομένων.

Το κύριο χαρακτηριστικό του ΣΙ αποτελούν τα μετα-δεδομένα (meta-data ή αλλιώς metainformation). Τα μετα-δεδομένα, είναι 'δεδομένα που περιγράφουν άλλα δεδομένα' οποιαδήποτε μορφής. Ένα αντικείμενο μετα-δεδομένων, μπορεί να περιγράφει ένα κομμάτι πληροφορίας ή μια συλλογή δεδομένων, μαζί με την δομή τους και τις σχέσεις τους. Μπορεί να περιέχει πληροφορίες σχετικά με το περιεχόμενο (σημασία) των δεδομένων, την ποιότητα και κατάσταση ή τα χαρακτηριστικά των δεδομένων. Έτσι, καταφέρνει να κάνει την αόριστη πληροφορία (κείμενο) σαφή, προσφέροντας μια οργάνωση προσπελάσιμη και επεξεργάσιμη από τις μηχανές. Τα μετα-δεδομένα αποθηκεύονται σε μια ιεραρχική διάταξη, με βάση το περιεχόμενο και τη σημασία τους, η οποία ονομάζεται οντολογία (ontology ή schema) [2]. Οι οντολογίες εκτός από τα δεδομένα, περιέχουν και τις σχέσεις μεταξύ τους καθώς και άλλα χαρακτηριστικά, ώστε να γίνεται ευκολότερα η επεξεργασία των σημασιών τους. Έτσι, μέσω των οντολογιών γίνεται μια προσπάθεια αναπαράστασης των δεδομένων όπως ακριβώς και στο μυαλό ενός ανθρώπου.

Για παράδειγμα, αν ο χρήστης επιθυμεί την εύρεση μιας διεύθυνσης ηλεκτρονικού ταχυδρομείου (electronic mail ή e-mail) ενός καθηγητή του τμήματος Πληροφορικής και Τηλεπικοινωνιών, θα πρέπει να μπει στον επίσημο ιστότοπο της σχολής και να κάνει μια αναζήτηση στις σελίδες του μέχρι να βρει την διεύθυνση που αναζητεί. Αυτό γίνεται διότι, το παρόν σύστημα κωδικοποίησης για δημιουργία ιστοσελίδων ή HTML (Hyper-Text Marking Language) έχει λειτουργίες που αφορούν μόνο τον τρόπο εμφάνιση των

δεδομένων δίχως να ξεχωρίζουν τα δεδομένα μεταξύ τους, όσον αφορά τις μηχανές. Ο ΣΙ κάνει την μετάβαση από την απλή εμφάνιση των δεδομένων, στην απόδοση σημασίας σε αυτά, με την χρήση διαφόρων ετικετών με περιγραφές. Στο παράδειγμα αυτό, ενώ η HTML θα χρησιμοποιήσει την ετικέτα <link> που χρησιμοποιεί για όλες τις συνδέσεις, ο ΣΙ θα χρησιμοποιήσει την ετικέτα <mailing address> που δηλώνει τον τύπο της πληροφορίας. Ακόμα, προσφέρει τη δυνατότητα για εύρεση σχέσεων μεταξύ των πληροφοριών αυτών με την χρήση κάποιων κανόνων αλλά και εργαλείων που περιέχουν οργανωμένες τις πληροφορίες και τις σχέσεις τους. Τα εργαλεία αυτά ονομάζονται οντολογίες και θα τα εξετάσουμε αργότερα στην παράγραφο 2.2.
Έτσι τελικά, οι μηχανές θα μπορούν να κάνουν περισσότερες λειτουργίες που προς το παρόν γίνονται από τους χρήστες αποκλειστικά, όπως αναζήτηση, σύνθεση, διασταύρωση και αντιστοίχιση πληροφοριών με την χρήση έξυπνων πρακτόρων λογισμικού. Θα 'αντιλαμβάνονται' πλέον τις έννοιες, θα μπορούν να συνδυάσουν διάφορες έννοιες από διάφορους ιστοτόπους και τελικά να παρουσιάσουν ένα σύνολο πληροφοριών που βρίσκονται διάσπαρτα στον ιστό. Για παράδειγμα, αν ζητήσω από ένα πράκτορα για πληροφόρηση για ένα ταξίδι που θα κάνω, μπορεί να μου παρουσιάσει προτάσεις όπως ρούχα που πρέπει να πάρω μαζί μου, τα ονόματα φίλων που βρίσκονται ήδη εκεί κ.λ.π. Τελικά, ο σκοπός του ΣΙ, με μια φράση, είναι η Ενοποίηση Δεδομένων (Data Integration).

Η αρχιτεκτονική του ΣΙ, προτάθηκε από τον Tim Berners-Lee στο συνέδριο XML 2000 [96]. Σχηματικά, τα επίπεδά του είναι τα παρακάτω:



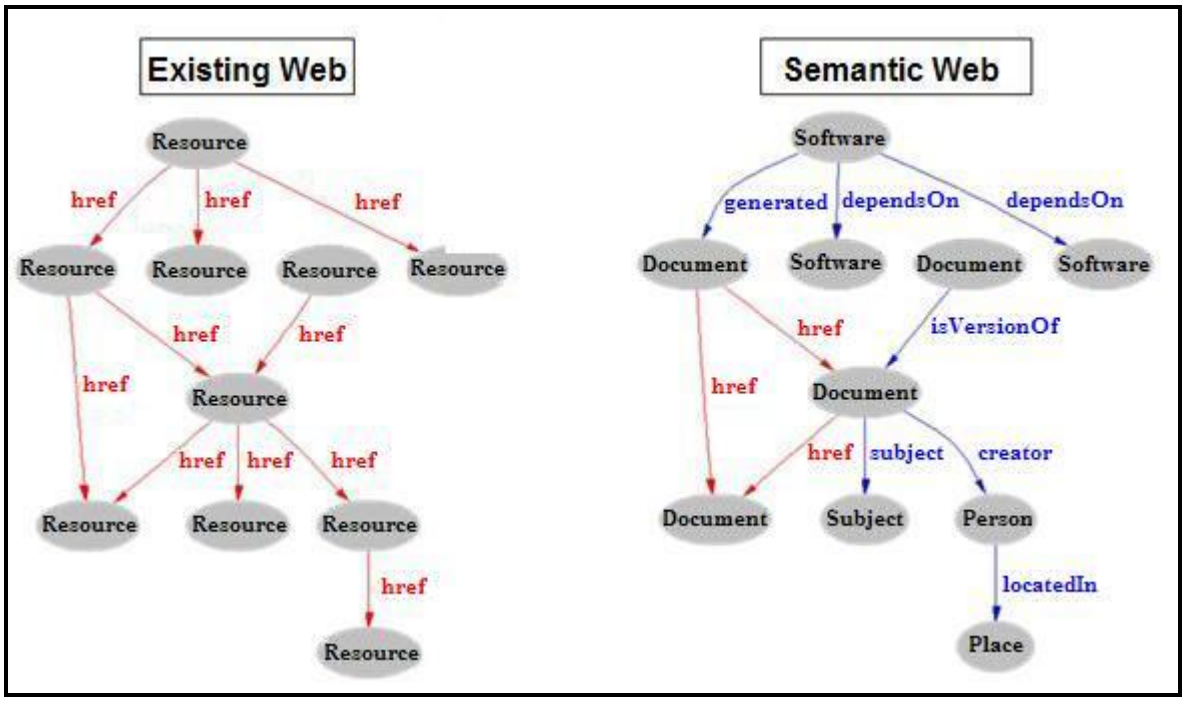
Εικόνα 2.1 Διάρθρωση των επιπέδων του Σημασιολογικού Ιστού.

Αναλυτικότερα τα περιεχόμενα του καθενός τμήματος έχουν ως εξής:

- **XML-XMLSchema (Extensible Markup Language)**: Αποτελεί την γνωστή γλώσσα περιγραφής και ανταλλαγής δεδομένων στο Διαδίκτυο. Το Schema κομμάτι είναι ένα πλαίσιο με το οποίο κανείς καθορίζει την συντακτική δομή των XML εγγράφων.
- **RDF-RDFSchema (Resource Description Framework)**: Μια ευέλικτη γλώσσα, ικανή για την περιγραφή όλων των ειδών πληροφορίας και μετα-δεδομένων. Η RDFSchema, είναι μια γλώσσα που προσφέρει τρόπους για τον καθορισμό του βασικού λεξικού με το οποίο εκφράζονται οι κατηγορίες των πόρων, οι πόροι, οι ιδιότητές τους και οι σχέσεις μεταξύ τους.
- **Ontology**: Αποτελεί μια οργάνωση των πληροφοριών και των μεταξύ τους σχέσεων. Η οργάνωση αυτή γίνεται σε μια δομή με κατηγορίες, υπο-κατηγορίες κ.λ.π. Ο όρος προέρχεται από τον τομέα της Φιλοσοφίας. Μια πλήρης περιγραφή των οντολογιών δίνεται στην παράγραφο 2.2.
- **Logic and Proof**: Ο λογικός συμπερασμός χρησιμοποιείται για την εξαγωγή συμπερασμάτων που δεν δηλώνονται ρητά και για την απόδοση λογικής και συνέπειας στο σύνολο των δεδομένων. Η απόδειξη επεξηγεί ή ιχνηλατεί τα βήματα του λογικού συμπερασμού. Μπορεί να χρησιμοποιηθεί από άλλα συστήματα για την επιβεβαίωση των βημάτων του συμπερασμού.
- **Trust**: Περιλαμβάνει τρόπους αυθεντικοποίησης και απόδειξης της ταυτότητας και της ακεραιότητας των δεδομένων και των υπηρεσιών. Ένας μηχανισμός αυθεντικοποίησης είναι οι ψηφιακές υπογραφές.

Στην βάση της ιεραρχίας περιμένουμε η πλειοψηφία των δεδομένων να έχει περιγραφεί με την γλώσσα XML. Κάθε επίπεδο είναι προοδευτικά πιο εξειδικευμένο και συνεπώς τείνει να είναι πιο περίπλοκο από το επίπεδο που βρίσκεται ακριβώς από κάτω του. Επίσης τα κατώτερα επίπεδα δεν εξαρτώνται από τα ανώτερα, οπότε η ανάπτυξη τους γίνεται χωριστά και ανεξάρτητα. Τέλος, θα πρέπει να σημειωθεί ότι το παραπάνω σχήμα αποτελεί την οπτική γωνία του World Wide Web Consortium (W3C) [90]. Συνεπώς, μπορούν να υπάρξουν εναλλακτικές λύσεις για κάποια από τα αναφερόμενα επίπεδα. Για παράδειγμα, ήδη υπάρχουν προσπάθειες δημιουργίας εναλλακτικών XML σχημάτων για την ανάπτυξη οντολογικών συστημάτων.

Συνοψίζοντας, στην εικόνα 2.2 παρουσιάζεται μια σχηματική αντιπαράθεση του Παγκόσμιου Ιστού με τον ΣΙ.



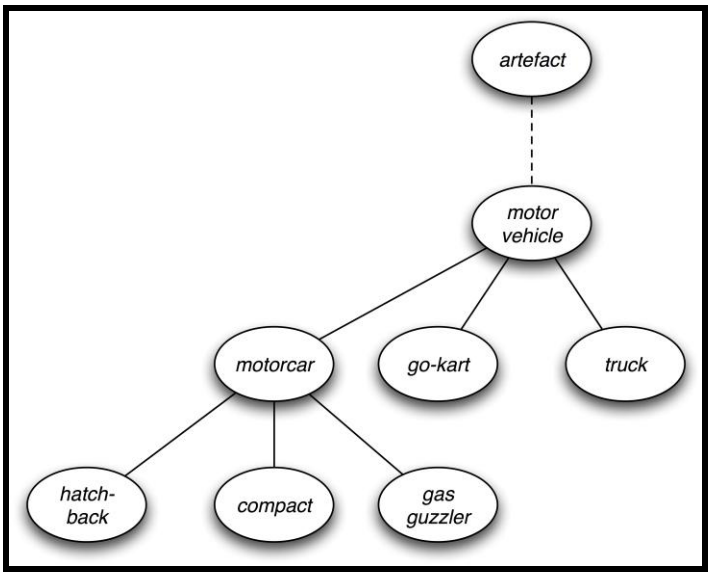
Εικόνα 2.2 – Παγκόσμιος Ιστός και Σημασιολογικός Ιστός.

2.2 Οντολογίες

Ο όρος της οντολογίας έχει τις ρίζες του στην αρχαία φιλοσοφία και αποτελούσε μια μελέτη της ύπαρξης. Στον τομέα των υπολογιστικών επιστημών, αναφέρεται σε ένα μοντέλο, που περιγράφει τον κόσμο το οποίο αποτελείται από τύπους, ιδιότητες και σχέσεις. Αποτελεί μια οργάνωση των πληροφοριών κατά τέτοιο τρόπο ώστε να μπορούν να γίνουν αντιληπτές οι έννοιες και η σύνδεση μεταξύ τους από τις μηχανές. Μέσα στις οντολογίες γίνεται μια αναπαράσταση των δεδομένων κατά τέτοιο τρόπο ώστε να πλησιάζει όσο το δυνατόν περισσότερο στον ανθρώπινο τρόπο αντίληψης τους. Χρησιμοποιείται κυρίως για αποθήκευση της γνώσης σε μια δομή, από την οποία αργότερα μπορεί να εξαχθούν κάποια συμπεράσματα, από τους έξυπνους πράκτορες του ΣΙ.

Μια κλασική οντολογία, η οποία χρησιμοποιείται ευρέως από ερευνητές της ανάλυσης κειμένου, της υπολογιστικής γλωσσολογίας και άλλων σχετικών περιοχών, αποτελεί το WordNet [94]. Το WordNet είναι ένα λεξικό, το οποίο περιέχει την πλειοψηφία των αγγλικών λέξεων, οι είναι διαρθρωμένες σε κάποια μορφή σημασιολογικού δικτύου, όπου οι συνδέσεις μεταξύ των κόμβων αναπαριστούν κάποια συγκεκριμένη σχέση

μεταξύ των εννοιών**.** Ακόμα, οι έννοιες οι οποίες έχουν πανομοιότυπη σημασία ομαδοποιούνται μεταξύ τους και σχηματίζουν κάποια σύνολα συνωνύμων (synonyms sets - synsets). Ένα synsets αναπαριστά μια διακριτή έννοια (sense ή concept). Επίσης περιέχει συνδέσεις μεταξύ των συνόλων για να περιγράψει σχέσεις όπως υπέρνυμα, υπόνυμα, ολόνυμα, περιγραφές των συνωνύμων με συμβολοσειρές κ.λ.π. Το WordNet περιέχει κάποιες γενικές έννοιες, όπως οι *Οντότητα, Κατάσταση, Γεγονός* οι οποίες είναι οι βασικές, γενικές και βρίσκονται στην κορυφή της ιεραρχίας, δηλαδή, αποτελούν κοινό πρόγονο όλων των εννοιών. Μια γραφική αναπαράσταση ενός τμήματος του φαίνεται στο παρακάτω σχήμα:



Εικόνα 2.3 – Γραφική αναπαράσταση τμήματος του WordNet.

Αν παρατηρήσουμε την εικόνα 2.3, γίνεται φανερή η ταξινόμηση των εννοιών, όπως και η σταδιακή μετακίνηση από μια γενικότερη έννοια σε ειδικότερη καθώς ταξιδεύουμε από υψηλότερο τμήμα της δομής σε χαμηλότερο.

Οι οντολογίες όπως περιγράψαμε και πάνω χρησιμοποιούνται για να περιγράψουν πληροφορίες και τις σχέσεις τους όπως γίνεται και στο μυαλό ενός ανθρώπου. Πολλοί ερευνητές έχουν επισημάνει τις ομοιότητες μεταξύ των οντολογιών και του αντικειμενοστραφή προγραμματισμού (Object-Oriented Programming – OOP). Και οι δύο κατευθύνσεις αποσκοπούν στην αντιστοίχιση του πραγματικού κόσμου στον εικονικό, και παρουσιάζουν την ίδια νοοτροπία για αυτήν. Ακόμα, και οι δύο χρησιμοποιούν πανομοιότυπα εργαλεία, όπως κλάσεις, στιγμιότυπα και ιδιότητες.

Παρακάτω, παρουσιάζονται τα κυριότερα χαρακτηριστικά των οντολογιών σε σύγκριση με την πραγματικότητα, που προσομοιώνουν:

- **Κλάσεις:** Οι κλάσεις αποτελούν ένα τρόπο αναπαράστασης ομάδων αντικειμένων της πραγματικότητας με τέτοιο τρόπο, ώστε να γίνεται αντιληπτό από τις οντότητες λογισμικού. Με μια κλάση μπορούμε να αναπαραστήσουμε έννοιες, τύπους αντικειμένων ή σύνολα δεδομένων. Για παράδειγμα, στην εικόνα 2.3 ο κόμβος truck αντιστοιχεί στην πραγματική έννοια του φορτηγού, ενώ ο κόμβος motor vehicle αναφέρεται στο σύνολο όλων των μηχανοκίνητων. Ένα άλλο παράδειγμα, είναι η έννοια 'άνθρωπος', όπου αφορά το σύνολο των ανθρώπων. Ακόμα, υπάρχουν κάποιες σχέσεις μεταξύ των κλάσεων όπως υπο-κλάση (υπο-σύνολο, μια έννοια πιο ειδική από την κλάση), υπέρ-κλάσεις (υπέρ-σύνολα), συμπληρωματικές έννοιες κ.λ.π. Οι κλάσεις περιέχουν ιδιότητες και λειτουργίες οι οποίες χαρακτηρίζουν τις έννοιες, όπως ακριβώς και στον OOP.
- **Ιδιότητες:** Οι ιδιότητες αποτελούν την αναπαράσταση των ιδιοτήτων, των χαρακτηριστικών ή/και των παραμέτρων κάποιων εννοιών ή αντικειμένων. Ορίζονται με κάποιο πεδίο ορισμού (domain), που είναι το αντικείμενο στο οποίο αναφέρεται και κάποιο πεδίο τιμών (range), όπου είναι ένας ορισμός των τιμών που μπορεί να πάρει. Για παράδειγμα, η κλάση 'άνθρωπος' που είδαμε παραπάνω, μπορεί να περιέχει τις ιδιότητες 'ύψος', 'υπηκοότητα', 'Γέννησης', 'πατέρας' αφού όλοι οι άνθρωποι έχουν αυτά τα χαρακτηριστικά. Η ιδιότητα 'ύψος' μπορεί να πάρει σαν τιμές(πεδίο τιμών) ακεραίους, π.χ. 179 για να δηλώσει τα εκατοστά του ύψους ή δεκαδικούς π.χ. 1.79 για να δηλώσει τα μέτρα. Παρόλα αυτά δεν μπορεί να πάρει την τιμή 'κόκκινο'. Αντίστοιχα, τα πεδία τιμών των υπολοίπων ιδιοτήτων θα μπορούσε να είναι συμβολοσειρά, ημερομηνία και κάποιο άλλο αντικείμενο της κλάσης 'άνθρωπος'.
- **Στιγμιότυπα:** Τα στιγμιότυπα αποτελούν την αντιστοίχιση των αντικειμένων-ατόμων της πραγματικότητας στους υπολογιστές, όπως ακριβώς και οι κλάσεις αποτελούν αντιστοίχιση των εννοιών. Μια κλάση μπορεί να περιέχει πολλά στιγμιότυπα, με τις ίδιες ιδιότητες, αλλά διαφορετικές τιμές. Για το παράδειγμά μας, η κλάση 'άνθρωπος' μπορεί να περιέχει το στιγμιότυπο 'Γιάννης'. Το στιγμιότυπο αυτό θα περιέχει τις ιδιότητες της κλάσης που ανήκει, με κάποιες τιμές. Έτσι, για την ιδιότητα 'ύψος' μπορεί να περιέχει την τιμή 1.79, για την 'υπηκοότητα' τη τιμή 'Ελληνική' και για την 'Γέννησης' την τιμή '06-09-1984'.

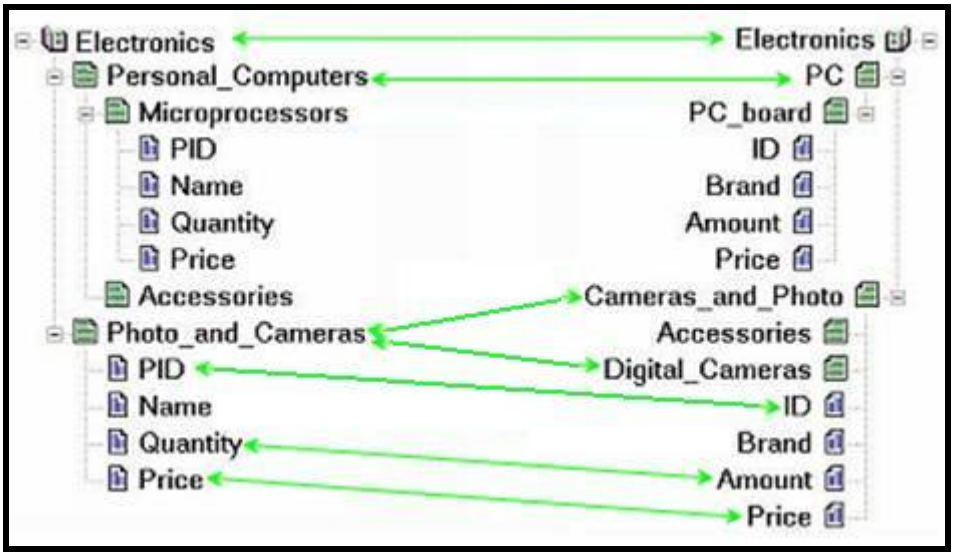
Άλλο στιγμιότυπο μπορεί να έχει διαφορετικές τιμές, αλλά πάντα μέσα στο πεδίο τιμών.

- **Σχέσεις:** Οι σχέσεις αποτελούν τους τρόπους με τους οποίους συνδέονται τα υπόλοιπα χαρακτηριστικά μεταξύ τους. Έχουμε έτσι, σχέσεις κλάσης -υπο-κλάσης, υπερ-κλάσης, συμπληρωματικότητας, υπερ-ιδιότητες, υπο-ιδιότητες κ.λ.π.
- **Εξισώσεις:** Οι εξισώσεις αποτελούν σύνθετες δομές που σχηματίζονται από διάφορες σχέσεις, και μπορούν να αντικαταστήσουν τα στιγμιότυπα σε μια πρόταση.
- **Περιορισμούς:** Κάποιες προτάσεις οι οποίες πρέπει να είναι αληθείς, για να γίνει μια ανάθεση τιμών.
- **Κανόνες:** Δηλώσεις της μορφής if-then που χρησιμοποιούνται κατά την ανάθεση για την εξαγωγή συμπερασμάτων.
- **Αξιώματα:** τα αξιώματα αποτελούν κάποιους κανόνες σε μορφή λογικών προτάσεων, οι οποίοι περιγράφουν την οντολογία. Η διαφορά των αξιωμάτων με τους κανόνες είναι ότι οι κανόνες χρησιμοποιούνται για την εξαγωγή συμπερασμάτων και μπορεί να ισχύουν ή όχι σε μια ανάθεση. Τα αξιώματα αντίθετα περιγράφουν την αρχική (a-priori) γνώση και ισχύουν πάντα για όλες τις αναθέσεις της οντολογίας.

Για την επιτυχή λειτουργία του ΣΙ, θα πρέπει να υπάρχει κάποια επικοινωνία και αντιστοίχιση μεταξύ των οντολογιών. Σημαντικό κομμάτι της επικοινωνίας αυτής αποτελεί το ταίριασμα. Το ταίριασμα Οντολογιών (ontology Matching ή Ontology Alignment) αποτελεί την διαδικασία κατά την οποία επισημαίνονται τα κοινά στοιχεία και οι αντιστοιχήσεις μεταξύ τους. Ιστορικά, η ανάγκη για το ταίριασμα οντολογιών προήλθε από την ανάγκη για ενσωμάτωση ετερογενών βάσεων δεδομένων, οι οποίες δημιουργήθηκαν ανεξάρτητα και η κάθε μια έχει τους δικούς τους τρόπους ονοματοδότησης και δομής, σε μια.

Η διαδικασία αυτή περιλαμβάνει την χρήση κάποιων εργαλείων που εξετάζουν την ομοιότητα των διαφόρων οντοτήτων που περιέχουν οι οντολογίες. Τα εργαλεία αυτά χρησιμοποιούν μια πληθώρα από στρατηγικές για να εξετάσουν την ομοιότητα, οι οποίες βασίζονται στα στοιχεία-σύμβολα που αποτελούν τα ονόματα των οντοτήτων (λεξικογραφική ομοιότητα) είτε στο νόημα που μπορεί να περιέχουν τα ονόματα και οι έννοιες αυτές (σημασιολογική ομοιότητα).

Τα τελευταία χρόνια το ενδιαφέρον προς αυτό το τομέα έχει οδηγήσει σε μεγάλη επιστημονική δραστηριότητα, που με την σειρά του έχει σαν αποτέλεσμα την ανάπτυξη μιας πληθώρας από αλγορίθμους που προσεγγίζουν το ταίριασμα από διαφορετική σκοπιά. Μια πιο λεπτομερείς ανάπτυξη των τεχνικών αυτών, καθώς και των αλγορίθμων για το ταίριασμα οντολογιών παρουσιάζεται στη συνέχεια της εργασίας (Κεφάλαια 4 και 5 αντίστοιχα). Μια μέτρηση του πόσο πετυχημένοι είναι οι αλγόριθμοι αυτοί αποτελούν οι τιμές 'ακρίβειας-ανάκλησης' (precision-recall), καθώς και η τιμή F-Measure. Η σημασία των τιμών αυτών, καθώς και οι τιμές των αλγορίθμων που μελετήσαμε παρουσιάζονται στο κεφάλαιο 7. Στην εικόνα 2.4 παρακάτω, παρουσιάζονται τα αποτελέσματα μιας διαδικασίας ταιριάσματος για δύο πανομοιότυπες οντολογίες.



Εικόνα 2.4 – Παράδειγμα Ταιριάσματος δύο οντολογιών.

Οι οντολογίες θα παίξουν ένα σημαντικό ρόλο στην επόμενη γενιά του Διαδικτύου, λόγω του γεγονότος ότι επιτρέπουν τον διαμοιρασμό της γνώσης μεταξύ των προγραμμάτων. Παραδείγματα χρήσης τους αποτελούν οι δικτυακοί τόποι ηλεκτρονικού εμπορίου, όπου διευκολύνουν τις διάφορες ενέργειες μεταξύ του αγοραστή με τον πωλητή και τις ενέργειες μεταξύ καταστημάτων καθώς και οι εφαρμογές που σχετίζονται με μηχανές αναζήτησης. Με την βοήθεια των οντολογιών η αναζήτηση δεν γίνεται με βάση συντακτικούς προσδιορισμούς από την πλευρά του χρήστη, αλλά με σημασιολογικές μεθόδους που σημαίνει ότι η ύπαρξη πιθανών συντακτικών διαφορών δεν επηρεάζει τα αποτελέσματα.

2.3 Επίλογος

Συνοψίζοντας, ο ΣΙ αποτελεί ένα νέο τμήμα του Παγκόσμιου Ιστού, ο οποίος επιτρέπει την επεξεργασία δεδομένων από τις μηχανές ώστε να διευκολυνθούν πολλές εργασίες του χρήστη. Οι εφαρμογές του είναι πολλές μεταξύ αυτών και στις αναζητήσεις, το ηλεκτρονικό εμπόριο, την εκπαίδευση, τα επιστημονικά πειράματα και μελέτες. Το κυριότερο χαρακτηριστικό του, αποτελούν τα μετα-δεδομένα, τα οποία είναι δεδομένα που περιγράφουν άλλα δεδομένα και δίνουν πληροφορίες για το περιεχόμενο τους. Αποθηκεύονται σε μια συγκεκριμένη, δομημένη διάταξη με βάση τη σημασία και το περιεχόμενο τους, που ονομάζεται οντολογία.

Οι οντολογίες αποτελούν το αποθηκευτικό μέσο των δεδομένων στο ΣΙ. Ο τρόπος οργάνωσης και δομής των πληροφοριών στις οντολογίες προσπαθεί να προσεγγίσει τον ανθρώπινο τρόπο αποθήκευσης αυτών των δεδομένων στον εγκέφαλο. Τα κυριότερα χαρακτηριστικά των οντολογιών αποτελούν οι κλάσεις (που αποθηκεύονται οι έννοιες), οι ιδιότητες (που αποθηκεύονται οι ιδιότητες και τα χαρακτηριστικά των εννοιών και των αντικειμένων), τα στιγμιότυπα (τα αντίστοιχα αντικείμενα και άτομα της πραγματικότητας), οι σχέσεις, οι εξισώσεις, οι περιορισμοί, οι κανόνες και τα αξιώματα. Το ταίριασμα οντολογιών αποτελεί την διαδικασία εύρεσης των κοινών και πανομοιότυπων στοιχείων μεταξύ δύο οντολογιών. Προήλθε από την ανάγκη για ανταλλαγή δεδομένων και αντιστοίχιση μεταξύ των οντολογιών μέσα στο ΣΙ. Για την διαδικασία αυτή έχουν προταθεί διάφορες τεχνικές και αλγόριθμοι. Στα επόμενα κεφάλαια, εξετάζουμε τις κυριότερες τεχνικές, χωρισμένες σε δύο κύριες κατηγορίες, τις λεξιλογικές και τις σημασιολογικές (Κεφάλαιο 4), καθώς και τους σημαντικότερους αλγόριθμους που τις χρησιμοποιούν (Κεφάλαιο 5).

ΚΕΦΑΛΑΙΟ 3

ΗΛΕΚΤΡΟΝΙΚΕΣ ΑΓΟΡΕΣ ΠΡΟΪΟΝΤΩΝ

3.1 Γενική Περιγραφή

Ο Παγκόσμιος Ιστός αποτελεί ένα μέσο παγκόσμια διαδεδομένο σε ένα ευρύ φάσμα κοινού. Η μεγάλη απήχηση που παρουσιάζει, καθώς και η διαδραστικότητα που παρουσιάζει αποτελούν μοναδικές ιδιότητες του μέσου αυτού. Ακόμα η ραγδαία ανάπτυξη των τεχνολογιών καθιστά δυνατές διάφορες εφαρμογές όπως τα συστήματα διαχείρισης των ψηφιακών δεδομένων των πελατών και τα συστήματα ηλεκτρονικής διαχείρισης πελατειακών σχέσεων (e-customer relationship management – eCRM [4]). Αυτά τα πλεονεκτήματα, έχουν ως αποτέλεσμα όλο και περισσότερες επιχειρήσεις να προσανατολίζουν τις εμπορικές τους δραστηριότητες στην προσομοίωση του παραδοσιακού εμπορίου στους υπολογιστές. Έχουν αναπτυχθεί και μελετηθεί κάποια μοντέλα, που ασχολούνται με αυτή τη ζήτηση και προσφορά αγαθών ηλεκτρονικά, καθώς και τη συμπεριφορά των οντοτήτων που τις παρουσιάζουν. Ένα από αυτά είναι και το μοντέλο της ηλεκτρονικής αγοράς(Electronic Marketplace – EM) .

Με τον όρο ηλεκτρονική αγορά εννοούμε κάποιες εικονικές τοποθεσίες, όπου γίνονται ανταλλαγές αγαθών, μεταξύ των οντοτήτων, συνήθως με κάποιο αντίτιμο. Τα αγαθά αυτά μπορεί να είναι εκτός από υλικά προϊόντα και έγγραφα, βίντεο, μουσική, προγράμματα κ.λ.π. Οι ηλεκτρονικές αγορές αποτελούν συστήματα στα οποία οι οντότητες που τις αποτελούν είναι αυτόνομες και ανεξάρτητες δίχως να ελέγχονται από μια άλλη οντότητα. Έτσι πετυχαίνεται μεγαλύτερη ευελιξία στη χρήση και την λειτουργία τους.

Οι ηλεκτρονικές αγορές, προσομοιώνουν τους παραδοσιακούς τρόπους αγοραπωλησιών που ισχύουν σήμερα. Έτσι έχουμε παρόμοια χαρακτηριστικά και λειτουργίες. Όπως και στο παραδοσιακό εμπόριο, σε ένα περιβάλλον ηλεκτρονικού εμπορίου υπάρχουν δύο κυρίως είδη οντοτήτων που συμμετέχουν: οι πελάτες, οι οποίοι επιθυμούν να παραλάβουν κάποιο αγαθό ή υπηρεσία και οι πωλητές οι οποίοι έχουν στην διάθεση τους ένα συγκεκριμένο αριθμό αγαθών και υπηρεσιών και επιθυμούν να τις πουλήσουν για κάποιο αντίτιμο. Η κάθε οντότητα από αυτές έχει κάποιους σκοπούς που προσπαθεί να εκπληρώσει, και κάποιο αντίτιμο που είναι διαθέσιμο να πληρώσει

για να εκπληρωθούν οι σκοποί αυτοί. Ένα άλλο κοινό χαρακτηριστικό της ηλεκτρονικής αγοράς με την παραδοσιακή αγορά, είναι η μεγάλη δυναμικότητα. Με τον όρο δυναμικότητα εννοούμε πως κάθε στιγμή μπορεί να αλλάξει ο αριθμός των οντοτήτων, είτε είναι πελάτες, είτε πωλητές, καθώς και οι σκοποί και το αντίστοιχο αντίτιμο τους. Συνήθως αυτά τα χαρακτηριστικά είναι άλληλο-επηρεαζόμενα.

Πέρα από τις όποιες ομοιότητες τους, οι ηλεκτρονικές αγορές προσφέρουν και πολλά πλεονεκτήματα στους χρήστες σε σχέση με το συμβατικό εμπόριο:

- **Μείωση κόστους μεταβίβασης**. Οι παραδοσιακές αγορές επηρεάζονται από κρατικούς μηχανισμούς και ενδιάμεσες οντότητες με αποτέλεσμα την αύξηση του κόστους των αγαθών. Αντίθετα, στην ηλεκτρονική του μορφή, δεν υπάρχουν αυτές οι επιρροές με αποτέλεσμα την άμεση αγοραπωλησία με χαμηλότερο κόστος.
- **Μείωση αναποτελεσματικότητας**. Σε πολλές περιπτώσεις η αναζήτηση πληροφοριών και αγαθών από τους χρήστες είναι δύσκολη λόγω του μεγάλου όγκου πληροφοριών και της μη ταξινόμησης τους. Έτσι, ένας χρήστης μπορεί να σπαταλήσει άσκοπα το χρόνο του σε μια αναζήτηση με αμφίβολα αποτελέσματα, λόγω ανθρώπινου λάθους π.χ. να μην γνωρίζει την κατάλληλη τοποθεσία που μπορεί να βρει το προϊόν που χρειάζεται κ.λ.π.
- **Μείωση του χρόνου.** Στην ηλεκτρονική τους μορφή οι αγορές προσφέρουν σημαντικά πλεονεκτήματα από άποψη χρόνου, αφού δεν χρειάζεται ο χρήστης να σπαταλήσει χρόνο για την αναζήτηση και την μετακίνηση του στο κατάλληλο κατάστημα. Μπορεί να έχει πρόσβαση στα αγαθά μέσα από τη σιγουριά του σπιτιού του.
- **Ευκολότερη η εύρεση των πωλητών,** μέσα από το ηλεκτρονικό περιβάλλον.
- **Αύξηση κέρδους των οντοτήτων λόγω της μείωσης ενδιάμεσων βημάτων.** Στις παραδοσιακές αγορές, υπάρχουν ένας αριθμός από μεσάζοντες όπως μεταφορείς που αυξάνουν την τελική τιμή των προϊόντων. Αντίθετα, στην ηλεκτρονική αγορά οι μεταβιβάσεις αγαθών είναι άμεσες, με αποτέλεσμα να έχουμε χαμηλότερο κόστος από πλευράς καταναλωτή, και μεγαλύτερο κέρδος από πλευράς πωλητή.

Τα πλεονεκτήματα αυξάνονται περισσότερο αν συνδυάσουμε τις ηλεκτρονικές αγορές με τους έξυπνους πράκτορες(Intelligent Agents – IA [67], [68]), τους οποίους θα παρουσιάσουμε στη συνέχεια.

Παρά τις όποιες ομοιότητες με τις παραδοσιακές αγορές, οι ηλεκτρονικές έχουν ως περιβάλλον ένα ηλεκτρονικό μέσο, και όχι την πραγματικότητα, και έτσι πρέπει να υιοθετηθούν κάποιες λειτουργίες ώστε να εξασφαλίζεται η σωστή λειτουργία. Έτσι, εκτός των δύο βασικών οντοτήτων, πελάτης και πωλητής, υπάρχουν και οι ενδιάμεσες οντότητες οι οποίες χρησιμοποιούνται κυρίως για βοηθητικούς σκοπούς και τη διαχείριση της αγοράς. Ακόμα, θα πρέπει να υπάρχουν κάποιοι μηχανισμοί στην αγορά, ώστε να επιτυγχάνονται οι ασφαλείς δοσοληψίες. Οι μηχανισμοί αυτοί μπορεί να είναι εγγραφή των χρηστών(για να υπάρχουν στοιχεία για αυτούς σε περίπτωση απάτης), μηχανισμοί εμπιστοσύνης για τους χρήστες, εξασφάλιση ασφαλούς δίαυλου μεταξύ χρήστη και πωλητή όταν αποφασιστεί μια δοσοληψία κ.λ.π.

3.2 Οντότητες που συμμετέχουν σε ηλεκτρονικές αγορές

Οι ηλεκτρονικές αγορές αποτελούν μέρη διακίνησης προϊόντων και αγαθών. Λόγω της αυτονομίας των οντοτήτων που τις αποτελούν υπάρχει μεγάλη αλληλό-επιρροή μεταξύ τους, καθώς και μεταξύ των κινήσεων που ακολουθούν. Για παράδειγμα, μεγάλη ζήτηση ενός συγκεκριμένου αγαθού, μπορεί να επηρεάσει το κόστος απόκτησης του. Σε αυτή την ενότητα περιγράφουμε τις οντότητες που υπάρχουν σε μια αγορά πληροφοριών, καθώς και τα βασικά χαρακτηριστικά και σκοπούς τους.

3.2.1 Πελάτες ή Καταναλωτές

Οι πελάτες, είναι οντότητες οι οποίες συνήθως επιθυμούν ένα αγαθό-υπηρεσία. Είναι διατεθειμένοι να δώσουν ένα αντίτιμο για την απόκτησή του. Το αντίτιμο αυτό, αντιστοιχεί σε μια προσωπική αποτίμηση για το αγαθό αυτό. Η τιμή αυτή, θεωρείται λογικό αντίτιμο για το αγαθό, και πάνω από αυτήν δεν δέχεται συνήθως ο καταναλωτής την δοσοληψία, λόγω του ότι θεωρείται πλέον ασύμφορη. Οι αγορές και πωλήσεις, γίνονται σε μονάδες που είναι γνωστές από πριν σε όλες τις οντότητες της αγοράς. Ο κάθε πελάτης ενδιαφέρεται μόνο για την απόκτηση του αγαθού που τον ενδιαφέρει, δίχως να ενδιαφέρεται για τις προτιμήσεις των υπολοίπων. Οι καταναλωτές έχουν τις δικές τους προτιμήσεις σε αγαθά, χωρίς απαραίτητα να συμπίπτουν μεταξύ τους. Σκοπός τους, αποτελεί η απόκτηση του αγαθού της προτίμησης τους, με όσο το δυνατόν λιγότερο κόστος.

3.2.2 Παροχείς Αγαθών ή Πωλητές

Οι Παροχείς είναι οντότητες που έχουν ένα αριθμό από αγαθά πληροφοριών στην κατοχή τους, τα οποία είναι διατεθειμένοι να πουλήσουν αν η τιμή είναι κατάλληλη. Έχουν ένα αρχικό κόστος για την απόκτηση των αγαθών αυτών, το οποίο προσπαθούν να αποσβήσουν με την πώληση τους. Γενικός σκοπός τους είναι να πουλήσουν αυτά τα αγαθά με όσο το δυνατόν πιο επικερδή τρόπο γίνεται. Έτσι, το αντίτιμο στο οποίο προσφέρουν το κάθε αγαθό, εμπεριέχει το κόστος απόκτησης του αγαθού από τη πηγή, καθώς και μια τιμή κέρδους. Θα μπορούσαμε να πούμε λοιπόν, πως η τιμή κόστους του κάθε αγαθού, αποτελεί ένα κατώτατο όριο για τις συναλλαγές σχετικά με αυτό το αγαθό, αφού κάτω από αυτό παύει να έχει κέρδος ο πωλητής. Το αντίτιμο της πώλησης επηρεάζεται από το κόστος απόκτησης του αγαθού από την πηγή, καθώς και από τη ζήτηση του συγκεκριμένου αγαθού.

3.2.3 Ενδιάμεσες οντότητες

Οι ενδιάμεσες οντότητες είναι βοηθητικές οντότητες με σκοπό τη διαχείριση, δηλαδή την διευκόλυνση αλλά και την ασφαλή λειτουργία της αγοράς. Ουσιαστικά, γεφυρώνουν το χάσμα μεταξύ των υπολοίπων. Χρησιμοποιούνται από τους καταναλωτές για την εύρεση των κατάλληλων αγαθών αναλόγως με τις προτιμήσεις και τις αποτιμήσεις του χρήστη. Ακόμη, χρησιμοποιούνται για την διακίνηση των διαφημιστικών των πωλητών. Οι ενδιάμεσες οντότητες μπορεί να είναι μεσίτες (brokers), οντότητες που ταιριάζουν τις απαιτήσεις των καταναλωτών με τις περιγραφές προϊόντων των πωλητών. (matchmakers), οντότητες αγορών (Shopping Bots - shopbots) ή άλλα διαχειριστικά εργαλεία.

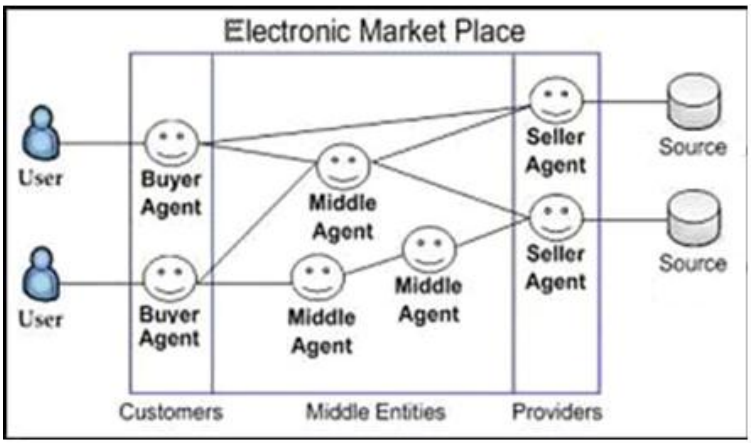
Οι brokers έχουν σαν σκοπό τους τον συντονισμό των ανταλλαγών των αγαθών, την αγορά από τους προμηθευτές για την επαναπώληση στους πωλητές καθώς και την υποστήριξη των διαπραγματεύσεων [76]. Οι matchmakers παρέχουν την ακριβής τοποθεσία-διεύθυνση των πωλητών δίχως άλλη παρέμβαση. Τα shopbots ή αυτόματες μηχανές αγοράς, αναλαμβάνουν να βρουν τα προϊόντα τα οποία ανταποκρίνονται στις απαιτήσεις των χρηστών, καθώς και να τους τα παρουσιάσουν

Όλες οι παραπάνω οντότητες μπορούν να αντικατασταθούν με οντότητες λογισμικού(έξυπνους πράκτορες), το οποίο έχει όλα τα χαρακτηριστικά και τους σκοπούς που περιγράψαμε παραπάνω. Να σημειωθεί επίσης πως οι ρόλοι και οι ονομασίες δεν είναι δηλωτικοί, δηλαδή δεν δηλώνουν επακριβώς τις πράξεις των οντοτήτων, αφού μια οντότητα μπορεί να έχει παραπάνω από ένα ρόλους π.χ. να είναι καταναλωτής και πωλητής ταυτόχρονα.

3.3 Αυτόματες οντότητες αγορών

Οι έξυπνοι πράκτορες(Intelligent Agents - IA) σε μια εικονική αγορά, αποτελούν λογισμικό το οποίο αντικαθιστά τις οντότητες που υπάρχουν σε ένα τέτοιο περιβάλλον, και προσομοιώνουν την συμπεριφορά τους. Οι χρήστες δίνουν τις προτιμήσεις τους, καθώς και περιορισμούς στη συμπεριφορά του πράκτορα, αναλόγως με τους σκοπούς τους. Ένας έξυπνος πράκτορας, αποτελεί μια αυτόνομη οντότητα, που χρησιμοποιεί και ενεργεί επάνω στο περιβάλλον το οποίο βρίσκεται. Χρησιμοποιούν κωδικοποιημένη γνώση ή γνώση που αποκτούν χρησιμοποιώντας το περιβάλλον, με σκοπό την επίτευξη κάποιων στόχων. Οι στόχοι αυτοί συνήθως συμπίπτουν με τους στόχους της οντότητας την οποία αντιπροσωπεύουν στην εικονική αγορά. Χρησιμοποιούνται για την μετάβαση ορισμένων λειτουργιών από το χρήστη στις μηχανές, όπως η αναζήτηση που ειπώθηκε παραπάνω. Πετυχαίνουμε έτσι μια γρηγορότερη, ευκολότερη και πιο πετυχημένη πραγματοποίηση των στόχων του.

Στο σχήμα 3.1 παρουσιάζουμε σχηματικά τη μορφή μιας ηλεκτρονικής αγοράς, όπου οι χρήστες χρησιμοποιούν πράκτορες χρηστών για την διεκπεραίωση των σκοπών τους και οι πωλητές κάνουν χρήση πρακτόρων πωλητών για τους δικούς τους.



Εικόνα 3.1 - Σχήμα Ηλεκτρονικής Αγοράς.

Η χρήση έξυπνων πρακτόρων ως μέλη της αγοράς έχει πολλά πλεονεκτήματα τα οποία είναι:

1. Πρώτα απ' όλα έχουμε μεγάλο κέρδος χρόνου. Οι έξυπνοι πράκτορες διεκπεραιώνουν τις διαπραγματεύσεις σε μικρότερο χρονικό διάστημα από ότι θα μπορούσε ένας χρήστης. Επίσης, αναλαμβάνουν εξ'ολοκλήρου τις

διαπραγματεύσεις, ελευθερώνοντας το χρόνο στους χρήστες για πιο ενδιαφέροντες εργασίες.

2. Είναι αυτόνομα όντα: Αντιλαμβάνονται σκοπούς και μπορούν να εκπληρώσουν εργασίες και να πάρουν κάποιες αποφάσεις, χωρίς καμία παρέμβαση από το χρήστη.
3. Οι έξυπνοι πράκτορες μπορούν να χρησιμοποιήσουν το περιβάλλον τους για να καταλήξουν στην κατάλληλη στρατηγική. Ελέγχουν τις μεθόδους των άλλων πρακτόρων αναλόγως με τα αγαθά και υιοθετούν μια πετυχημένη στρατηγική βασιζόμενοι σε θεωρίες και κωδικοποιημένη γνώση π.χ. τη Θεωρία Παιγνίων (Game Theory) [7]. Έτσι πετυχαίνουν μεγαλύτερο κέρδος σε σχέση με τους ανθρώπους, αφού ένας απλός χρήστης δεν έχει τις κατάλληλες γνώσεις ή υπολογιστική ικανότητα για να το πραγματοποιήσει αυτό.

Εδώ, αξίζει να παρατηρήσουμε πως οι σκοποί των οντοτήτων που περιγράψαμε, δεν είναι προς την ίδια κατεύθυνση, αλλά συχνά άλληλο-συγκρούονται. Οι πελάτες επιθυμούν να αποκτήσουν το αγαθό με όσο το δυνατόν λιγότερο κόστος, ενώ οι πωλητές επιθυμούν να το πουλήσουν σε όσο το δυνατόν περισσότερο κέρδος. Έτσι θα πρέπει να καταλήξουν σε μια συμφωνία που θα συμφέρει και τις δύο πλευρές. Σε τέτοιες περιπτώσεις θα μπορούσαμε να μοντελοποιήσουμε την συμπεριφορά τους με την χρήση του Παιγνίου Διαπραγμάτευσης (Bargaining Game) [15].

Μια κατηγορία έξυπνων πρακτόρων, που μας απασχολεί στη παρούσα εργασία, είναι τα shopbots. Οι αυτόματες μηχανές αγοράς, γνωστότερες ως shopbots αποτελούν μια σημαντική καινοτομία του διαδικτύου για τους καταναλωτές. Τα shopbots αναζητούν (αυτόματα και αποτελεσματικά), σε ένα μεγάλο αριθμό πωλητών, και παρέχουν στο καταναλωτή σχεδόν όλες τις προσφορές, σχετικά με ένα προϊόν, που υπάρχουν μια δεδομένη στιγμή στο διαδίκτυο.

Η ύπαρξη τους προήλθε από την ανάγκη εύρεσης όλων των προϊόντων που συμπίπτουν με τις προτιμήσεις του καταναλωτή. Συνήθως, η εύρεση του κατάλληλου αγαθού το οποίο επιθυμεί ο πελάτης είναι μια διαδικασία πολύ δύσκολη. Επίσης είναι χρονοβόρα και πολλές φορές αναποτελεσματική λόγω της έλλειψης αρκετού χρόνου ή προσοχής από το χρήστη. Αυτό συμβαίνει για τέσσερις κυρίως λόγους:

- Η μεγάλη πληθώρα δεδομένων. Σύμφωνα με εκτιμήσεις, οι σελίδες στον Παγκόσμιο Ιστό ξεπερνούν τα 29.7 δισεκατομμύρια σε αριθμό. Έτσι είναι πολύ δύσκολο να βρεθεί, ένα συγκεκριμένο προϊόν μέσα σε τόσο μεγάλο όγκο δεδομένων.

- Η έλλειψη μιας μορφής αρχειοθέτησης σελίδων. Οι πανομοιότυπες πληροφορίες και αγαθά που πληρούν τις προτιμήσεις του καταναλωτή, μπορεί να βρίσκονται διασκορπισμένα σε διάφορες σελίδες του Παγκόσμιου Ιστού. Έτσι, για να βρεθούν όλα τα προϊόντα που ενδιαφέρεται ο καταναλωτής, και να κάνει σύγκριση για αυτό που τον συμφέρει περισσότερο, χρειάζεται η αναζήτηση σε όλο το κορμό του Παγκόσμιου Ιστού.
- Η έλλειψη μιας μορφής αρχειοθέτησης σε έγγραφα. Οι πληροφορίες είναι ελεύθερες στα κείμενα και στις σελίδες και έτσι η ανεύρεση μιας συγκεκριμένης πληροφορίας σε ένα ηλεκτρονικό έγγραφο συνήθως απαιτεί την ανάγνωση όλου του εγγράφου. Η ακόμα και αν αναζητεί κάποια συγκεκριμένα χαρακτηριστικά στα προϊόντα θα πρέπει να διαβάσει όλο το έγγραφο, για να βρεί τα χαρακτηριστικά αυτά, αφού συνήθως δεν είναι οργανωμένες οι πληροφορίες, αλλά δίνονται όλες μαζί, σαν μια συμβολοσειρά.
- Τέλος, σημαντικό ανασταλτικό παράγοντα αποτελεί και η διαφορά κουλτούρας, γλώσσας κ.λ.π.

Έτσι, ένας χρήστης μπορεί να βρει ένα μικρό ποσοστό των προσφορών που τον ενδιαφέρουν. Με την χρήστη των shopbots έχει πρόσβαση σε όλες τις προσφορές, και έτσι μπορεί να επιλέξει αυτό που ταιριάζει περισσότερο στις προτιμήσεις του. Όπως αναφέρθηκε και παραπάνω, σκοπός ενός καταναλωτή είναι η απόκτηση του αγαθού που τον ενδιαφέρει στη μικρότερη δυνατή τιμή. Έτσι τα shopbots του προσφέρουν περισσότερες επιλογές, με λιγότερο κόπο από μέρους του για να επιλέξει αυτό με την καλύτερη προσφορά. Έρευνες έχουν δείξει πως, η χρήση shopbots, σε σύγκριση με τη παραδοσιακή ακολουθιακή αναζήτηση (που περιγράφηκε απ' τον Stigler – 1961) μειώνει ξεκάθαρα το κόστος απόκτησης των αγαθών.

Ο καταναλωτής (είτε πρόκειται για ανθρώπινο ων είτε αυτόματη οντότητα), παρουσιάζει μια περιγραφή των χαρακτηριστικών των προϊόντων που προτιμά (περιγραφή προϊόντος, εύρος τιμής που είναι αποδεκτό, τοποθεσία ηλεκτρονικού καταστήματος, τρόπος απόκτησης του αγαθού κ.λ.π.). Το shopbot στη συνέχεια αναζητά σε όλες τις τοποθεσίες και αναλαμβάνει να βρει όλα τα προϊόντα που αντιστοιχούν στη περιγραφή αυτή.

Το περιβάλλον της ηλεκτρονικής αγοράς φέρνει σε επαφή μια πληθώρα από πωλητές και αγοραστές. Κάθε μια από αυτές τις οντότητες μπορεί να χρησιμοποιήσει τη δική του δομή για την αναπαράσταση των προϊόντων που τον αφορούν. Τα πολύπλοκα προϊόντα χρειάζονται ένα τρόπο αναπαράστασης ώστε να μπορούν να προσομοιωθούν

όλα τα χαρακτηριστικά τους με τέτοιο τρόπο ώστε να γίνεται αντιληπτό από τα shopbots, ώστε να μπορούν να τα συγκρίνουν με τις απαιτήσεις του καταναλωτή. Ακόμη, για την επιτυχημένη λειτουργία τους, είναι ανάγκη να υπάρξει μια επικοινωνία και ανταλλαγή δεδομένων ώστε να μπορούν να δημιουργηθούν κατάλογοι διαφόρων προϊόντων που υπάρχουν σε διάφορους πωλητές με διαφορετικό τρόπο απεικόνισης. Αυτό γίνεται για να μπορεί ο καταναλωτής να επιλέξει εύκολα και γρήγορα το προϊόν που προτιμά από τα προτιμώμενα. Έτσι, αν υπάρχουν N καταναλωτές και M πωλητές μια δεδομένη στιγμή στο περιβάλλον της αγοράς, θα πρέπει να μπορεί να υπάρξει μια NxM αντιστοίχιση μεταξύ των N καταλόγων προϊόντων των καταναλωτών, και των M δομών απεικόνισης των πωλητών.

Για την αντιμετώπιση των παραπάνω προβλημάτων γίνεται χρήση των οντολογιών. Με τις οντολογίες για την αναπαράσταση των πολύπλοκων προϊόντων, πετυχαίνεται μια ευέλικτη και επεκτάσιμη απεικόνιση τους, καθώς και όλων των ιδιοτήτων και χαρακτηριστικών τους. Ακόμη, με την διαδικασία του ταιριάσματος οντολογιών, που περιγράφηκε σε προηγούμενο κεφάλαιο, μπορεί να γίνει η αντιστοίχιση των διαφόρων δομών, για την δημιουργία καταλόγων. Πιο συγκεκριμένα, τα πλεονεκτήματα της χρήσης οντολογιών για τη λειτουργία των shopbots είναι:

- Η διαδικασία της εύρεσης των κατάλληλων αποτελεσμάτων (match-making) είναι συχνά αναποτελεσματική σε μια ΕΜ, λόγω των άκαμπτων δηλώσεων των προϊόντων δίχως πολλά χαρακτηριστικά. Οι οντολογίες προσφέρουν ένα κοινό, ευέλικτο και επεκτάσιμο ορισμό των προϊόντων και των απαιτήσεων τους.
- Ακόμη, είναι δύσκολη η δήλωση πολύπλοκων απαιτήσεων και προδιαγραφών όσον αφορά στα προϊόντα, επειδή οι σχέσεις μεταξύ των χαρακτηριστικών τους καθώς και οι τιμές τους δεν μπορούν να ληφθούν υπόψη. Με τις οντολογίες, οι πολύπλοκες απαιτήσεις αποσυνθέτονται σε μικρότερες, πιο απλές και έτσι προσφέρουν περισσότερες επιλογές στις αναζητήσεις των καταναλωτών.
- Όσον αφορά τις προτάσεις των πρακτόρων για προϊόντα σχετικά με τις παραμέτρους αναζήτησης των πελατών, είναι πιθανές μόνο μέσα στα πλαίσια μιας κατηγορίας. Ακόμη, είναι δύσκολη η ομαδοποίηση πωλητών και καταναλωτών με παρόμοια ενδιαφέροντα. Με την χρήση των οντολογιών και την μεταφορά δεδομένων μεταξύ τους, πετυχαίνεται μια πιο ευέλικτη ανάλυση των δεδομένων και αφομοίωσή τους από τους πράκτορες. Έτσι, μπορεί να γίνει ευκολότερα η ομαδοποίηση οντοτήτων με παρόμοιες προτιμήσεις, καθώς και των παρόμοιων προϊόντων για προτάσεις προς τους καταναλωτές.

- Μειώνεται κατά πολύ ο χρόνος αναζήτησης των shopbots. Για την εύρεση των κατάλληλων προϊόντων, στις ηλεκτρονικές αγορές, η μηχανή αγοράς θα πρέπει να πάει στις τοποθεσίες των πωλητών, και να ελέγξει όλο το κώδικα HTML για να βρει τα χαρακτηριστικά των προϊόντων. Με τη χρήση οντολογιών για αναπαράσταση τους, αρκεί ένα υπερώτημα για κάθε χαρακτηριστικό.
- Άλλο ένα πλεονέκτημα των οντολογιών αποτελεί ο τρόπος εμφάνισης των αποτελεσμάτων των shopbots στο καταναλωτή. Ο χρήστης θα μπορεί να επιλέξει τη μορφή και τη σειρά που του παρουσιάζονται τα δεδομένα. Ακόμα, θα μπορεί να επιλέξει κάποια από τα προϊόντα αυτά μέσω υπερωτημάτων, εύκολα, δίχως να χρειάζεται να ελέγξει όλα τα αποτελέσματα.

Εν κατακλείδι, αξίζει να σημειώσουμε πως οι μελέτες της χρήσης οντολογιών σε περιβάλλον ηλεκτρονικής αγοράς, έχουν επικεντρωθεί στη διαδικασία του match-making, και πιο συγκεκριμένα της δήλωσης των δεδομένων και της οργάνωσης τους σε μια οντολογία. Στη παρούσα εργασία παρουσιάζουμε μια μελέτη, υπό το πρίσμα της ανταλλαγή δεδομένων μεταξύ διαφόρων οντολογιών ώστε να διευκολυνθούν οι υπόλοιπες διεργασίες των shopbots όπως η δημιουργία καταλόγων με προϊόντα από διάφορες πηγές-πωλητές.

3.4 Επίλογος

Στο παρόν κεφάλαιο εξετάσαμε τις ηλεκτρονικές αγορές, οι οποίες αποτελούν μέρη για πιθανές αγοραπωλησίες προϊόντων. Οι EMs προσομοιώνουν τις παραδοσιακές αγορές στο Παγκόσμιο Ιστό. Έτσι, παρέχουν τη δυνατότητα αγοράς αγαθών, με επιπλέον πλεονεκτήματα όπως η εξοικονόμηση χρόνου από πλευράς χρήστη (δεν αναλώνεται στην αναζήτηση και την μετακίνηση όπως στις παραδοσιακές αγορές), η μεγαλύτερη αποτελεσματικότητα και το αυξημένο κέρδος (λόγω άμεσης δοσοληψίας, δίχως μεσάζοντες).

Οι οντότητες που το συνιστούν είναι οι πελάτες, οι οποίοι ενδιαφέρονται για την απόκτηση ενός αγαθού μέχρι κάποιο συγκεκριμένο αντίτιμο σε όσο το δυνατόν λιγότερο κόστος, οι πωλητές, οι οποίοι ενδιαφέρονται για την πώληση αγαθών τουλάχιστον σε τιμή κόστους, με όσο το δυνατόν μεγαλύτερο αντίτιμο και οι ενδιάμεσες οντότητες που έχουν κυρίως βοηθητικό και ρυθμιστικό χαρακτήρα. Μια κατηγορία ενδιάμεσων οντοτήτων αποτελούν τα shopbots, τα οποία αναλαμβάνουν να βρουν όλα τα προϊόντα, από διάφορους πωλητές, που αντιστοιχούν στις προτιμήσεις των χρηστών.

Οι έξυπνοι πράκτορες αποτελούν ευέλικτο και αυτόνομο λογισμικό, με σκοπό την αντιπροσώπευση των οντοτήτων και την εκπλήρωση των σκοπών τους. Χρησιμοποιούν κωδικοποιημένη γνώση, ή γνώση που προήλθε παρατηρώντας το περιβάλλον τους για να πετύχουν τους στόχους αυτούς. Οι πράκτορες, αυξάνουν τα πλεονεκτήματα των ηλεκτρονικών αγορών, αφού πραγματοποιούν όλες τις συναλλαγές και παίρνουν αποφάσεις δίχως την παρέμβαση των χρηστών, ελευθερώνοντας έτσι το χρόνο τους για άλλες ασχολίες. Ακόμα, μπορούν να πετύχουν μεγαλύτερη αποτελεσματικότητα, αφού μπορούν να παρατηρήσουν τη συμπεριφορά των άλλων πρακτόρων, και να υιοθετήσουν την κατάλληλη συμπεριφορά για να πετύχουν το μεγαλύτερο πιθανό κέρδος.

Οι έξυπνοι πράκτορες μπουν να χρησιμοποιηθούν και ως shopbots, όπου θα αναλαμβάνουν τη διαδικασία match-making και την παρουσίαση των αποτελεσμάτων στους χρήστες. Με τη λειτουργία τους κερδίζεται πολύτιμος χρόνος που θα σπαταλούσαν οι χρήστες για την αναζήτηση, αλλα και μειώνεται ο κόπος τους. Ακόμα, η αναζήτηση γίνεται με περισσότερα αποτελέσματα, προσφέροντας έτσι στο χρήστη τη δυνατότητα να επιλέξει το καλύτερο προιόν για τις προτιμήσεις του.

Η λειτουργία τους μπορεί να βελτιστοποιηθεί με την χρήση οντολογιών για απεικόνιση των προϊόντων στους πωλητές, και την εμφάνιση των αποτελεσμάτων στους καταναλωτές. Με αυτό τον τρόπο πετυχαίνουμε μια πιο ευέλικτη απεικόνιση των δεδομένων αλλά και των χαρακτηριστικών τους. Έτσι, η δήλωση των χαρακτηριστικών των προϊόντων που προτιμά ο καταναλωτής μπορεί να γίνει περισσότερο εξειδικευμένη και να παρουσιαστούν λιγότερα άχρηστα αποτελέσματα. Ακόμα, έχουμε λιγότερο χρόνο εκτέλεσης, αφού για την εύρεση ενός χαρακτηριστικού, δεν χρειάζεται αναζήτηση σε μια ολόκληρη σελίδα HTML, αλλά αρκεί ένα υπερώτημα στην αντίστοιχη οντολογία. Τέλος, η παρουσίαση των αποτελεσμάτων στο χρήστη βελτιώνεται, αφου θα μπορεί να επιλέξει τη σειρά και τη μορφή απεικόνισης καθώς και να πραγματοποιήσει υπερωτήματα στα αποτελέσματα. Στα επόμενα κεφάλαια, παρουσιάζουμε τρόπους επικοινωνίας, αντιστοίχισης και μεταφοράς δεδομένων από μια οντολογία σε μια άλλη, εργασίες απαραίτητες για την επίτευξη των παραπάνω λειτουργιών.

ΚΕΦΑΛΑΙΟ 4

ΤΕΧΝΙΚΕΣ ΛΕΞΙΚΟΓΡΑΦΙΚΗΣ – ΣΗΜΑΣΙΟΛΟΓΙΚΗΣ ΟΜΟΙΟΤΗΤΑΣ

4.1 Γενική Περιγραφή

Το ταίριασμα οντολογιών, είναι σημαντικό για μια πληθώρα εφαρμογών όπως η εξαγωγή πληροφοριών και ο συμπερασμός στις οντολογίες, καθώς και για την επικοινωνία μεταξύ των αυτόνομων οντοτήτων λογισμικού του ΣΙ που αναφέρθηκε πιο πάνω. Γενικά, το κυριότερο πρόβλημα που αντιμετωπίζουν αυτές οι εφαρμογές είναι οι διαφορετικοί τρόποι οργάνωσης και απεικόνισης των δεδομένων. Διαφορετικοί δημιουργοί οντολογιών, έχουν διαφορετικά πιστεύω, συνήθειες, εργαλεία, κάνουν διαφορετικές συμβάσεις κ.λ.π. πράγμα που οδηγεί σε διαφορετική αναπαράσταση ακόμα και παρόμοιων πληροφοριών. Με το ταίριασμα οντολογιών προσπαθούμε να εξαλείψουμε την ετερογένεια αυτή. Με τον όρο ταίριασμα, εννοούμε μια διαδικασία, κατά την οποία έχουμε σαν είσοδο δύο οντολογίες και στην έξοδο έχουμε τις σχέσεις μεταξύ των στοιχείων των οντολογιών αυτών, δηλαδή κατά πόσο όμοιες είναι οι οντολογίες μεταξύ τους και ποια χαρακτηριστικά τους είναι παρόμοια. Οι σχέσεις μπορεί να έχουν μορφή είτε σαν βαθμός ομοιότητας (πιθανότητα ομοιότητας), είτε σαν λογικοί τελεστές (πιο γενικό, πιο ειδικό, ισότητα ή διαφορετικό). Η μορφή αυτή, εξαρτάται από την εκάστοτε τεχνική. Η ομοιότητα μεταξύ των στοιχείων εξαρτάται από τα κοινά χαρακτηριστικά που αυτά περιέχουν καθώς και από τις μεταξύ τους διαφορές. Στην πρώτη περίπτωση η ομοιότητα είναι ανάλογη με τα κοινά χαρακτηριστικά, ενώ στην δεύτερη αντιστρόφως ανάλογη με τις διαφορές τους. Έχουν αναπτυχθεί μια πληθώρα από τεχνικές για να λυθεί το πρόβλημα του ταιριάσματος των οντολογιών. Στη συνέχεια αυτού του κεφαλαίου εξετάζουμε τις σημαντικότερες τεχνικές. Η παρουσίαση αυτή είναι χωρισμένη στις δύο βασικές κατηγορίες των τεχνικών αυτών: τις τεχνικές λεξικογραφικής ομοιότητας καθώς και αυτές της σημασιολογικής ομοιότητας. Για μια εκτενέστερη παρουσίαση τεχνικών, ο αναγνώστης μπορεί να προστρέξει στο [99].

4.2 Λεξικογραφική Ομοιότητα

Η λεξικογραφική ομοιότητα αναφέρεται στην τιμή που παράγεται μέσα από εξέταση δύο στοιχείων διαφορετικών οντολογιών ως προς τα λεξικογραφικά χαρακτηριστικά τους,

δηλαδή τα γράμματα που τις αποτελούν. Οι τεχνικές λεξικογραφικής ομοιότητας χωρίζονται σε δύο βασικές κατηγορίες: αυτούς που υπολογίζουν την απόσταση μεταξύ των λέξεων και αυτούς που υπολογίζουν την ομοιότητα τους.

Η απόσταση μεταξύ δύο συμβολοσειρών (Edit Distance) ορίζεται ως ο αριθμός των αλλαγών (εισαγωγών, διαγραφών και μετατοπίσεων) που χρειάζονται για να μετατραπεί η μία συμβολοσειρά στην άλλη. Για παράδειγμα, αν έχουμε δύο συμβολοσειρές, τις ‘test’ και ‘tend’, η απόσταση τους θα είναι 2 αφού χρειάζονται δύο αντικαταστάσεις (το ‘n’ σε ‘s’ και το ‘t’ σε ‘d’) για να μετατραπεί η πρώτη στη δεύτερη. Ο πιο γνωστός αλγόριθμος υπολογισμού της απόστασης μεταξύ δύο συμβολοσειρών είναι του Levenshtein [47]. Τον αλγόριθμο αυτόν, όπως και διάφορες παραλλαγές του, τον παρουσιάζουμε παρακάτω.

Για τον υπολογισμό της ομοιότητας μεταξύ δύο συμβολοσειρών, συνήθως χρησιμοποιείται το αρχικό στάδιο του υπολογισμού της απόστασης μεταξύ τους, και στη συνέχεια γίνεται κανονικοποιήση ώστε ο αριθμός των μετατροπών να βρίσκεται στο διάστημα [0, 1]. Η κανονικοποιήση αυτή πρέπει να γίνει με το σκεπτικό ότι αν δύο συμβολοσειρές είναι όμοιες (0 μετατοπίσεις), η τιμή που θα παραχθεί θα πρέπει να είναι 1, ενώ σε αντίθετη περίπτωση 0. Οι τιμές μεταξύ τους θα πρέπει να είναι τέτοιες ώστε όσο μεγαλύτερη είναι η τιμή (υψηλή ομοιότητα) τόσο λιγότερες μετατοπίσεις πρέπει να γίνουν.

Συνοψίζοντας λοιπόν, παρατηρούμε πως η πρώτη κατηγορία της απόστασης δηλώνει το πόσο διαφορετικές είναι οι δύο συμβολοσειρές, ενώ η δεύτερη μετασχηματίζει την πρώτη τιμή ώστε να παρουσιαστεί σαν βαθμός ομοιότητας. Στη συνέχεια, παρουσιάζονται οι σημαντικότεροι αλγόριθμοι υπολογισμού της λεξικογραφικής ομοιότητας.

4.2.1 Αλγόριθμος Levenshtein

Όπως ειπώθηκε και παραπάνω, ο αλγόριθμος Levenshtein υπολογίζει την απόσταση μεταξύ δύο συμβολοσειρών. Πιο συγκεκριμένα, δημιουργεί ένα πίνακα με όλους τους συνδυασμούς ζευγαριών συμβόλων, ένα από κάθε συμβολοσειρά. Στη συνέχεια γεμίζει το πίνακα με τιμή 0 αν τα σύμβολα είναι ίσα, ή 1 αν είναι διαφορετικά. Μετά από αυτό το βήμα, υπολογίζει τρεις τιμές για κάθε κελί του πίνακα – ζεύγος συμβόλων – από τις οποίες επιλέγει τη μικρότερη και αντικαθιστά την προηγούμενη τιμή (0 ή 1). Οι τιμές αυτές είναι η τιμή του ακριβώς από πάνω κελιού συν 1, η τιμή του αριστερού κελιού συν ένα και η τιμή του διαγώνιου κελιού πάνω αριστερά συν ένα. Μετά το πέρας αυτών των

βημάτων, η συνολική απόσταση των δύο συμβολοσειρών βρίσκεται από το περιεχόμενο του κατώτατου και δεξιότερου κελιού.

4.2.2 Αλγόριθμος Needleman-Wunch

Ο αλγόριθμος Needleman-Wunch [62] αποτελεί μια παραλλαγή του Levenshtein που ειπώθηκε παραπάνω. Αρχικά δημιουργεί ένα πίνακα όπου κάθε σειρά αντιστοιχεί σε ένα σύμβολο της πρώτης συμβολοσειράς και κάθε στήλη αναφέρεται σε ένα σύμβολο της δεύτερης. Έτσι κάθε κελί του πίνακα αυτού αναφέρεται σε ένα ζεύγος δύο σύμβολων, ένα από κάθε συμβολοσειρά. Έπειτα, γεμίζει κάθε κελί με 1, αν τα δύο σύμβολα είναι ίσα ή 0 αντίθετα. Στο επόμενο βήμα του ο αλγόριθμος ξεκινά από το κάτω δεξιό κελί και προχωρώντας προς τα πάνω και αριστερά διασχίζει όλα τα κελιά. Σε κάθε κελί, αν βρει ομοιότητα (τιμή 1), προχωρά στο επόμενο. Αν βρει ανομοιότητα (τιμή 0) ο αλγόριθμος προσθέτει σε αυτήν ένα κόστος προσαρμογής. Το κόστος προσαρμογής υπολογίζεται ως εξής:

$$D(i,k) = \min \begin{cases} D(i-1,k-1) + d(s_i,t_k) & \text{// για αντικατάσταση ή αντιγραφή} \\ D(i-1,k) + G & \text{// για εισαγωγή} \\ D(i,k-1) + G & \text{// για διαγραφή} \end{cases} \quad (4.1)$$

όπου το $d[s_i,t_k]$ αντιστοιχεί σε μια συνάρτηση απόστασης χαρακτήρων η οποία μπορεί να σχετίζεται για παράδειγμα με τυπογραφικές συχνότητες χαρακτήρων ή στην αντικατάσταση αμινοξέων (αρχική περιοχή έρευνας από την οποία προέκυψε ο αλγόριθμος)και το G αντιστοιχεί στο κόστος μεταβολής. Τέλος, για να βρεθεί η τελική τιμή του, ο αλγόριθμος προσθέτει τις τιμές αυτές και το μονοπάτι με το μεγαλύτερο άθροισμα αναγνωρίζεται ως η ιδανική αντιστοιχία μεταξύ των δύο συμβολοσειρών. Θα πρέπει να σημειωθεί πως υπάρχουν διάφορες παραλλαγές του αλγορίθμου αυτού.

4.2.3 Αλγόριθμος Jaro

Η απόσταση Jaro [38] λαμβάνει υπόψη τυπικούς ορθογραφικούς μετασχηματισμούς σε σχέση με το αρχικό μήκος της κάθε συμβολοσειράς. Η τιμή ομοιότητας για δύο συμβολοσειρές s και t, στον αλγόριθμο αυτό, υπολογίζεται από τον παρακάτω τύπο:

$$\text{Jaro}(s,t) = \frac{1}{3}\left(\frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'| - T_{s',t'}}{|s'|}\right) \quad (4.2)$$

όπου |s| είναι το μήκος της συμβολοσειράς s, |t| το μήκος της συμβολοσειράς t, |s'| το μήκος της ακολουθίας των χαρακτήρων της συμβολοσειράς s που είναι κοινοί με χαρακτήρες της t, |t'| το μήκος της ακολουθίας των χαρακτήρων της συμβολοσειράς t που είναι κοινοί με χαρακτήρες της s, $T_{s',t'}$ είναι το μισό του πλήθους των μετασχηματισμών των s' & t'. Με τον όρο μετασχηματισμοί θεωρούμε μια παραλλαγή της ακολουθίας των s' & t' ώστε σε κάθε θέση i να υπάρχουν χαρακτήρες διαφορετικοί μεταξύ τους. Ακόμη, ένα σύμβολο a_i της μίας συμβολοσειράς είναι όμοιο με ένα σύμβολο της άλλης συμβολοσειράς αν ισχύει $b_j=a_i$ τέτοιο ώστε i-H<= j <=i+H με $H=\frac{\min(|s|,|t|)}{2}$.

Μια παραλλαγή του αλγορίθμου παρουσιάστηκε από τον Winkler (1999), η οποία χρησιμοποιεί το μήκος της μέγιστης ακολουθίας χαρακτήρων που υπάρχει στην αρχή των δύο συμβολοσειρών (prefix). Πιο συγκεκριμένα, η τιμή που υπολογίζεται είναι η εξής:

$$\text{Jaro-Winkler}(s, t) = \text{Jaro}(s, t) + \frac{P'}{10}(1-\text{Jaro}(s, t)) \quad (4.3)$$

όπου P'=max(P,4) και P ισούται με το μήκος του κοινού prefix.

4.2.4 Τεχνική q-gram

Στον αλγόριθμο αυτό, η κάθε συμβολοσειρά θεωρείται ως ένας αριθμός από μικρότερες συμβολοσειρές q μήκους (q-grams). Για παράδειγμα, αν έχουμε q ίσο με 3, η συμβολοσειρά ACDERWA θα περιέχει πέντε 3-grams και πιο συγκεκριμένα τα: ACD, CDE, DER, ERW, RWA. Αφού λοιπόν εξάγουμε τα q-grams, η ομοιότητα δύο συμβολοσειρών s και t, υπολογίζεται με τον εξής τύπο:

$$\text{sim}(s, t) = \frac{|common\ \ qgrams\ \ in\ \ s,t|}{|qgrams\ \ in\ \ s,t|} \quad (4.4)$$

Δηλαδή, ο αριθμός των q-grams που είναι κοινά και στις δύο συμβολοσειρές, προς τον συνολικό αριθμό q-grams που περιέχονται στις s και t. Ουσιαστικά ο αλγόριθμος στηρίζεται στο γεγονός ότι αν δύο συμβολοσειρές έχουν πολλά κοινά q-grams, θα έχουν και πολλά κοινά σύμβολα μεταξύ τους, άρα θα έχουν κοινή ρίζα και θα είναι πανομοιότυπες.

4.2.5 Αλγόριθμος Maedche-Staab

Αυτός ο αλγόριθμος [52] ουσιαστικά κανονικοποιεί την απόσταση Levenshtein κατά τον παρακάτω τρόπο:

$$\text{sim(s, t)} = \max(0, \frac{\min(|s|,|t|) - editDistance(s,t)}{\min(|s|,|t|)}) \quad (4.5)$$

Όπως παρατηρείται από τον τύπο, το μήκος της μικρότερης συμβολοσειράς παίζει βασικό ρόλο στον υπολογισμό της ομοιότητας, αφού η sim(s,t) θα είναι 0 αν η απόσταση των δύο συμβολοσειρών είναι μεγαλύτερη από αυτόν τον αριθμό.

4.2.6 Μετρική Jaccard

Στη μετρική Jaccard [37], όπως και στους υπόλοιπες τεχνικές που ακολουθούν, οι συμβολοσειρές εισόδου αντιμετωπίζονται σαν ένα σύνολο επιμέρους συμβολοσειρών-τμημάτων (λέξεων-tokens). Η διαφορά με τις παραπάνω μεθόδους είναι ότι σε αυτές τις περιπτώσεις τα επιμέρους τμήματα υπολογίζονται αν χωρίσουμε την εκάστοτε συμβολοσειρά με βάση κάποιο σύμβολο ή είδος συμβόλου όπως κεφαλαίους χαρακτήρες κ.λ.π. Έτσι, αν έχουμε την συμβολοσειρά ‘String-forTesting’, και χρησιμοποιήσουμε ως κριτήριο υπολογισμού των τμημάτων τους κεφαλαίους χαρακτήρες και τα μη αλφαβητικά και αριθμητικά σύμβολα, θα παραχθούν τρία τμήματα: String, for και Testing.

Η μετρική Jaccard λοιπόν, υπολογίζει την ομοιότητα ως τον αριθμό των tokens που υπάρχουν και στις δύο συμβολοσειρές, προς τον συνολικό αριθμό των tokens και των δύο συμβολοσειρών.

$$\text{Jaccard(s, t)} = \frac{|s \cap t|}{|s \cup t|} \quad (4.6)$$

4.2.7 Μετρική TF-IDF

Η μετρική *TF-IDF* ή μετρική του συνημίτονου (cosine similarity) χρησιμοποιεί τον παρακάτω τύπο για να υπολογίσει την ομοιότητα:

$$\text{TFIDF(s, t)} = \sum_{w \in s \cap t} V(w,s) \cdot V(w,t) \quad (4.7)$$

όπου

$$V(w, s) = \frac{V'(w,s)}{\sqrt{\sum_{w'} V'(w,s)^2}} \quad \text{και} \quad V'(w, s) = \log(TF_{w,s}+1)\cdot\log(IDF_w) \quad (4.8)$$

με $TF_{w,s}$ να αντιστοιχεί στην συχνότητα της λέξης w στην ακολουθία s και IDF_w είναι μια παράμετρος που υποδεικνύει το αντίστροφο του μέρους των ονομάτων στο σώμα κειμένου που περιέχει την λέξη w.

Μια άλλη μέθοδος, η οποία αποτελεί επέκταση της *TF-IDF* και ονομάζεται *soft TF-IDF,* χρησιμοποιεί τον παρακάτω τύπο για τον υπολογισμό της τελικής τιμής:

$$SoftTFIDF(s, t) = \sum_{w \in CLOSE(k,s,t)} V(w,s)\cdot V(w,t)\cdot D(w,t) \quad (4.9)$$

Σ' αυτή την έκδοση της τεχνικής χρησιμοποιείται μια συνάρτηση CLOSE(k,s,t) η οποία υποδηλώνει όλες τις λέξεις για τις οποίες όταν για $w \in s$ υπάρχει $u \in t$ τέτοιο ώστε dist(w,u)>k και για $w \in$ CLOSE(k,s,t) έχουμε $D(w,t)= \max_{u \in t} dist(w,u)$.

4.2.8 Μέθοδος Prefix (και Suffix)

Στη μέθοδο Prefix ελέγχεται αν η μικρότερη συμβολοσειρά περιέχεται στην αρχή της άλλης. Είναι όμοιες αν εμπεριέχεται στην αρχή της άλλης, και άνισες στην αντίθετη περίπτωση.

Μια άλλη μέθοδος παρόμοια με αυτήν είναι η Suffix, στην οποία γίνεται ο ίδιος έλεγχος αλλά με τα σύμβολα στο τέλος των συμβολοσειρών.

Επίσης υπάρχουν και διάφορες παραλλαγές τους, πιο ευέλικτες (δεν υπάρχουν μόνο οι τιμές ομοιότητας 0 και 1 αλλά και οι ενδιάμεσες τους). Σε αυτές ελέγχεται πόσα σύμβολα είναι κοινά και στις δύο συμβολοσειρές στην αρχή ή το τέλος τους αντίστοιχα (δηλαδή δεν ελέγχεται αν η μια περιέχει την άλλη μόνο, αλλά το πόσα σύμβολα είναι κοινά. Έτσι για παράδειγμα οι συμβολοσειρές 'asder' και 'asdefse' στην μέθοδο Prefix κρίνονται ανόμοιες με τιμή ομοιότητας 0, ενώ στις παραλλαγές τους κρίνονται όμοιες με μεγάλη τιμή αφού τα τέσσερα πρώτα γράμματα από την μια περιέχονται στη αρχή της άλλης). Η ομοιότητα τους υπολογίζεται με βάση τον αριθμό των κοινών συμβόλων καθώς και το συνολικό αριθμό συμβόλων.

4.2.9 Σύνοψη

Στην ενότητα 4.2 εξετάσαμε την λεξικογραφική ομοιότητα καθώς και τις βασικότερες τεχνικές της. Είδαμε πως για τον υπολογισμό της ομοιότητας δύο συμβολοσειρών οι διάφοροι αλγόριθμοι βασίζονται στα στοιχεία-σύμβολα που τις αποτελούν, είτε σαν σύμβολα μεμονωμένα, είτε σαν σύνολα συμβόλων (tokens, q-grams κ.λ.π). Παρόλα αυτά, η λεξικογραφική μέθοδος υπολογισμού της ομοιότητας ελλοχεύει και πολλούς κινδύνους-πιθανά λάθος ταιριάσματα, αφού δεν θέτει υπόψη της τη σημασιολογία των συμβολοσειρών. Έτσι για παράδειγμα, αν έχουμε σαν δύο συμβολοσειρές εισόδου τις ‘car’ και ‘automobile’ η λεξικογραφική ομοιότητα θα παράγει μικρή έως μηδενική ομοιότητα, αφού οι δύο λέξεις δεν περιέχουν κανένα κοινό γράμμα, παρόλο που είναι πανομοιότυπες. Και στη αντίθετη περίπτωση όμως, με συμβολοσειρές τις ‘dog’ και ‘god’ θα παράγουν μεγάλη ομοιότητα αφού οι δυο συμβολοσειρές περιέχουν τα ίδια ακριβώς γράμματα με άλλη σειρά. Παρόλα αυτά έχουν πολύ μικρή σχέση μεταξύ τους. Για αυτές τις αδυναμίες γίνεται προσπάθεια επίλυσης μέσω της σημασιολογικής προσέγγισης της ομοιότητας, η οποία εστιάζει στη σημασιολογική χροιά των συμβολοσειρών. Στην επόμενη ενότητα, παρουσιάζουμε την σημασιολογική ομοιότητα καθώς και τους κυριότερους αλγόριθμους που την εφαρμόζουν.

4.3 Σημασιολογική Ομοιότητα

Σαν σημασιολογική ομοιότητα θεωρούμε μια μετρική, η οποία υπολογίζει τα κοινά και διαφορετικά στοιχεία και γνωρίσματα για δύο οντότητες διαφορετικών οντολογιών. Για παράδειγμα, αν εξετάζουμε την έννοια {σκύλος}, τότε ως χαρακτηριστικό γνώρισμα της θα μπορούσε να θεωρηθεί το {έχει τέσσερα πόδια}. Με αυτή τη λογική, η έννοια σκύλος μοιάζει περισσότερο με την έννοια {γάτα} παρά με την έννοια {ουρανός}. Έτσι, θα μπορούσαμε να πούμε πως η σημασιολογική ομοιότητα παράγει μια τιμή που δηλώνει κατά πόσο σημασιολογικά μοιάζουν οι δύο οντότητες. Σαν σημασιολογική συσχέτιση ορίζουμε τη σχέση που μπορεί να έχουν δύο οντότητες χωρίς απαραίτητα να είναι και όμοιες. Για παράδειγμα, η έννοια {αυτοκίνητο} είναι περισσότερο συσχετισμένη με την έννοια {βενζίνη} παρά με την έννοια {ποδήλατο}, αφού μπορεί να συνδέονται μέσω μιας ιδιότητας μεταξύ τους. Παρόλα αυτά, η {αυτοκίνητο} έχει μεγαλύτερη ομοιότητα με την έννοια {ποδήλατο} αφού μπορεί να έχουν κάποιες ιδιότητες κοινές όπως οι {έχουν ρόδες}, {κινείται} κ.λ.π. Έτσι λοιπόν, η σημασιολογική ομοιότητα αναφέρεται σε αντικείμενα που έχουν κάποια ομοιότητα μεταξύ τους, ενώ η σημασιολογική συσχέτιση

μπορεί να υπάρξει και σε αντίθετες έννοιες. Θα μπορούσαμε να πούμε ότι η συσχέτιση είναι πιο γενική από την ομοιότητα. Παρόλα αυτά, μέσω της συσχέτισης μπορούμε να πάρουμε διάφορες πληροφορίες για την σημασιολογία των οντοτήτων και έτσι χρησιμοποιείται και αυτή στους παρακάτω αλγόριθμους. Στη συνέχεια θα παρουσιάζονται οι έννοιες σημασιολογική ομοιότητα και συσχέτιση ως ταυτόσημοι όροι.
Η σημασιολογική ομοιότητα αποτελεί έναν άλλο τρόπο για να υπολογιστεί η ομοιότητα μεταξύ δύο κλάσεων. Η διαφορά με τις λεξικολογικές μεθόδους ομοιότητας είναι ότι δεν στηρίζεται στα στοιχεία-σύμβολα που αποτελούν το όνομα των κλάσεων, αλλά επικεντρώνεται στο σημασιολογικό περιεχόμενό τους. Και εδώ έχουμε κάποιες κατηγορίες αλγορίθμων αναλόγως με την προσέγγιση που εφαρμόζεται για να βρεθεί η σημασιολογία.
Ορισμένοι αλγόριθμοι χρησιμοποιούν κάποιο λεξικό για να βρουν συσχετίσεις μεταξύ των οντολογιών. Υπάρχουν κάποια λεξικά, το πιο διαδεδομένο από τα οποία είναι το WordNet. Στο WordNet, οι έννοιες έχουν τη μορφή ταξινομίας και έτσι, μετρώντας τις αποστάσεις των λέξεων μπορεί να βρεθεί ένα πόρισμα όσον αφορά τις συσχετίσεις τους.
Άλλοι χρησιμοποιούν ένα σώμα κειμένου στο οποίο στηρίζονται για να υπολογίσουν κάποια στατιστικά δεδομένα για τις λέξεις που περιέχουν και να τα χρησιμοποιήσουν για την εύρεση ομοιοτήτων των κειμένων αυτών. Συνήθως τα κείμενα αυτά προέρχονται από τις ονομασίες, τις ιδιότητες και τα σχόλια των οντολογιών.

Τέλος, υπάρχουν κάποιοι οι οποίοι βασίζονται στη δομή των οντολογιών για τον υπολογισμό της σημασιολογίας. Ελέγχουν δηλαδή τη θέση που βρίσκεται μια οντότητα, σε σχέση με τις υπόλοιπες για να υπολογίσουν την ομοιότητα. Στηρίζονται στην συναίσθηση, ότι όσο πιο κάτω στη δομή της οντολογίας βρίσκεται μια οντότητα, τόσο πιο ειδική είναι, ενώ όσο πιο πάνω, τόσο πιο γενική είναι.

4.3.1 Αλγόριθμος Wu-Palmer

Οι Wu και Palmer [95] πρότειναν έναν αλγόριθμο εύρεσης σημασιολογικής ομοιότητας, ο οποίος βασίζεται και αυτός, στις αποστάσεις στο λεξικό. Ακόμα, χρησιμοποιείται ο *ελάχιστος κοινός γονέας (Least Common Subsumer – LCS).* Ως LCS θεωρούμε τον μεγαλύτερο, από άποψη βάθους, κόμβο που εμπεριέχει στους απογόνους του και τους δύο υπό εξέταση κόμβους. Θα μπορούσαμε να πούμε λοιπόν, ότι ο LCS είναι ο πρώτος

κοινός πρόγονος των δύο εννοιών στο γράφο του λεξικού. Το αποτέλεσμα του αλγορίθμου δίνεται από τον τύπο:

$$sim_{wup}(c1,c2)=\frac{2\cdot dist(c3,root)}{dist(c1,c3)+dist(c2,c3)+2\cdot dist(c3,root)} \quad (4.10)$$

όπου dist(x, y) είναι η απόσταση του x από τον y κόμβο μετρημένη σε κόμβους (και όχι ακμές). Root είναι η ρίζα του γράφου, c1 και c2 τα δύο υπό εξέταση synsets και c3 ο LCS τους. Έτσι, από τον τύπο, αν δύο έννοιες είναι πανομοιότυπες, ο θα ανήκουν στο ίδιο synset, ο LCS θα είναι αυτό το synsets, και οι τιμές dist(c1, c3) και dist(c2, c3) θα είναι ίσες με μηδέν και άρα η συνολική ομοιότητα θα είναι ίση με 1. Αν δεν έχουν καμία σχέση μεταξύ τους, τότε το LCS θα είναι η ρίζα του γράφου, άρα dist(c3, Root)=0, άρα η ομοιότητα θα είναι ίση με μηδέν.

Μια άλλη διατύπωση έγινε από τον Resnik και είναι η εξής:

$$sim_{rwup}(c1,c2)=\frac{2\cdot depth(c3)}{depth(c1)+depth(c2)} \quad (4.11)$$

όπου οι αποστάσεις είναι μετρημένες σε ακμές.

4.3.2 Αλγόριθμος Jiang-Conrath

Οι Jiang και Conrath πρότειναν ένα αλγόριθμο [41] στον οποίο γίνεται χρήση του περιεχομένου πληροφορίας όπως και παραπάνω. Η διαφορά με τον Resnik είναι πως γίνεται χρήση των περιεχομένων των δύο εννοιών και του κοινού τους γονέα, και όχι μόνο του γονέα. Έτσι, καταφέρνει να ξεπεράσει τα μειονεκτήματα που περιγράψαμε προηγουμένως. Η εξίσωση υπολογισμού της συσχέτισης είναι:

$$sim_{jcn}=\frac{1}{dist_{jcn}(c1,c2)} \quad (4.12)$$

όπου

$$dist_{jcn}(c1,c2)=IC(c1)+IC(c2)-2\cdot IC(LCS(c1,c2)) \quad (4.13)$$

Σε περιπτώσεις που η $dist_{jcn}(c1,c2)$ είναι ίση με 0 (όταν τα περιεχόμενα και των τριών εννοιών είναι ίσα, ή όταν το άθροισμα των περιεχομένων των δύο εννοιών ισούται με το διπλάσιο του περιεχομένου του κοινού γονέα), σημαίνει πως οι δύο έννοιες είναι πανομοιότυπες και έτσι θα πρέπει να επιστραφεί η τιμή 1.

4.3.3 Αλγόριθμος Lin

Μια άλλη προσέγγιση που χρησιμοποιεί το περιεχόμενο των εννοιών είναι αυτή που προτάθηκε από τον Lin [49] με τύπο:

$$\text{sim}_{\text{lin}}(c1,c2)=\frac{2\cdot IC(LCS(c1,c2))}{IC(c1)+IC(c2)} \quad (4.14)$$

Ο παρονομαστής γίνεται μηδέν σε περιπτώσεις που το περιεχόμενο των δύο εννοιών είναι μηδέν. Σε αυτή την περίπτωση θα επιστραφεί η κατάλληλη τιμή.

4.3.4 Συμπληρωματικοί αλγόριθμοι δομής

Οι αλγόριθμοι που ανήκουν σε αυτή τη κατηγορία χρησιμοποιούν τις τεχνικές που περιγράψαμε παραπάνω (σημασιολογικές ή λεξικογραφικές ή συνδυασμός τους) ως βάση, για να παραχθεί μια αρχική τιμή ομοιότητας μεταξύ δύο εννοιών. Έπειτα στηρίζονται στη δομή των οντολογιών (το τρόπο που είναι συνδεδεμένοι οι κόμβοι) για να αυξήσουν ή να μειώσουν την τιμή αυτή ώστε να καταλήξουμε σε μια ορθότερη τιμή ομοιότητας. Χρησιμοποιούνται δηλαδή συμπληρωματικά, εισάγοντας τη δομή στην ομοιότητα, ώστε να επικαλύψουν τα όποια μειονεκτήματα μπορεί να προκύψουν από τις παραπάνω τεχνικές. Σε αυτούς τους αλγόριθμους αντιμετωπίζεται η κάθε οντολογία ως ένας γράφος με κόμβους και ακμές.

Μια υποκατηγορία αυτών, είναι οι τεχνικές κατά τις οποίες ανάγεται η ομοιότητα δύο κόμβων σε άλλους εξαρτώμενους από αυτούς. Εδώ έχουμε:

Super (sub)-concepts rules: αυτή η τεχνική στηρίζεται στην ιδέα ότι όσο περισσότερο μοιάζουν οι άμεσοι πρόγονοι των κόμβων, τόσο θα μοιάζουν και οι κόμβοι μεταξύ τους. Στην ουσία, η ομοιότητα δύο κόμβων θα έχει ως αποτέλεσμα την αύξηση της ομοιότητας των άμεσων απογόνων τους. Αντίθετα, αν οι κόμβοι αυτοί δεν έχουν μεγάλη ομοιότητα, θα έχουμε μείωση της ομοιότητας των απογόνων. Η αύξηση/μείωση αυτή γίνεται είτε προσθέτοντας/αφαιρώντας μια σταθερή τιμή στην ήδη υπάρχουσα ομοιότητα, είτε προσθέτοντας/αφαιρώντας μια τιμή, η οποία είναι ποσοστό της τιμής ομοιότητας των προγόνων τους. Εδώ θεωρούμε άτυπα πως από τη στιγμή που οι δύο κόμβοι, πρόγονος και απόγονος, είναι συνδεδεμένοι, θα έχουν παρόμοια σημασιολογία.

Children: Στη παραπάνω λογική στηρίζεται και αυτή η τεχνική, από την αντίθετη πλευρά όμως. Η ομοιότητα δύο κόμβων θα επηρεάζει την ομοιότητα των προγόνων

τους κατά τρόπο παρόμοιο, όπως επηρεάζει τους απογόνους τους στην παραπάνω τεχνική. Ουσιαστικά αυτές οι δύο τεχνικές είναι αλληλοεπικαλυπτόμενες.

Leaves: Εδώ, η ομοιότητα των δύο εξεταζόμενων κόμβων εξαρτάται από τα φύλλα των υποδέντρων που σχηματίζονται με ρίζα τους κόμβους αυτούς. Σε αυτή τη περίπτωση θεωρούμε πως οι οντολογίες έχουν δημιουργηθεί με σημασιολογική διαβάθμιση από το γενικότερο στο ειδικότερο. Έτσι, τα φύλλα των αντιστοίχων υποδέντρων των κόμβων, θα έχουν πιο ειδικό σημασιολογικό περιεχόμενο (σε περίπτωση σημασιολογικής εξέτασης) αλλά και όνομα (σε περίπτωση λεξικογραφικής εξέτασης) και έτσι θα είναι πιο δύσκολο να ταιριάξουν λανθασμένα με έναν άλλο κόμβο. Και εδώ, η μεγάλη τιμή ομοιότητας των φύλλων συνεπάγεται αύξηση της τιμής ομοιότητας των εξεταζόμενων κόμβων, ενώ η μικρή ομοιότητα συνεπάγεται μείωση.

4.3.5 Σύνοψη

Σε αυτή τη παράγραφο (§ 4.3) ασχοληθήκαμε με την σημασιολογική ομοιότητα. Δώσαμε ένα ορισμό, καθώς και μια κατάταξη των αλγορίθμων αναλόγως με τη σκοπιά από την οποία εξετάζουν την ομοιότητα: με τη βοήθεια λεξικού, με την βοήθεια μιας τρίτης οντολογίας, με την χρήση στατιστικών επάνω σε ένα σώμα κειμένου, με τη χρήση της δομής της οντολογίας στην οποία ανήκουν οι έννοιες. Έπειτα, παρουσιάσαμε τους κυριότερους αλγόριθμους και τεχνικές.

Και στη σημασιολογική ομοιότητα έχουμε κάποια σοβαρά μειονεκτήματα, εκτός από τα μειονεκτήματα που μπορεί να έχει ο εκάστοτε αλγόριθμος. Κατά πρώτον, όλοι οι αλγόριθμοι εξετάζουν κάποια από τα χαρακτηριστικά των οντοτήτων, αλλά όχι όλα. Έτσι δεν έχουν ολοκληρωμένα δεδομένα για τον υπολογισμό της τελικής τιμής, αλλά εξετάζουν μονομερή στοιχεία.

Ακόμα, οι οντότητες που βρίσκονται κοντά σε μια άλλη οντότητα, θα έχουν παραπλήσιο σημασιολογικό περιεχόμενο. Έτσι, σε έναν έλεγχο μπορεί μια οντότητα της μιας οντολογίας να ταιριάξει με πολλές από την άλλη και συνήθως με μικρή διαφορά της τιμής ομοιότητας. Έτσι είναι δύσκολο να αποφανθούμε με ποια οντότητα από όλες θα πρέπει να ταιριάξει αν κάποια.

4.4 Επίλογος

Στο παρόν κεφάλαιο εξετάσαμε διάφορες τεχνικές λεξικογραφικής και σημασιολογικής ομοιότητας που χρησιμοποιούνται για την επίλυσης της ετερογένειας. Παρόλα αυτά, όπως φάνηκε και παραπάνω, όλες οι τεχνικές έχουν πλεονεκτήματα και μειονεκτήματα. Καταφέρνουν να αντιμετωπίσουν επιτυχώς ορισμένες περιπτώσεις ετερογένειας ενώ κάποιες άλλες όχι. Η κάθε προσέγγιση αντιμετωπίζει την ετερογένεια από διαφορετική σκοπιά δίχως να καταφέρνει να προσφέρει ικανοποιητικά αποτελέσματα σε όλες τις περιπτώσεις. Για να αντιμετωπιστεί αυτό το πρόβλημα, χρησιμοποιείται ο συνδυασμός διαφόρων τεχνικών είτε λεξικογραφικής, είτε σημασιολογικής ομοιότητας για να καλυφθεί ένα ευρύ φάσμα περιπτώσεων. Στο επόμενο κεφάλαιο εξετάζονται μια πληθώρα από αλγόριθμους που χρησιμοποιούν τις διάφορες τεχνικές (σημασιολογικές και λεξικογραφικές) που είδαμε στο κεφάλαιο αυτό, οι οποίες χρησιμοποιούνται σε διαφόρους συνδυασμούς για να παραχθούν έτσι ικανοποιητικότερα αποτελέσματα.

ΚΕΦΑΛΑΙΟ 5

ΤΕΧΝΙΚΕΣ ΤΑΙΡΙΑΣΜΑΤΟΣ ΟΝΤΟΛΟΓΙΩΝ

5.1 Γενικά

Στα προηγούμενα κεφάλαια παρουσιάσαμε το πρόβλημα της ετερογένειας στα πλαίσια της ηλεκτρονικής αγοράς προϊόντος. Για αυτό το πρόβλημα έχουν προταθεί διάφορα εργαλεία που εξετάζουν κάποια χαρακτηριστικά των πληροφοριών, όπως τα στοιχεία που αποτελούν τα ονόματα τους (λεξικογραφικός έλεγχος), τη σημασιολογία τους από πλευράς συνωνύμων (λεξικά), τη θέση τους στην οντολογία πηγής (σημασιολογικός έλεγχος). Οι αλγόριθμοι αυτοί μελετήθηκαν και παρουσιάστηκαν στο προηγούμενο κεφάλαιο. Παρόλα αυτά, οι έλεγχοι που κάνουν οι τεχνικές αυτές είναι μονομερείς, εξετάζουν λίγα χαρακτηριστικά από τις οντότητες, και έτσι καταφέρνουν να αντιμετωπίσουν ικανοποιητικά κάποιες περιπτώσεις ετερογένειας ενώ παράγουν μικρό ποσοστό επιτυχίας σε άλλες.

Ο συνδυασμός τεχνικών που εστιάζουν σε διάφορα σημεία του προβλήματος θα μπορούσε να αποφέρει πιο ικανοποιητικά αποτελέσματα. Με αυτό τον τρόπο, ένας αλγόριθμος ελέγχει πολλά χαρακτηριστικά των οντοτήτων ταυτόχρονα, και έτσι καταφέρνει να αντιμετωπίσει ικανοποιητικά περισσότερες περιπτώσεις ετερογένειας από ότι η οποιαδήποτε τεχνική μεμονωμένα. Ένα άλλο θετικό του συνδυασμού είναι ότι κάποια ενδεχόμενα λάθη των επιμέρους τεχνικών διορθώνονται μέσω των άλλων τεχνικών. Ένα ενδεχόμενο λάθος μιας τεχνικής, είναι πολύ δύσκολο να επαναληφθεί και από τις άλλες τεχνικές, αφού ελέγχουν διαφορετικά χαρακτηριστικά και έχουν διαφορετικό τρόπο λειτουργίας. Και επειδή στον υπολογισμό της τελικής ομοιότητας δύο οντοτήτων συμμετέχουν όλες οι τεχνικές, το λάθος αυτό επικαλύπτεται από τα αποτελέσματα των άλλων. Αν δηλαδή δύο έννοιες είναι πανομοιότυπες, και μια τεχνική παράγει μικρό ποσοστό ομοιότητας μεταξύ τους π.χ. 0.2, και οι άλλες υψηλό, π.χ. 0.9 με τον συνδυασμό των τιμών η συνολική τιμή θα είναι μεγαλύτερη από το 0.2 που είχε σαν αποτέλεσμα η μια τεχνική, άρα και σωστότερη.

Παρόλα αυτά, με το συνδυασμό τεχνικών, προκύπτουν κάποια προβλήματα. Κατά πρώτον, για το σενάριο που εξετάζουμε, με το ηλεκτρονικό εμπόριο, θα πρέπει όλες οι διαδικασίες να λειτουργούν σε πραγματικό χρόνο. Αυτό συνεπάγεται ότι δεν μπορούμε

να χρησιμοποιήσουμε πολλές τεχνικές για να ελέγξουμε την ομοιότητα. Όσο περισσότερες χρησιμοποιούμε, τόσο αυξάνεται ο χρόνος εκτέλεσης, και κατά συνέπεια τόσο πιο απαγορευτικός γίνεται ο συνδυασμός για το πρόβλημά μας. Θα πρέπει λοιπόν να γίνει μια σωστή επιλογή τεχνικών, που να είναι όσο το δυνατόν λιγότερες, γρήγορες και να καλύπτουν ένα μεγάλο ποσοστό ετερογένειας.

Ένα άλλο σημείο που πρέπει να προσεχθεί είναι ο τρόπος με τον οποίο θα γίνει ο συνδυασμός των τεχνικών. Όπως είπαμε, η κάθε τεχνική μπορεί να παράγει ικανοποιητικά αποτελέσματα σε ορισμένες περιπτώσεις και μη ικανοποιητικά σε αντίθετη περίπτωση. Έτσι, θα πρέπει να βρεθεί ένας τρόπος συνδυασμού των τεχνικών, ώστε σε κάθε περίπτωση ετερογένειας να μπορεί να αναδειχθεί η τεχνική που έχει τα ικανοποιητικότερα αποτελέσματα και να επηρεάσει την τελική τιμή ομοιότητας κατά τέτοιο τρόπο ώστε να καταλήξουμε σε ικανοποιητικότερο αποτέλεσμα.

Στη συνέχεια του κεφαλαίου παρουσιάζουμε τους σημαντικότερους αλγόριθμους συνδυασμού τεχνικών, καθώς και τους τρόπους με τους οποίους αντιμετωπίζουν τα παραπάνω προβλήματα. Ακόμα, παρουσιάζονται τα πλεονεκτήματα και τα μειονεκτήματα τους που θα μας βοηθήσουν να καταλήξουμε σε ένα νέο προτεινόμενο αλγόριθμο.

5.2 Αλγόριθμοι – Εργαλεία

Στη συνέχεια παρουσιάζουμε τον τρόπο λειτουργίας των αλγορίθμων που μελετήσαμε καθώς και κάποια βασικά χαρακτηριστικά τους.

5.2.1 Prompt και Anchor-Prompt

Ο Prompt [63] είναι ένας αλγόριθμος που χρησιμοποιεί λεξικογραφικές τεχνικές (Prefix, N-gram, Tokenization κ.λ.π.) για να βρει την ομοιότητα μεταξύ των στοιχείων και να προτείνει ορισμένα ταιριάσματα. Τις τεχνικές αυτές μπορεί να τις ορίσει ο χρήστης. Μετά, υπάρχει μια επικοινωνία με το χρήστη, όπου ο αλγόριθμος προτείνει κάποια ταιριάσματα, επιλέγει ή κάνει αλλαγές ο χρήστης επάνω σε αυτά, και ο αλγόριθμος κάνει εκ νέου προτάσεις για τις οποίες στηρίζεται στην δομή του δένδρου εκτός από τις λεξικογραφικές τεχνικές. Σε περιπτώσεις που πάνω από ένα στοιχείο της μιας οντολογίας, ταιριάζει με ένα στοιχείο της δεύτερης, εμφανίζει στο χρήστη όλους τους πιθανούς συνδυασμούς, και επαφίεται στην κρίση του για την διευθέτησή τους. Το μεγαλύτερο αρνητικό του είναι η έλλειψη αυτοματισμού, και εξάρτηση από το χρήστη, πράγμα σημαντικό για το σενάριο που εξετάζουμε.

Επέκταση του Prompt αποτελεί ο Anchor-Prompt [66]. Χρησιμοποιεί τον Prompt για να υπολογίσει κάποιες αρχικές τιμές ομοιότητας. Έπειτα, επιλέγονται κάποια από τα ζευγάρια που ορίστηκαν τα οποία θα χρησιμοποιήσει στη συνέχεια. Η επιλογή αυτή γίνεται συνήθως με βάση μια τιμή όριο (όσα ζευγάρια έχουν τιμή πάνω από την τιμή αυτή επιλέγονται, ενώ αντίθετα απορρίπτονται) ή με βάση την σειρά (π.χ. τα 10 πρώτα αποτελέσματα). Τα ζευγάρια αυτά ονομάζονται άγκυρες (anchors). Στη συνέχεια, υπολογίζονται όλα τα μονοπάτια σε κάθε οντολογία, που έχουν αρχή και τέλος οντότητες αυτών των ζευγαριών. Τέλος, για όλες τις κλάσεις που βρίσκονται στην ίδια θέση στο μονοπάτι αυτό, αυξάνει την τιμή ομοιότητας τους κατά μια σταθερά Χ. Για παράδειγμα, έστω ότι έχουν προκύψει τα μονοπάτια Δίκη – Βιβλιοθήκη - Άνθρωπος για το O_1 ,και δίκη –βιβλιοπωλείο - ανθρώπινος για το O_2 (όπου O_1 και O_2 είναι οι εξεταζόμενες οντολογίες). Τότε θα δημιουργηθεί ένα ζεύγος Βιβλιοθήκη – βιβλιοπωλείο αφού βρίσκονται στη δεύτερη θέση και οι 2 έννοιες, και θα αυξηθεί η ομοιότητα τους κατά μια σταθερά Χ π.χ. 0.2.

Ο αλγόριθμος στηρίζεται στη δομή των οντολογιών κυρίως. Πιο συγκεκριμένα, οι έννοιες που βρίσκονται στην ίδια θέση, κατά πάσα πιθανότητα θα ταιριάζουν μεταξύ τους. Παρόλα αυτά, δεν παράγει ακριβή αποτελέσματα σε μεγάλα μονοπάτια. Επίσης, δεν παράγει σωστά αποτελέσματα όταν η μια από τις οντολογίες έχει μεγάλο βάθος, με πολλές υπό-κλάσεις ενώ η άλλη οντολογία έχει μικρό βάθος.

5.2.2 OMEN (Ontology Mapping Enhancer)

Ο OMEN [59] παίρνει σαν είσοδο μια αρχική κατανομή πιθανοτήτων για ταιριάσματα, ένα θετικό και αρνητικό φράγμα, καθώς και ένα αριθμό Κ. Χρησιμοποιεί τις αρχικές πιθανότητες (a priori), για να υπολογίσει κάποιες τελικές (a posteriori). Πιο συγκεκριμένα, αν η πιθανότητα ενός ταιριάσματος υπερβαίνει το θετικό φράγμα, δημιουργεί ένα κόμβο εμπιστοσύνης, ο οποίος αντιπροσωπεύει το ταίριασμα αυτό. Επιπρόσθετα, δημιουργεί κόμβους που αντιπροσωπεύουν όλα τα πιθανά ταιριάσματα των υπολοίπων οντοτήτων. Στη συνέχεια δημιουργεί ακμές μεταξύ τους, με βάση την υλοποίηση κάτω-πάνω (bottom-up) ή πάνω-κάτω (top-down). Για περισσότερες λεπτομέρειες σχετικά με τις υλοποιήσεις αυτές, παραπέμπουμε τον αναγνώστη στην εκτεταμένη παρουσίαση του αλγορίθμου αυτού, δηλαδή στο [59]. Αφού απορρίψει τα ταιριάσματα τα οποία έχουν απόσταση από Κ και πάνω από κόμβο εμπιστοσύνης, χρησιμοποιεί το πλέγμα Bayes (Bayes Net) για να υπολογίσει τις a posteriori πιθανότητες. Αυτές υπολογίζονται αρχικά χρησιμοποιώντας τον αλγόριθμο ONION [58].

Έπειτα, χρησιμοποιεί μετα-κανόνες για να βελτιώσει τα αποτελέσματα του αλγορίθμου (για παράδειγμα, αν δύο έννοιες έχουν υψηλή ομοιότητα αυξάνει και την ομοιότητα των παιδιών τους). Από αυτές, θα κρατήσει όσες είναι πάνω από το δοσμένο φράγμα. Ο αλγόριθμος εκτελείται πολλαπλές φορές, έχοντας ως είσοδο τα καλύτερα ταιριάσματα του προηγούμενου κύκλου. Παρόλα αυτά, η χρήση μόνο λεξικογραφικής ομοιότητας (ONION) μπορεί να παράγει λανθασμένα ταιριάσματα, τα οποία θα θεωρηθούν ως κόμβοι εμπιστοσύνης στον επόμενο κύκλο και έτσι δεν μπορούν να διορθωθούν.

5.2.3 LSD (Learning Source Descriptions)

Ο LSD [24] χρησιμοποιεί ένα σύνολο από μηχανές εκμάθησης (learners) οι οποίοι παίρνουν ορισμένα ζεύγη από ταιριάσματα που έχουν ήδη γίνει (με την χρήση άλλων τεχνικών ή με χειροκίνητο ταίριασμα από το χρήστη) μεταξύ δύο οντολογιών και χρησιμοποιώντας τα, υπολογίζει επιπλέον ταιριάσματα. Τα αποτελέσματα αυτών συνδυάζονται μέσω του meta-learner με την μέθοδο stacking [93]. Πιο συγκεκριμένα οι learners του LSD είναι οι εξής:

- Ο *Whirl Learner* ο οποίος αντιστοιχεί ένα στιγμιότυπο βασιζόμενος στις ετικέτες των γειτονικών του κόμβων[18].
- Ο *Naïve Bayesian Learner* ο οποίος χρησιμοποιεί τη συχνότητα λέξεων που υπάρχουν για κάθε έννοια[25].
- Ο *Name Matcher* όπου αντιστοιχεί τα στοιχεία βασιζόμενος στην ομοιότητα των ονομάτων τους.
- Ο *Country-Name Recognizer* όπου βρίσκει αν ένα στιγμιότυπο είναι όνομα χώρας.
- Ο *Meta-Learner* ο οποίος συνδυάζει τα αποτελέσματα των υπολοίπων χρησιμοποιώντας την μέθοδο stacking.

Παρόλα αυτά, μπορεί να χρησιμοποιήσει οποιαδήποτε τεχνική η οποία παράγει ένα αποτέλεσμα της μορφής <Ψ, Φ, S1> για δύο οντολογίες Α και Β (όπου Ψ είναι στοιχείο της Α οντολογίας, Φ στοιχείο της Β και S1 είναι η τιμή ομοιότητας μεταξύ τους), αφού ο αλγόριθμος είναι ουσιαστικά ένας συνδυασμός άλλων αλγορίθμων. Στη παρούσα υλοποίηση χρησιμοποιούνται οι τεχνικές TF/IDF (στους Whirl Learner και Name Matcher), συχνότητα εμφάνισης λέξεων (Naïve Bayesian Learner), ομοιότητας λέξεων (Name Matcher).

Ο LSD παράγει μικρό ποσοστό επιτυχίας σε περιπτώσεις που έχουμε ασυμβατότητα στους τύπους των εννοιών, δηλαδή δεν υπάρχει καμία λεξικογραφική ομοιότητα. Ακόμη, αποτυγχάνει όταν υπάρχει μικρή, υποκειμενική, ή καθόλου διάκριση των στοιχείων.

5.2.4 GLUE

Ο αλγόριθμος GLUE [83] χωρίζεται σε τρία στάδια. Στο πρώτο πραγματοποιεί μια εκτίμηση κατανομών. Πιο συγκεκριμένα υπολογίζει τα $P(A,B)$, $P(A,\overline{B})$, $P(\overline{A},B)$, τα οποία είναι η πιθανότητα το στιγμιότυπο να ανήκει στην A έννοια και στην B, να ανήκει στην A αλλά όχι στην B, να ανήκει στην B αλλά όχι στην A αντίστοιχα. Ο υπολογισμός αυτός γίνεται με την βοήθεια της μηχανής εκμάθησης περιεχομένου (Content Learner). Στο επόμενο στάδιο, χρησιμοποιεί αυτές τις πιθανότητες, για να υπολογίσει το τελικό ταίριασμα βασιζόμενο στο συντελεστή Jaccard (βλέπε παράγραφο 4.2.5) καθώς και στη δομή της ταξονομίας. Στο τελευταίο επίπεδο, παίρνει τα αποτελέσματα του προηγούμενου επιπέδου και χρησιμοποιεί διαφόρους περιορισμούς και ευριστική/κωδικοποιημένη γνώση για να παράγει την καλύτερη χαρτογράφηση. Παίρνει σαν είσοδο τις 2 οντολογίες O_1 και O_2 σε μορφή ταξονομίας, με αντίστοιχα στιγμιότυπα U_1 και U_2.

Σε κάθε επίπεδο του ο αλγόριθμος κάνει χρήση και διαφορετική τεχνική. Έτσι ο Content Learner στο πρώτο βήμα χρησιμοποιεί τη συχνότητα των λέξεων για να υπολογίσει ομοιότητα. Στο δεύτερο βήμα, υπολογίζει το συντελεστή Jaccard, που βασίζεται σε πιθανότητες ή το «Πιο-Συγκεκριμένος-Γονιός» (most-specic-parent) μέτρο που βασίζεται στην ιεραρχία για να υπολογίσει ένα πίνακα ομοιότητας. Το τελευταίο επίπεδο, βασίζεται στο σκεπτικό ότι η ετικέτα ενός κόμβου, εξαρτάται από τους γειτονικούς κόμβους. Έτσι, χρησιμοποιεί τεχνικές που βασίζονται στην δομή του γράφου.

Ο Content Learner δεν λειτουργεί σωστά για έννοιες με μικρές σε μέγεθος ή αριθμητικές τιμές ενώ παράγει ικανοποιητικά αποτελέσματα όταν τα στοιχεία περιέχουν μεγάλες περιγραφές ή είναι αυστηρά διακριτές. Σε περιπτώσεις που υπάρχουν πολλαπλά επικαλυπτόμενα αποτελέσματα, χρησιμοποιείται το τελευταίο στάδιο για να ξεκαθαρίσει το αποτέλεσμα που ταιριάζει περισσότερο. Τέλος ο GLUE παράγει λανθασμένα αποτελέσματα σε περιπτώσεις που δεν υπάρχουν ταιριάσματα μεταξύ των εννοιών-κόμβων αφού παράγει ταιριάσματα για όλους τους κόμβους άρα και σ' αυτούς.

5.2.5 S-Match

Ο αλγόριθμος S-Match [30] έχει ως είσοδο 2 «γράφους» των Οντολογιών (XML). Αρχικά υπολογίζει για κάθε κόμβο των 2 οντολογιών την $C_{L.}$ Η C_L είναι ένας αριθμός λέξεων που αντιστοιχούν στον εκάστοτε κόμβο, ενωμένες με διάφορες πράξεις όπως ένωση, τομή κ.λ.π. Για να πάρει αυτές τις λέξεις περνάει από ένα αριθμό σταδίων: Χωρισμό στις βασικές συμβολοσειρές-tokens, υπολογισμό των βασικών μορφών κάθε λέξης (Lemmatization), υπολογισμός των συνώνυμων της λέξεων (μέσω WordNet) και δημιουργία σύνθετων προτάσεων (μετασχηματίζουμε όλες τις συνδετικές λέξεις σε προτασιακά σύμβολα π.χ. το 'και' μεταφράζεται σαν ένωση).

Στη συνέχεια, υπολογίζεται η C_N (η τομή των εννοιών-ονομάτων από την ρίζα μέχρι τον κόμβο που εξετάζουμε) για κάθε κόμβο. Τέλος, παίρνει για κάθε ζεύγος από κόμβους των O_1 και O_2 οντολογιών και υπολογίζει τις μεταξύ τους σχέσεις σύμφωνα με τα C_L και $C_{N.}$ Ο υπολογισμός των C_L σχέσεων γίνεται με την βοήθεια του WordNet, ενώ των C_N με ένα επιλυτή SAT (SAT Solver), ο οποίος χρησιμοποιείται για επίλυση δυαδικών εξισώσεων.

Το κυριότερο αρνητικό του αλγορίθμου είναι ότι στη έξοδο επιστρέφει όλα τα ζευγάρια, δίχως να αποφασίζει για τα πιθανότερα ταιριάσματα. Έτσι θα πρέπει ο χρήστης να αποφασίσει ποια θα είναι αυτά. Ακόμα, τα αποτελέσματα είναι με μορφή σχέσεων, και όχι πιθανοτήτων ώστε να θέσουμε κάποιο όριο. Για το σενάριο μας θα πρέπει ο αλγόριθμος να αποφασίζει τα σωστά ταιριάσματα αυτόματα, ώστε να περάσει μετά στη διαδικασία αντιγραφής των τιμών από τη μια οντολογία στην άλλη.

5.2.6 Falcon –AO

Το εργαλείο Falcon-AO [40] εμπεριέχει 2 τεχνικές/ταιριαστές (matchers) για να υπολογίσει το ταίριασμα μεταξύ των 2 οντολογιών της εισόδου. Ο πρώτος, ο LMO χρησιμοποιεί την απόσταση των συμβολοσειρών (λεξικογραφική ομοιότητα) για να υπολογίσει την τιμή της εξίσωσης:

$$SS = 1 / e^{\frac{ed}{s1.len+s2.len-ed}} \quad (5.1)$$

Επίσης, παράγει κάποια σύνολα συμβολοσειρών για κάθε κόμβο (με τη βοήθεια του WordNet από το οποίο εξάγει τα συνώνυμα), στα οποία μετράει στατιστικά (TF / IDF) για να υπολογίσει την ομοιότητα. Οι δύο αυτές τιμές συνδυάζονται με άθροισμα, αφού τεθούν τα αντίστοιχα βάρη.

Ο δεύτερος Matcher μετατρέπει τις οντολογίες σε διμερής γράφους και αποκλείει κάποια ταιριάσματα με βάση τα αποτελέσματα του A, καθώς και άλλων κανόνων. Στην συνέχεια δημιουργεί ένα πίνακα για κάθε οντολογία, τον πίνακα αντιπροσώπευσης, με βάση τον

γράφο. Τέλος κάνει κάποιες επαναλήψεις όπου ανανεώνει συνέχεια κάποιες εξισώσεις σχετικά με τους πίνακες, και στο τέλος συγκρίνει τους πίνακες για να βρεθούν 1-1 ομοιότητες. Λεπτομέρειες σχετικά με τις εξισώσεις και τη διαδικασία του GMO υπάρχουν στο [40].

Ο GMO Matcher πάντα προσπαθεί να παράγει όλα τα πιθανά ζεύγη εννοιών. Για να αξιολογηθούν τα αποτελέσματα και των δύο ταιριαστών, το Falcon-AO αυτά χρησιμοποιεί τις τιμές λεξικολογική συγκρισιμότητα (Linguistic Comparability - LC) και δομική συγκρισιμότητα (Structural Comparability - SC). Η LC ορίζεται ως:

$$LC = \frac{M}{\sqrt{N_{O1} * N_{O2}}} \qquad (5.2)$$

όπου M είναι ο αριθμός των ζευγαριών που είναι πάνω από μια σταθερά X (η σταθερά αυτή προσδιορίζεται εμπειρικά). N_{O1} και N_{O2} αποτελούν τον αριθμό οντοτήτων στις οντολογίες O_1 και O_2 αντίστοιχα.

Η SC ορίζεται ως:

$$SC = \frac{V_1 * V_2}{\|V_1\| * \|V_2\|} = \frac{\sum_{j=1}^{n} u_{1j} \quad u_{2j}}{\sqrt{\sum_{j=1}^{n} u_{1j} \quad u_{1j}} \sqrt{\sum_{j=1}^{n} u_{2j} \quad u_{2j}}} \qquad (5.3)$$

όπου V_1, V_2 τα διανύσματα που αντιπροσωπεύουν την συχνότητα των ενσωματωμένων ιδιοτήτων στις O_1 και O_2 αντίστοιχα (που είναι τα RDF [33], RDFS [12] and OWL [71] λεξικά, χρησιμοποιούμενα ως ιδιότητες σε τριπλέτες π.χ. rdf:type, rdfs:subClassOf και owl:onProperty). Το u_{ij} αποτελεί τον αριθμό εμφανίσεων της ιδιότητας p_j στην O_i οντολογία.

Παρόλαυτα, ο αλγόριθμος δεν περιέχει καμία λειτουργία, για να απορρίπτει κάποια ζεύγη ή να στηρίζεται και σε άλλα στοιχεία των οντολογιών ώστε να μειωθούν τα λανθασμένα και τα διπλό-ταιριάσματα.

5.2.7 NOM (Naïve Ontology Matching)

Ο NOM [26] χρησιμοποιεί διάφορα μέτρα σύγκρισης (κανόνες), στα οποία θέτει ένα συγκεκριμένο βάρος και παίρνει το άθροισμά τους, το οποίο χαρακτηρίζει το ταίριασμα των 2 εννοιών που συγκρίνει εκείνη τη στιγμή. Πιο συγκεκριμένα, τα μέτρα αυτά είναι:

1. Σύγκριση μεταξύ ετικετών (λεξικογράφηση π.χ. με edit Distance)
2. Σύγκριση μεταξύ προσδιοριστών.
3. Σύγκριση μεταξύ ιδιοτήτων κλάσεων
4. Αρχή (domain) και τέλος (range) ιδιοτήτων
5. Αν οι γονείς 2 κλάσεων ταιριάζουν, θα ταιριάζουν και οι ίδιες οι κλάσεις.

6. Αν τα παιδιά 2 κλάσεων ταιριάζουν, θα ταιριάζουν και οι ίδιες οι κλάσεις.
7. Σύγκριση στιγμιοτύπων κλάσεων
8. Σύγκριση κλάσεων στις οποίες ανήκουν δύο στιγμιότυπα.
9. Αν δύο κλάσεις έχουν παρόμοιο ποσοστό στιγμιότυπων θα είναι παρόμοιες.
10. Αν 2 στιγμιότυπα ενώνονται μ' ένα άλλο μέσω της ίδιας ιδιότητας, τότε τα 2 αρχικά στιγμιότυπα είναι παρόμοια.
11. Αν 2 ιδιότητες ενώνουν τα ίδια 2 στιγμιότυπα, τότε οι ιδιότητες αυτές θα ταιριάζουν.
12. Κάποιες συγκεκριμένες ιδιότητες (π.χ. same AS) θα έχουν παρόμοιο domain με range.
13. Αν ο κώδικας hash (ένας μοναδικός αριθμός που προσδιορίζει τις οντότητες σε ένα πρόγραμμα) δύο αρχείων ίσος, θα είναι ίσα και τα αρχεία.
14. Ίδιο MIME (τύπος) μεταξύ αρχείων, συνεπάγεται μεγάλη ομοιότητα των αρχείων.

Τα παραπάνω βήματα εκτελούνται πολλαπλές φορές (είτε ένα συγκεκριμένο αριθμό φορών, είτε μέχρι ο αριθμός των αλλαγών ανά επανάληψη πέσει κάτω από ένα δοσμένο κατώφλι). Στο τέλος, αφαιρεί τα χαμηλά και τα διπλά ταιριάσματα, εμφανίζοντας την πιο πιθανή χαρτογράφηση. Αυτό πετυχαίνεται με τη χρήση ενός ορίου, όπου πάνω από αυτό τα αποτελέσματα γίνονται δεκτά. Το όριο αυτό υπολογίζεται με ένα αριθμό δοσμένο από το χρήστη. Ο αριθμός αυτός μπορεί να χρησιμοποιηθεί αμέσως σαν όριο. Ακόμα, μπορεί να αφαιρεθεί από τη μεγαλύτερη τιμή ομοιότητας που προέκυψε και το αποτέλεσμα να θεωρηθεί σαν όριο. Τέλος, αυτός ο αριθμός μπορεί να θεωρηθεί σαν ποσοστό το οποίο αφαιρείται και πάλι από τη μεγαλύτερη τιμή για να υπολογιστεί το όριο.

5.2.8 Similarity Flooding (SF)

Ο αλγόριθμος αυτός [56], χρησιμοποιεί σύγκριση prefix για να δημιουργήσει μια αρχική χαρτογράφηση. Μετά, χρησιμοποιεί τον αλγόριθμο SFJoin ο οποίος δημιουργεί πρώτα ένα γράφο όπου ο κάθε κόμβος του αντιστοιχεί σε ένα ζεύγος κόμβων των 2 γράφων στηριζόμενη τις σχέσεις των αρχικών κόμβων. Ύστερα ο SFJoin αναθέτει κάποιες πιθανότητες σε κάθε ακμή του γράφου ανάλογα με τις σχέσεις των κόμβων που το αποτελούν. Τέλος υπολογίζει με πολλαπλές εκτελέσεις αυτού του βήματος κάποιες τιμές που αντιστοιχούν σε κάθε ζεύγους. Οι τιμές αυτές, προέρχονται από την ομοιότητα των γειτονικών κόμβων τους. Στηρίζεται στην παρατήρηση, ότι αν δύο στοιχεία των οντολογιών είναι πανομοιότυπα, τότε και η ομοιότητα των γειτονικών τους στοιχείων θα

αυξάνεται. Έτσι, με την επαναληπτική διαδικασία, η αρχική ομοιότητα δύο στοιχείων, ενσωματώνεται, επηρεάζει την ομοιότητα των υπολοίπων γειτονικών του στοιχείων. Ο SF συνεχίζει αποκόπτοντας κάποια από τα αποτελέσματα που προέκυψαν από τον SFJoin με βάση κάποιους περιορισμούς και τέλος, διαλέγει ένα πιθανό αποτέλεσμα με το τελεστή Select Threshold για να το παρουσιάσει στο χρήστη.
Ο αλγόριθμος αυτός παράγει ένα πίνακα με όλα τα ταιριάσματα και τις πιθανότητές τους. Για να περιορίσει το χώρο των αποτελεσμάτων αυτών χρησιμοποιεί κάποιους περιορισμούς (ανάλογα με το πρόβλημα) όπως π.χ. έλεγχο συμβατότητας τύπων κ.λ.π. Έτσι, απορρίπτει διάφορα λανθασμένα ταιριάσματα ή ταιριάσματα που επικαλύπτουν το ένα το άλλο. Τέλος κάνει χρήση του τελεστή Select Threshold στα αποτελέσματα των περιορισμών για να επιλέξει την πιθανότερη λύση και να τις παρουσιάσει στο χρήστη.

5.2.9 CROSI (CMS)

Παίρνει σαν είσοδο τα ονόματα δύο εννοιών και υπολογίζει την απόσταση μεταξύ τους. Ο υπολογισμός αυτός γίνεται χρησιμοποιώντας την τεχνική edit distance στην πιο απλή μορφή της, είτε υπολογίζοντας ορισμένες πολυπλοκότερες υβριδικές συναρτήσεις απόστασης. Χρησιμοποιεί μια πληθώρα από matchers, ανεξάρτητους μεταξύ τους, που συνδυαζόμενοι αποτελούν τον Crosi [42]. Ο συνδυασμός των matchers γίνεται με ρυθμίσεις από τον χρήστη.
Οι matchers που περιέχει χωρίζονται σε 2 κατηγορίες: τους ονοματικούς (Name Matchers) και τους σημασιολογικούς (Semantic Matchers).
Οι *Name Matchers* ποικίλουν από καθαρά συντακτικοί μέχρι πιο σημασιολογικοί. Χρησιμοποιούν τεχνικές που βασίζονται στο όνομα των εννοιών και λειτουργούν σε λεκτικό επίπεδο (π.χ. με την χρήση του WordNet) και σε επίπεδο πρότασης-φράσης (π.χ. αλλαγή όλων των φράσεων στην ενεργητική τους μορφή). Υπολογίζουν μια τιμή ομοιότητας για κάθε ζεύγος a, b οντοτήτων των οντολογιών, την simlocal(a,b).
Οι *Semantic Matchers* χρησιμοποιούν τα αποτελέσματα των Name Matchers - simlocal(a,b) – και χωρίζονται σε 2 κατηγορίες: αυτούς που αξιοποιούν την δομή της οντολογίας (Structure-aware) και αυτούς που χρησιμοποιούν τον ορισμό της έννοιας (intension-aware).

- Οι *Structure-aware* υπολογίζουν την ομοιότητα ως εξής: Sim (a, b)=c*simlocal(a,b) + d* sim(a_i, b_i) όπου c και d είναι τα βάρη, και a_i, b_i η ομοιότητα των αντίστοιχων γονέων των εννοιών που ελέγχονται.

- Οι *intension-aware* ελέγχουν τα ονόματα, το πεδίο ορισμού τους και το πεδίο τιμών των ιδιοτήτων των εννοιών και αναλόγως αυξάνουν ή μειώνουν την simlocal(a,b).

Για να ξεπεραστεί ο σκόπελος των διαφορετικών εννοιών με παρόμοια ονομασία, ή ίδιων εννοιών με διαφορετικές ονομασίες χρησιμοποιεί τα NLP (Natural Language Processing) πακέτα που ειπώθηκαν και πιο πάνω (Name και Semantic Matchers). ¨Ένα αρνητικό του αλγόριθμου αυτού είναι ότι ο συνδυασμός των matchers γίνεται από τον χρήστη. Έτσι, όμως, αφού σε κάθε ταίριασμα αλλάζουν οι οντολογίες, θα αλλάζουν και τα βάρη αφού κάθε matcher παράγει καλά αποτελέσματα σε ορισμένες περιπτώσεις (και πρέπει να έχει μεγάλο βάρος σε αυτές) ενώ σε άλλες περιπτώσεις έχει χαμηλό ποσοστό επιτυχίας (και πρέπει να του δοθεί μικρό βάρος) και θα χρειαστούν πολλές δοκιμές από τους χρήστες για να βρεθούν τα βέλτιστα βάρη για κάθε περίπτωση. Ακόμη με αυτό το τρόπο έχουμε έλλειψη αυτοματισμού. Ένας τρόπος να λυθεί αυτό είναι να χρησιμοποιηθούν τεχνικές εκμάθησης μηχανής ή σταθερά βάρη τα οποία έχουν προκύψει μέσα από πειραματισμό. Τέλος ο αλγόριθμος περιλαμβάνει και την θέση των εννοιών στην ιεραρχία της οντολογίας και με αυτό το τρόπο μειώνει κατά πολύ τα λανθασμένα θετικά ταιριάσματα.

5.2.10 Ctx – Match (Context Match)

Ο Context Match [54] όπως υπαινίσσεται και το όνομά του είναι ένας αλγόριθμος ο οποίος βασίζεται αποκλειστικά στο περιεχόμενο-νόημα των λέξεων που χρησιμοποιήθηκαν για την δημιουργία των κόμβων. Υπολογίζει αρχικά μια λογική πρόταση βασιζόμενος στο όνομα των κόμβων την οποία την εμπλουτίζει με το νόημα των προγόνων αλλά και των συνωνύμων του από το WordNet. Πιο συγκεκριμένα, χρησιμοποιεί το Alembic [19] για να αντιστοιχήσει κάθε λέξη σε ένα μέρος του λόγου, και να δημιουργήσει μια λογική πρόταση με βάση τα μέρη του λόγου (για παράδειγμα το και ερμηνεύεται σαν ένωση, το κόμμα σαν τομή κ.λ.π.). Έπειτα χρησιμοποιείται το WordNet, για την εξαγωγή διαφόρων συνωνύμων με τις λέξεις του προηγούμενου βήματος, τα οποία προσάπτονται σ’ αυτές. Στη συνέχεια, δημιουργούνται διάφορες περιγραφικές προτάσεις για τους προγόνους των κόμβων που εξετάζονται, οι οποίες συνδυάζονται με αυτές που έχουν ήδη δημιουργηθεί για τον κόμβο. Τέλος χρησιμοποιεί τεχνικές συμπερασμού μέσω ενός επιλυτή SAT για να υπολογίσει τις σχέσεις μεταξύ των κόμβων.

Χρησιμοποιεί τεχνικές που βασίζονται στη δομή των οντολογιών (για να υπολογίσει μέσω των γονιών το περιεχόμενο του κάθε κόμβου) καθώς και τεχνικές συμπερασμού για να υπολογίσει μέσω των λογικών προτάσεων τις σχέσεις μεταξύ των κόμβων. Επιστρέφει μια εξίσωση για κάθε κόμβο διατυπωμένη σε περιγραφική λογική, και τελικώς, υπολογίζει μια χαρτογράφηση μεταξύ όλων των ζευγών κόμβων, η οποία για κάθε ζεύγος περιέχει την σχέση μεταξύ των εννοιών δηλαδή ισότητα, πιο γενικό, πιο ειδικό, ανεξάρτητο.

Ένα αρνητικό του αλγορίθμου αυτού είναι ότι χρειάζεται έγγραφα κειμένου και έτσι δεν μπορεί να χρησιμοποιηθεί όταν δεν υπάρχουν τέτοια (π.χ. έχουμε εικόνες). Παρόλα αυτά έχει ικανοποιητικά αποτελέσματα με ουσιαστικά που αντιστοιχούν σε αντικείμενα, αλλά όχι με αυτά που αντιστοιχούν σε αφηρημένα αντικείμενα. ¨Ένα άλλο αρνητικό του αλγορίθμου είναι ότι η χρήση του Alembic δεν επιτρέπει την επίλυση του συντονισμού των ασαφειών στις ονοματικές ενώσεις, αλλά ούτε υπάρχει κανένας αλγόριθμος που να διευθετεί αυτό το πρόβλημα.

5.2.11 Cupid

Ο Cupid [51] χρησιμοποιεί και αυτός ένα συνδυασμό από διαφορετικές τεχνικές ταιριάσματος. Αρχικά εκτελείται το γλωσσικό ταίριασμα. Εδώ, υπάρχει ένα στάδιο προετοιμασίας όπου γίνεται χωρισμός των ονομάτων σε tokens, διαγραφή λέξεων χωρίς περιεχόμενο (κυρίως συνδετικές λέξεις όπως ‘και’ ή άρθρα) και επέκταση ακρωνύμιων. Έπειτα, γίνεται μια κατηγοριοποίηση των στοιχείων της οντολογίας αναλόγως με τους τύπους τους. Τέλος, γίνεται σύγκριση των συμβολοσειρών που προέκυψαν με τη χρήση ενός λεξικού, όπως το WordNet. Στο γλωσσικό ταίριασμα γίνεται δηλαδή σημασιολογικός έλεγχος ομοιότητας, όπου ελέγχονται τα στοιχεία ξεχωριστά ένα προς ένα.

Στο επόμενο βήμα του αλγορίθμου, χρησιμοποιείται μια ρουτίνα, η οποία δημιουργεί ένα σημασιολογικό δέντρο από τον γράφο της οντολογίας. Η ομοιότητα των κόμβων υπολογίζεται ως το ποσοστό των φύλλων των υποδέντρων τους που ταιριάζουν. Η ομοιότητα των φύλλων θα είναι η ομοιότητα που υπολογίστηκε παραπάνω. Ακόμα, η ομοιότητα των κόμβων και των φύλλων τους είναι αναδρομική: Αν δύο κόμβοι παρουσιάσουν υψηλή τιμή ομοιότητας, θα αυξήσει και την τιμή των φύλλων τους. Στην αντίθετη περίπτωση θα μειωθεί. Μέσα από αυτόν τον έλεγχο αυξάνεται η ευελιξία του αλγορίθμου σε διαφορετική διάταξη των στοιχείων αφού εξαρτάται από τα φύλλα παρά από την άμεση δομή.

Ο αλγόριθμος, μπορεί να ξεπεράσει ορισμένα προβλήματα που αφορούν την ονομασία των στοιχείων λόγω της κανονικοποίησης που γίνεται στο όνομα τους (tokenization, elimination, expansion). Παρόλα αυτά, αποτυγχάνει σε περίπτωση που υπάρχουν κύκλοι περιεκτικότητας και σχέσεων *παράγεται-από*.

5.2.12 COMA

Ο COMA [22] αλγόριθμος αποτελεί ένα συνδυασμό από επιμέρους τεχνικές οι οποίοι χωρίζονται σε 3 κατηγορίες: απλές, υβριδικές και προσανατολισμένες στην επαναχρησιμοποίηση.

Οι απλές περιλαμβάνουν τις Affix (ψάχνει για κοινά συνθετικά των λέξεων), n-gram, Edit Distance, Soundex (υπολογισμός φωνητικής-ακουστικής ομοιότητας), Synonym (με συνώνυμα με την χρήση λεξικών) και DataType (χρησιμοποιεί ένα πίνακα διαθέσιμων τύπων στους οποίους ταιριάζει πρώτα τους τύπους των στοιχείων των οντολογιών και μετά συγκρίνει).

Οι υβριδικές περιλαμβάνουν τις Name (εφαρμόζει πρώτα το χωρισμό σε tokens και expansion ενώ μετά εφαρμόζει τις Affix, Synonym, 3-gram και συνδυάζει τα αποτελέσματα), NamePath (αλλάζει τα ονόματα των κόμβων σε αυτό του μονοπατιού τους από τη ρίζα στον εκάστοτε κόμβο και εκτελεί την Name), TypeName (συνδυάζει την DataType με την Name), Children (αποφασίζει την ομοιότητα των φύλλων με την TypeName ενώ τους ενδιάμεσους σύμφωνα με την ομοιότητα των παιδιών τους και αυτά αναδρομικά με α δικά τους παιδιά κ.λ.π μέχρι τα φύλλα) και Leaves (αποφασίζει την ομοιότητα των ενδιάμεσων κόμβων ελέγχοντας την ομοιότητα των φύλλων των υποδέντρων των κόμβων αυτών. Και εδώ για σύγκριση των φύλλων χρησιμοποιείται η TypeName).

Τέλος, στις τεχνικές προσανατολισμένες στην επαναχρησιμοποίηση ανήκει ο αλγόριθμος MatchCompose, ο οποίος δημιουργεί ένα καινούριο ταίριασμα από ήδη υπάρχοντα. Αυτός στηρίζεται στην ιδέα ότι αν ένας κόμβος α αντιστοιχεί σε ένα β, και ο β αντιστοιχεί με την σειρά του σε ένα γ, τότε και ο α θα αντιστοιχεί στον γ. Έτσι παίρνει το μέσο όρο των πιθανοτήτων των επιμέρους ταιριασμάτων ((έστω P(α↔β)=0.7 και P(β↔γ)=0.5) για να υπολογίσει και την πιθανότητα του νέου ταιριάσματος(P(α↔γ)=(0.7+0.5)/2=0.6). Επίσης περιέχει κάποιους αλγορίθμους οι οποίοι χρησιμοποιούν τον MatchCompose, όπως ο Schema (που ελέγχει όλη την οντολογία και αποθηκεύει τα ταιριάσματα για μελλοντική χρήση) και ο Fragments ο οποίος αναφέρεται σε κομμάτια από οντολογίες και όχι σε ολόκληρες όπως ο Schema.

Ο συνδυασμός των αποτελεσμάτων μπορεί να γίνει με 4 τρόπους:

- Max (επιλέγει την μεγαλύτερη τιμή που έχει προκύψει από τους matchers).
- Weighted (υπολογίζει το άθροισμα των αποτελεσμάτων του κάθε matcher πολλαπλασιασμένο επί ένα αριθμό-βάρος από 0 ως 1 που αντιστοιχεί στο πόσο σημαντικός και αξιόπιστος είναι ο matcher).
- Average (είναι μια υποπερίπτωση του Weighted όπου τα αποτελέσματα θεωρούνται όλα ισότιμα-έχουν το ίδιο βάρος).
- Min (επιλέγει την μικρότερη τιμή που έχει προκύψει από τους matchers).
- Dice (πηλίκο με αριθμητή τον αριθμό των στοιχείων που μπορούν να ταιριάξουν προς το συνολικό αριθμό των στοιχείων)

Ο αλγόριθμος αυτός έχει μια πληθώρα τεχνικές για να αποφεύγει τα διπλό-ταιριάσματα και τα λανθασμένα ταιριάσματα. Μια από αυτές είναι η χρήση ενός κατωφλίου, όπου επιλέγονται τα στοιχεία που έχουν πιθανότητα πάνω από την τιμή αυτή. Άλλη είναι η επιλογή των N καλύτερων αποτελεσμάτων (όπου N δοσμένο από το χρήστη). Τέλος μια άλλη τεχνική αποτελεί η MaxDelta, όπου επιλέγονται τα στοιχεία με την μεγαλύτερη πιθανότητα η οποία διαφέρει από την μέγιστη κατά ένα παράγοντα δέλτα.
Γενικά, παρόλο που ο κάθε matcher μεμονωμένα μπορεί να παράγει λανθασμένα αποτελέσματα ο ταυτόχρονος συνδυασμός τους οδηγεί σε μια πιο σταθερή και ακριβής συμπεριφορά, με μεγαλύτερη αποτελεσματικότητα στον Schema.

5.2.13 COMA++

Το COMMA++ [8] αποτελεί μια προέκταση του αρχικού αλγορίθμου, COMMA. Περιέχει την MatchCompose προσέγγιση του προκάτοχο του, αλλά και την επεκτείνει χρησιμοποιώντας διαμοιραζόμενες ταξονομίες, νέους ταιριαστές, καθώς και έναν αλγόριθμο για να αποδόμηση μεγάλων προβλημάτων ταιριάσματος σε μικρότερα, ώστε να τα λύνει πιο εύκολα. Ο αλγόριθμος αυτός (fragment-based matching) την τακτική του διαίρει και βασίλευε, ώστε να μειώσει το μέγεθος του προβλήματος και έτσι να πετύχουμε καλύτερη ποιότητα στα αποτελέσματα καθώς και μείωση του χρόνου εκτέλεσης. Πιο συγκεκριμένα έχουμε:

Ταίριασμα βασισμένο σε τεμάχια: Εδώ, περιέχονται δύο κύριες φάσεις.

- *Προσδιορισμός των παρόμοιων τεμαχίων.* Ανάλογα με τους τύπους των τεμαχίων, βρίσκει τα τεμάχια από τους εισαγόμενους γράφους-οντολογίες και τα συγκρίνει για να προσδιορίσει τα παρόμοια τα οποία θα συγκρίνει πλήρως αργότερα.

- *Ταίριασμα των τεμαχίων.* Κάθε ζεύγος παρομοίων τεμαχίων αποτελεί και ένα ξεχωριστό πρόβλημα το οποίο λύνεται σε μία εκτέλεση η οποία προσδιορίζει τις ομοιότητες μεταξύ των στοιχείων που τα αποτελούν. Το αποτέλεσμα είναι ένα σύνολο χαρτογραφήσεων που περιέχουν τις αντιστοιχίες μεταξύ των συστατικών των τεμαχίων οι οποίες μετά συγχωνεύονται σε ένα αποτέλεσμα.

Ταίριασμα προσανατολισμένο στην επαναχρησιμοποίηση: Ενώ το COMMA εμπεριέχει την χρήση παλιότερων ταιριασμάτων (και κάνει χρήση τύπου ένωση, π.χ. χρησιμοποιεί τις Α-Β και Β-Γ για να αποφασίσει για την Α-Γ) το COMMA++ επεκτείνει αυτή τη τεχνική σε πιο έμμεσες περιπτώσεις (σε περίπτωση που δεν υπάρχουν):

- *Άμεση χαρτογράφηση*. Η ιδανική περίπτωση όπου μια ή πολλαπλές προηγούμενες χαρτογραφήσεις υπάρχουν ήδη για το δοσμένο πρόβλημα.
- *Πλήρεις μονοπάτι χαρτογράφησης*. Σε αυτή την περίπτωση ο αλγόριθμος ψάχνει για μονοπάτια διαφορετικών μηκών 2, 3 κ.ο.κ. ώστε να 'συνδεθούν' οι οντολογίες εισόδου σε ένα μονοπάτι χαρτογραφήσεων
- *Ατελές μονοπάτι χαρτογράφησης*. Σε αυτή την περίπτωση κάνει αναζήτηση για χαρτογραφήσεις, οι οποίες μπορεί να μην έχουν γίνει ακόμη, .αλλά είναι πιο εύκολα υπολογίσιμες από τον απευθείας υπολογισμό του προβλήματος.

Να σημειωθεί πως ειδικά η τελευταία περίπτωση απαιτεί ευριστικές για να αποφασίσει τέτοια 'ελαφρές' χαρτογραφήσεις. Για αυτό το λόγο ψάχνει για υπάρχουσες χαρτογραφήσεις που περιέχουν μοντέλα που είναι παρόμοια με τουλάχιστον ένα από τα μοντέλα εισόδου. Αυτή η τεχνική αποτελεί μια παραλλαγή της εύρεσης παρόμοιων τεμαχίων στην παραπάνω τεχνική.

Τέλος να συμπληρώσουμε πως μπορεί να χρησιμοποιηθεί όχι μόνο για επίλυση προβλημάτων αλλά και για αξιολόγηση των διαφόρων αλγορίθμων και στρατηγικών ταιριάσματος. Όσον αφορά τις επιδόσεις του αλγορίθμου δεν έχουν γίνει εκτενείς δοκιμές για την αξιολόγηση των αποτελεσμάτων του.

5.2.14 IF-Map

Ο αλγόριθμος αυτός [43] χρησιμοποιεί τη ροή πληροφορίας (Information Flow). Πιο συγκεκριμένα βασίζεται σε κοινές παραδοχές μεταξύ των οντολογιών (θεωρείται πως για την δημιουργία των δύο οντολογιών, υπάρχει μια τρίτη, κοινή, η οποία χρησιμοποιείται σαν βάση, για τη δημιουργία της δομής τους και της ονοματολογίας. Δηλαδή οι παραδοχές που έχουν γίνει για τη δημιουργία των δύο οντολογιών έχουν κάποια κοινή βάση). Όσο περισσότερες κοινές παραδοχές υπάρχουν, τόσο μεγαλύτερη

ροή πληροφορίας υπάρχει. Για τον προσδιορισμό της ροής πληροφορίας στον αλγόριθμο χρησιμοποιείται η τοπική λογική. Η τοπική λογική ορίζεται ως ένα τετραπλό L=(I, T, |=, |-) όπου I είναι ένα σύνολο στιγμιότυπων, T ένα σύνολο τύπων, |= μια δυαδική σχέση μεταξύ των I και T και |- μια δυαδική σχέση μεταξύ υποσυνόλων του T. Η τοπική λογική χωρίζεται σε 2 βασικά κομμάτια: την θεωρία (theory), δηλαδή το ζεύγος (T, |-) το οποίο περιέχει πληροφορίες για τις έννοιες της κάθε οντολογίας και τις σχέσεις μεταξύ τους (όπως ΕίναιΥποσύνολοΤου, ΕίναιΞένοΜε κ.λ.π), καθώς και την κατάταξη (classification), δηλαδή το ζεύγος (I, T, |=) το οποίο εμφανίζει σε ποιες έννοιες αντιστοιχούν τα στιγμιότυπα. Έτσι ο αλγόριθμος παράγει πρώτα την θεωρία για κάθε οντολογία, μετά δημιουργεί τις κατατάξεις μεταξύ των στιγμιότυπων και των εννοιών και μετά δημιουργεί ισομορφισμούς που αντιστοιχούν στις θεωρίες. Σαν ισομορφισμό εννοούμε ένα σύνολο < f*, f** > όπου f* μια συνάρτηση που ταιριάζει τις έννοιες T → T' και f** μια συνάρτηση που ταιριάζει τα στιγμιότυπα I → I'. Η δημιουργία ισομορφισμών γίνεται παίρνοντας όλες τις πιθανές περιπτώσεις αποκόπτοντας αυτές που δεν ταιριάζουν με την θεωρία καθώς και αυτές που δεν ταιριάζουν με ισομορφισμό των σχέσεων των οντολογιών.

Ο αλγόριθμος εξαρτάται από κοινές ονομασίες μεταξύ των σχέσεων των 2 οντολογιών για να παράγει γρήγορα και σωστά αποτελέσματα. Σε περίπτωση διαφορετικών ονομασιών αυξάνεται εκθετικά ο χρόνος εκτέλεσης αφού ελέγχει όλους τους πιθανούς συνδυασμούς ισομορφισμών. Παρόλα αυτά έχει μηχανισμούς για να μειωθεί ο χρόνος εκτέλεσης όπως έλεγχο συμβατότητας τύπων, ισομορφισμό σχέσεων.

5.2.15 RiMOM (Risk Minimization based Ontology Mapping)

Ο RiMOM [87] είναι ένας αλγόριθμος ο οποίος είναι υπεύθυνος για χαρτογράφηση οντολογιών. Αρχικά χρησιμοποιεί διάφορες τεχνικές για να προετοιμάσει τα διάφορα στοιχεία των οντολογιών όπως τον χωρισμό σε tokens, την αφαίρεση των stop-words (λέξεων κοινών που δεν έχουν καμία σημασιολογική αξία όπως προθέσεις κ.λ.π), κανονικοποίηση κ.λ.π. Αργότερα χρησιμοποιεί πολλαπλές στρατηγικές οι οποίες αποφασίζουν ανεξάρτητα για την ομοιότητα των οντολογιών τα αποτελέσματα των οποίων αργότερα τα συνδυάζει σε ένα σύνθετο αποτέλεσμα:

- Σύγκριση ονομάτων. Πιο συγκεκριμένα χρησιμοποιεί τον εξής τύπο:

$$\mathrm{sim}(w_1, w_2) = \frac{[\mathrm{simd}(w_1, w_2) + \mathrm{sims}(w_1, w_2)]}{2} \qquad (5.4)$$

όπου $simd(w_1,w_2)$ είναι η ομοιότητα μεταξύ των στοιχείων w_1 και w_2 όσον αφορά το WordNet. Η ομοιότητα μεταξύ 2 εννοιών ορίζεται ως:

$$simd(S_1,S_2) = \frac{2 * logp(s)}{[logp(s_1) + logp(s_2)]} \qquad (5.5)$$

όπου p(s)=αριθμός των s/συνολικό αριθμό. Έτσι η ομοιότητα 2 λέξεων ορίζεται ως η μέγιστη ομοιότητα των εννοιών τους δηλαδή:

$$simd(w_1,w_2) = max[simd(s_{1_j},s_{2_j})] \qquad (5.6)$$

όπου s_{1_j} μια έννοια του S(W1), s_{2_j} μια έννοια του S(W2).

Η $sims(w_1,w_2)$ είναι η στατιστική ομοιότητα μεταξύ των στοιχείων. Για την στατιστική ομοιότητα υπάρχει ένα λεξικό το οποίο υπολογίζει την ομοιότητα μεταξύ των λέξεων βασισμένο στην κατανομή τους στο κείμενο[69]. Ακόμη, πριν εισάγουμε στο λεξικό τις λέξεις ακολουθούμε την τεχνική του tokenization.

- τεχνική βασισμένη στην περιγραφή. Εδώ χρησιμοποιεί τη συχνότητα των λέξεων στη περιγραφή εννοιών για να φτιάξει ένα ταξινομητή Bayes (Bayesian classifier). Μετά χρησιμοποιεί τις περιγραφές στις έννοιες της άλλης οντολογίας για να κάνει προβλέψεις.
- τεχνική βασισμένη στα στιγμιότυπα. Αυτή η στρατηγική χρησιμοποιεί τη συχνότητα λέξεων στο περιεχόμενο των στιγμιότυπων. Πιο συγκεκριμένα, το περιεχόμενο των στιγμιότυπων επεξεργάζεται σαν ένας αριθμός λέξεων που είναι προϊόν διαδικασίας tokenization και αφαίρεσης stop-word. Εδώ χρησιμοποιείται ο Naïve Bayesian (NB) Classifier, για να δημιουργήσει ένα μοντέλο μέσω εκμάθησης.
- τεχνική βασισμένη στην ταξονομία (δομή). Αποφασίζεται η ομοιότητα μέσω του ταξονομικού περιεχόμενου κάθε κλάσης (το οποίο περιέχει τους προγόνους, τους απογόνους της). Αυτό προκύπτει από την διαίσθηση ότι 2 έννοιες πιθανότατα θα ταιριάζουν αν ταιριάζουν και οι υπό-κλάσεις/υπέρ-κλάσεις τους.
- τεχνική βασισμένη σε περιορισμούς. Εδώ ορίζονται ορισμένοι περιορισμοί όπως έννοιες με ίδιο αριθμό ιδιοτήτων τείνουν να ταιριάζουν μεταξύ τους κ.λ.π.

Τέλος υπολογίζει το Bayesian ρίσκο για κάθε χαρτογράφηση και επιλέγει την χαρτογράφηση με το μικρότερο ρίσκο.

Ο αλγόριθμος παράγει ικανοποιητικά αποτελέσματα στη λειτουργία του. Η επιτυχία του οφείλεται κυρίως στη χρήση πληθώρας τεχνικών που εξετάζουν διαφορετικά χαρακτηριστικά των οντοτήτων, και υπολογίζουν την ομοιότητα από άλλη σκοπιά ο

καθένας. Με την χρήση της στατιστικής ομοιότητας, σε συνδυασμό με τις άλλες μεθόδους, πετυχαίνει μεγάλο ποσοστό επιτυχίας ακόμα και όταν τα ονόματα είναι διαφορετικά μεταξύ τους με σημασιολογική σχέση. Με την χρήση των περιορισμών και της ευριστικής γνώσης αυξάνεται ακόμα περισσότερο η ακρίβεια του αλγορίθμου.
Παρόλα αυτά έχει και κάποια αρνητικά, τα οποία πληρούν βελτίωση. Τα βάρη των περιορισμών παρέχονται από τον χρήστη και έτσι μειώνεται ο αυτοματισμός, καθώς και η απόδοσή του. Ακόμη έχει χαμηλή απόδοση σε περιπτώσεις όπου τα στιγμιότυπα έχουν λίγες ομοιότητες μεταξύ τους. Επιπρόσθετα, οι τεχνικές με βάση το όνομα και τα στιγμιότυπα είναι ευαίσθητες αφού στηρίζονται σε ένα μόνο είδος πληροφορίας των εννοιών. Ακόμη βελτίωση χρειάζεται και στους περιορισμούς, αφού το 12% περίπου από τις αποτυχίες για σωστό ταίριασμα οφείλεται σε λανθασμένο φιλτράρισμα από τους κανόνες περιορισμών.

5.2.16 ASMOV

Ο αλγόριθμος ASMOV [39] υπολογίζει επαναληπτικά την ομοιότητα μεταξύ εννοιών δύο οντολογιών αναλύοντας τέσσερα χαρακτηριστικά:

- την περιγραφή κειμένου-textual description (που περιλαμβάνει το id, την ετικέτα και τα σχόλια)
- την εξωτερική δομή (πρόγονοι και απόγονοι κόμβοι)
- την εσωτερική δομή (περιορισμοί των ιδιοτήτων για τις κλάσεις) και
- την μεμονωμένη ομοιότητα.

Στο τέλος κάθε επανάληψης μια διαδικασία αποκοπής απορρίπτει τα λανθασμένα ταιριάσματα αναλύοντας δύο σημασιολογικές ασυνέπειες:

- *χιαστή ταίριασμα (crisscross mappings)*: Η χιαστή χαρτογράφηση προκύπτει όταν μια οντότητα και το παιδί της, ταιριάζει με μια οντότητα και το πατέρα της αντίστοιχα.
- *πολλά-σε-ένα χαρτογραφήσεις (many-to-one mappings)*: Οι πολλά-σε-ένα χαρτογραφήσεις είναι ασυνεπείς όταν δεν μπορεί να αξιολογηθεί από την οντολογία αν είναι ισοδύναμες ή η μια είναι απόγονος της άλλης.

Η επανάληψη σταματά όταν η διαφορά στις τιμές μεταξύ δύο συνεχόμενων επαναλήψεων είναι κάτω από ένα δοσμένο όριο και δεν βρεθούν ασυνέπειες ή όταν βρεθεί μια κυκλική κατάσταση.
Τα αποτελέσματα και των τεσσάρων λειτουργιών συνδυάζονται μεταξύ τους σε μια τιμή χρησιμοποιώντας το άθροισμα με βάρη. Αφού ολοκληρωθεί η επαναληπτική διαδικασία,

ξεκινά η επικύρωση της χαρτογράφησης. Αυτό το βήμα πραγματοποιεί μια δομική ανάλυση χρησιμοποιώντας γράφους που προέκυψαν από την χαρτογράφηση και πληροφορίες από τις οντολογίες. Η δομική ανάλυση γίνεται σε τρεις φάσεις: επικύρωση κλάσεων, επικύρωση ιδιοτήτων και επικύρωση ιδιοτήτων-εννοιών. Αν βρεθεί κάποια ασυνέπεια ξεκινά πάλι η επαναληπτική διαδικασία. Τέλος να συμπληρώσουμε πως ο παρών αλγόριθμος παράγει κλάση-προς-κλάση και ιδιότητα-προς-ιδιότητα χαρτογραφήσεις, συμπεριλαμβάνοντας χαρτογραφήσεις από ιδιότητες αντικειμένου προς ιδιότητες τύπου και αντίστροφα.

Η απόδοση του αλγορίθμου αυτού είναι πολύ ικανοποιητική. Αυτό οφείλεται στη χρήση διαφόρων διαφορετικών χαρακτηριστικών των οντολογιών καθώς και στην ικανότητα του να προσαρμόζει τα βάρη στα χαρακτηριστικά της κάθε οντολογίας καθιστώντας έτσι τον ASMOV ένα πολύ ευέλικτο αλγόριθμο. Παρόλα αυτά, η διαδικασία, ανάθεσης βαρών χρειάζεται ακόμα πολλές βελτιώσεις.

Ακόμη η διεργασία σημασιολογικής επικύρωσης καταφέρνει να απορρίψει πολλά λανθασμένα ταιριάσματα. Παρόλα αυτά, ο αλγόριθμος υπολογίζει την πιθανότητα ταιριάσματος για κάθε ένα ζεύγος, και με την επαναληπτική διαδικασία, καθίσταται η χρήση του απαγορευτική για μεγάλες οντολογίες από άποψη χρόνου. Ακόμη η επικύρωση χαρτογράφησης εξαρτάται από την οντολογία πηγής καθιστώντας έτσι τη χαρτογράφηση μονομερής. Επιπρόσθετα, η χρήση κατωφλίου μπορεί να κάνει τον αλγόριθμο να συγκλίνει πρόωρα.

5.2.17 GAOM (Genetic Algorithm based Ontology Matching)

Χρησιμοποιεί ένα γενετικό αλγόριθμο, για να βελτιώσει ένα υπάρχον ταίριασμα. Αρχικά, δημιουργεί ένα αριθμό ταιριασμάτων (pop-size) μεταξύ των O_1 και O_2, σε μορφή πινάκων (είτε με είσοδο του χρήστη, είτε με άλλες τεχνικές). Μετά ελέγχει κατά πόσον πλησιάζει κάθε ταίριασμα στην λύση, μέσω μιας εξίσωσης (fitness):

$$\text{Similarity}_{O_1,O_2}(M) = \frac{f\left(\left(F_{O_1} \cap F_{O_2}\right) \mid M\right)}{f\left(\left(F_{O_1} \cap F_{O_2}\right) \mid M\right) + \alpha f\left(\left(F_{O_1} - F_{O_2}\right) \mid M\right) + \beta f\left(\left(F_{O_2} - F_{O_1}\right) \mid M\right)} \quad (5.3)$$

Επιλέγει διάφορα άτομα βασισμένα στο fitness τους και «μεταλλάσσονται» ώστε να δημιουργηθεί ένας νέος πληθυσμός, ο οποίος θα χρησιμοποιηθεί στην επόμενη επανάληψη του αλγορίθμου. Ο GAOM [91] τελειώνει μετά από ένα πεπερασμένο αριθμό γενεών (επαναλήψεων) ή μόλις φτάσει ένα συγκεκριμένο κατώφλι η εξίσωση fitness.

5.2.18 MapPSO

Στην μέθοδο MapPSO [11] χρησιμοποιείται μια παραλλαγή της τεχνικής *Βελτιστοποίηση με Σμήνη Σωματιδίων (particle swarm optimisation),* η οποία είναι εμπνευσμένη από την κοινωνική αλληλεπίδραση ζώων σε σμήνη. Όπως ακριβώς και στο φυσικό κόσμο, υπάρχει ένα 'σμήνος'-μια ομάδα από ζώα-σωματίδια. Το κάθε σωματίδιο αντιστοιχεί σε ένα ταίριασμα μεταξύ των δύο οντολογιών. Ο αλγόριθμος εκτελείται με πολλαπλές επαναλήψεις οπού σε κάθε επανάληψη μεταβάλλονται οι αντιστοιχίες των χαρτογραφήσεων των σωματιδίων τυχαία. Για τον έλεγχο των χαρτογραφήσεων χρησιμοποιείται η αντικειμενική συνάρτηση, η οποία παρέχει μια τιμή ικανότητας (fitness value) για κάθε υποψήφιο ταίριασμα. Για τον υπολογισμό της, χρησιμοποιείται ένα σύνολο από τεχνικές:

- Λεξικογραφική απόσταση ονομάτων και ετικετών
- Απόσταση WordNet για ονόματα και ετικέτες
- Ομοιότητα διανυσματικού κειμένου (Vector Space Similarity) για σχόλια οντοτήτων
- Απόσταση ιεραρχίας για να διαδοθεί η ομοιότητα των υπέρ-κλάσεων/υπέρ-ιδιοτήτων
- Δομική ομοιότητα των κλάσεων που προήλθε από τις ιδιότητες που τις έχουν ως πεδία τιμών και ορισμού
- Δομική ομοιότητα των ιδιοτήτων που αντλούνται από τις κλάσεις των πεδίων τιμών και ορισμού τους

Ο MapPSO, όπως και ο GAOM που παρουσιάστηκε παραπάνω, παρέχουν ικανοποιητικά αποτελέσματα. Παρόλα αυτά, υπάρχουν κάποια σημεία τα οποία θα μπορούσαν να βελτιώσουν κατά πολύ την απόδοση του αλγορίθμου. Για παράδειγμα, οι περιπτώσεις όπου η χαρτογράφηση-ταίριασμα εξαρτιόταν από δομικά κριτήρια προκάλεσαν πρόβλημα στον αλγόριθμο. Αυτό μπορεί να λυθεί με την χρήση και άλλων τεχνικών οι οποίες θα στηρίζονται στην δομή των οντολογιών για να αποφασίσουν. Ένα άλλο πρόβλημα του αλγορίθμου είναι ότι αναλόγως την περίπτωση χρειάζεται και διαφορετικός συνδυασμός από τεχνικές για να λειτουργήσει ο αλγόριθμος στο βέλτιστο του, πράγμα το οποίο δεν μπορεί να επιλέξει ο αλγόριθμος. Έτσι, παρά την καλή λειτουργία του είναι πιθανόν να παράγει φτωχά αποτελέσματα λόγω χρήσης λανθασμένων τεχνικών. Τέλος να σημειώσουμε πως ο αλγόριθμος είναι μη

ντετερμινιστικός και έτσι θα παρουσιάσει διαφορές στα αποτελέσματα αν τον χρησιμοποιήσουμε πολλαπλές φορές στο ίδιο πρόβλημα.

5.2.19 Dublin20

Για τον υπολογισμό της εσωτερικής ομοιότητας κάνει συγκρίσεις με την χρήση διαφόρων τεχνικών. Οι τεχνικές αυτές περιλαμβάνουν το WordNet, τον αλγόριθμο Jaro-Winker, τον edit distance του Levenshtein. Από τα αποτελέσματα των τεχνικών αυτών επιλέγονται αυτά με την μεγαλύτερη τιμή.

Για τον υπολογισμό της εξωτερικής ομοιότητας χρησιμοποιεί την δομή των οντολογιών. οπού για κάθε οντότητα, δημιουργεί ένα πίνακα που αποθηκεύει την σχέση μεταξύ της οντότητας και των άλλων οντοτήτων. Στη συνέχεια, υπολογίζει ομοιότητα βάσει αυτών των σχέσεων.

Επίσης χρησιμοποιεί προηγούμενες, γνωστές χαρτογραφήσεις για να υπολογίσει το παρόν πρόβλημα. Έτσι, αν π.χ. η μια οντολογία είναι η Α και η άλλη η Β, υποθέτουμε μια τρίτη Γ και μια χαρτογράφηση (Β,Γ). Υπολογίζουμε την ομοιότητα για κάθε στοιχείο της Α με την Β και την Γ ξεχωριστά. Μετά, για κάθε ζεύγος (U_β, U_γ) υποθέτουμε τη μέγιστη τιμή των (U_α, U_β) και (U_α, U_γ) σαν την τιμή που θα χρησιμοποιηθεί στον αλγόριθμο από κάτω για την ομοιότητα (U_α, U_β). Αυτή η λειτουργία γίνεται και για την εσωτερική και για την εξωτερική ομοιότητα.

Ο αλγόριθμος [34] έχει γενικά ένα ικανοποιητικό ποσοστό απόδοσης. Η χρήση των προηγούμενων χαρτογραφήσεων βελτιώνει τα αποτελέσματα. Ακόμη, κατά την εσωτερική ομοιότητα, απορρίπτονται οι ομοιότητες που είναι μικρότερες από ένα κατώφλι και έτσι αποφεύγεται η υπερεκτίμηση χαμηλών ομοιοτήτων. Επιπροσθέτως, η χρήση της εξωτερικής ομοιότητας αυξάνει περισσότερο την επιτυχία των αποτελεσμάτων.

Παρόλα αυτά, ο αλγόριθμος έχει χαμηλά ποσοστά επιτυχίας αν οι δύο οντολογίες δεν είναι δομικά όμοιες, αφού έτσι αποτυγχάνει η εξωτερική ομοιότητα. Ακόμα, παράγει στην έξοδό του 1-προς-1 χαρτογραφήσεις μεταξύ των οντοτήτων. Αυτό, έχει αρνητική συνέπεια ως προς την αντιγραφή δεδομένων από τη μια οντολογία στην άλλη στο σενάριο που εξετάζουμε. Αν δηλαδή, υπάρχουν δύο οντότητες της οντολογίας πηγής, οι οποίες ταιριάζουν με μια της οντολογίας προϊόντος, θα πρέπει τα στιγμιότυπα και των δύο κλάσεων να αντιγραφούν στην κλάση στην οποία ταιριάζουν. Με την 1-προς-1 χαρτογράφηση, το ένα ζεύγος από αυτά θα απορριφθεί, με αποτέλεσμα κάποια δεδομένα να μην αντιγραφούν στη θέση που χρειάζεται. Ένα τελευταίο αρνητικό του

αλγορίθμου, είναι ότι η χρήση προηγούμενων χαρτογραφήσεων μπορεί να είναι παραπλανητική αν η τρίτη οντολογία είναι πολύ διαφορετική από την οντολογία πηγής.

5.2.20 Υπόλοιποι αλγόριθμοι

Ακόμη υπάρχει μια πληθώρα από αλγόριθμους οι οποίοι χρησιμοποιούν ένα αριθμό από ανεξάρτητες τεχνικές για να υπολογίσουν ομοιότητα τις οποίες μετά τις συνθέτουν χρησιμοποιώντας άθροισμα με βάρη, για να υπολογιστεί η τελική ομοιότητα. Οι αλγόριθμοι αυτοί δεν έχουν τίποτα το ενδιαφέρον πέρα από τις τεχνικές και έτσι παρουσιάζονται επιγραμματικά:

ο *DSSim* [61] που χρησιμοποιεί το WordNet καθώς και την τεχνική Jaro-Winkler. Για τον συνδυασμό των τιμών, δεν χρησιμοποιείται το άθροισμα με βάρη, αλλά ο κανόνας του Dempster[84, 20].

Ο *H-match* [14] που χρησιμοποιεί το WordNet για εξαγωγή σχέσεων μεταξύ των εννοιών (πιο γενικό, πιο ειδικό, ίσο, διαφορετικό). Ακόμα χρησιμοποιεί το n-gram και μια τεχνική βασισμένη στην δομή των οντολογιών: δίνεται ένα συγκεκριμένο βάρος W_{sp} για τις ιδιότητες που είναι σημαντικές για το νόημα της έννοιας και ένα άλλο για πιο αδύναμες ιδιότητες W_{wp} και μετράται η ομοιότητα τους μεταξύ δύο κόμβων. Τέλος η ομοιότητα των δύο ιδιοτήτων πολλαπλασιάζεται με τη διαφορά των βαρών τους για να βρεθεί η σημασιολογική ομοιότητα. Σε αυτή τη τεχνική γίνεται σημασιολογική σύγκριση μεταξύ των εννοιών.

ο *JHU/APL* [10] που χρησιμοποιεί την μετρική Jaro-Winkler, την τεχνική n-gram καθώς και ένα λεξικό. Συγκεκριμένα, χρησιμοποιεί τη μηχανή αναζήτησης κειμένου Lucene [50] η οποία ευρετηριοποιεί το κείμενο ως προς τους όρους στη μια οντολογία και χρησιμοποιεί τους όρους της άλλης ως κλειδιά για αναζήτηση στο ευρετήριο. Μεγάλο ποσοστό επιτυχιών σημαίνει και μεγάλη ομοιότητα ενώ μικρό ποσοστό το αντίθετο. Ακόμα, χρησιμοποιεί τον αλγόριθμος Neighborhood Match, όπου η ομοιότητα 2 κόμβων-εννοιών καθορίζεται από τον αριθμό των γειτονικών κόμβων και ακμών που ταιριάζουν. Επίσης κάνει χρήση κανόνων (“*Αν μια κλάση σε μια οντολογία δειχθεί ισοδύναμη με μια άλλη, τότε και οι υπέρ-κλάσεις τους θα είναι ισοδύναμες*”, “*Αν τα πεδία τιμών και ορισμού δύο ιδιοτήτων είναι ίσα, τότε θα είναι και οι ίδιες οι ιδιότητες*”). Τέλος, χρησιμοποιεί τον υβριδικό αλγόριθμο Onto-Mapology όπου παράγει αρχικά, ταιριάσματα με βάση την τεχνική Jaro-Winkler, τα υπολειπόμενα στοιχεία τα ταιριάζει σύμφωνα με την τεχνική ταιριάσματος λημμάτων (lemma matching technique) και όσα απομείνουν πάλι τα ταιριάζει σύμφωνα με το τύπο.

5.3 Επίλογος

Σε αυτό το κεφάλαιο μελετήσαμε διάφορους αλγορίθμους που χρησιμοποιούνται για την εύρεση πανομοιότυπων οντοτήτων σε δύο οντολογίες (ταίριασμα οντολογιών). Παρουσιάσαμε τα κυριότερα χαρακτηριστικά τους, και το τρόπο λειτουργίας τους. Ακόμα, παρουσιάσαμε κάποια κύρια μειονεκτήματα τους. Στη συνέχεια, εξετάζουμε την υλοποίηση στην οποία καταλήξαμε με βάση τα χαρακτηριστικά, τις αδυναμίες και τα προτερήματα των αλγορίθμων αυτών.

ΚΕΦΑΛΑΙΟ 6

ΠΕΡΙΓΡΑΦΗ ΜΕΘΟΔΟΛΟΓΙΑΣ ΤΑΙΡΙΑΣΜΑΤΟΣ

6.1 Εισαγωγή

Όπως ειπώθηκε και στο Κεφάλαιο 1, οι διαδικασίες ταιριάσματος και μεταφοράς των τιμών, γίνονται μεταξύ δύο οντολογιών. Η οντολογία πηγής (Source Ontology - SO), είναι η οντολογία του πωλητή, η οποία περιέχει τα δεδομένα των προϊόντων του, τα οποία θέλουμε να επεξεργαστούμε. Η οντολογία στόχου (Target Ontology ή Product Ontology - PO), αποτελεί την οντολογία του χρήστη, στην οποία θέλουμε να συγκεντρώσουμε δεδομένα από διάφορα προϊόντα και πωλητές (με πολλαπλές εκτελέσεις για κάθε οντολογία πηγής) για μετέπειτα χρήση. Για περισσότερες λεπτομέρειες η γενική οντολογία προϊόντος παρατίθεται στο Παράρτημα Β. Είναι ολοφάνερο, πως η PO θα πρέπει να περιέχει όσο το δυνατόν περισσότερα χαρακτηριστικά διαφόρων προϊόντων σωστά οργανωμένα ώστε να διευκολυνθεί η διαδικασία ταιριάσματος και μετάφρασης. Ακόμα, η PO που θα χρησιμοποιηθεί θα πρέπει να περιέχει κάποιες ιδιότητες σταθερές για να εξασφαλιστεί η σωστή λειτουργία του αλγορίθμου. Οι ιδιότητες αυτές είναι:

- η Ιδιότητα Προεπιλογής (Default Property): κατά την αντιγραφή των πληροφοριών από την οντολογία πηγής στην οντολογία προϊόντος υπάρχει περίπτωση μερικές τιμές να μην ταιριάζουν πουθενά ώστε να πραγματοποιηθεί η αντιγραφή. Παρόλαυτα, δεν πρέπει να χάσουμε καμία πληροφορία κατά την αντιγραφή. Έτσι έχουμε μια ιδιότητα, την ιδιότητα προεπιλογής, στην οποία εισάγουμε τις τιμές, που δεν μπορούν να εισαχθούν αλλού, ώστε να μην χαθούν.
- Η hasName ιδιότητα, στην οποία αποθηκεύεται πάντα το όνομα του στιγμιοτύπου του προϊόντος το οποίο κάνουμε εισαγωγή.
- Η hasSellerName ιδιότητα στην οποία αποθηκεύεται το όνομα του πωλητή, από τον οποίο προήλθαν οι πληροφορίες για το συγκεκριμένο προϊόν. Ο λόγος που αποθηκεύεται η συγκεκριμένη πληροφορία είναι η πιθανότητα ύπαρξης του ίδιου προϊόντος από άλλο πωλητή. Έτσι, αν παρουσιαστεί ένα νέο προϊόν προς

εισαγωγή, το οποίο υπάρχει ήδη στην PO, ελέγχεται η τιμή της ιδιότητας αυτής. Αν είναι ίδιες, τότε το προϊόν αυτό υπάρχει ήδη στην οντολογία μας, από παλαιότερο ταίριασμα. Έτσι, ανανεώνουμε τις τιμές του ήδη υπάρχων στιγμιοτύπου με τις νέες. Αν είναι διαφορετικές, τότε το προϊόν αυτό παρέχεται από άλλο πωλητή και έτσι αποθηκεύεται ως διαφορετικό προϊόν.

- Η hasPID ιδιότητα στην οποία αποθηκεύεται ένας αριθμός, μοναδικός για κάθε προϊόν, ο οποίος το χαρακτηρίζει.

Να σημειωθεί πως οι ιδιότητες αυτές, αποκλείονται από τη διαδικασία του ταιριάσματος, μιας και έχουν συγκεκριμένες τιμές που πρέπει να εισαχθούν ανεξαρτήτως των τιμών των ιδιοτήτων της SO.

Ο αλγόριθμος που προτείνεται περιέχει ένα συνδυασμό λεξικογραφικών και σημασιολογικών τεχνικών. Οι τεχνικές αυτές περιλαμβάνουν τις λεξικογραφικές Maedche-Staab, Lin και Jaro καθώς και τις αυστηρά σημασιολογικές Rondriguez-Engenhofer και Wu-Palmer.Οι τεχνικές αυτές χρησιμοποιούνται στο Name Matching με εισόδους τα ονόματα ή μονοπάτια των στοιχείων. Ο όρος μονοπάτια αφορά τις κλάσεις, όπου αντί για τα ονόματα των κλάσεων χρησιμοποιούμε το μονοπάτι από την κλάση αυτή μέχρι την ρίζα της οντολογίας αν φανταστούμε την οντολογία σαν γράφο. Επίσης χρησιμοποιούμε και σύγκριση πεδίων ορισμού (domain) και πεδίου τιμών (range) για τις ιδιότητες. Η σύγκριση αυτή γίνεται χρησιμοποιώντας τα αποτελέσματα-ομοιότητες των κλάσεων οι οποίες πρέπει να έχουν προηγηθεί.

Ένα άλλο στοιχείο που πρέπει να αναφερθεί είναι ότι ο αλγόριθμος χρησιμοποιεί ένα εργαλείο το οποίο ονομάζουμε SimilarityMatrix. Το εργαλείο αυτό το χρησιμοποιούμε για εύκολη και γρήγορη αποθήκευση των ομοιοτήτων των διαφόρων στοιχείων (κλάσεων, ιδιοτήτων, στιγμιοτύπων ή απλών λέξεων) των οντολογιών με μορφή πίνακα, ανάκληση της ομοιότητας 2 στοιχείων, αλλαγή της ή διαγραφή. Ακόμη, το εργαλείο αυτό παρέχει και λειτουργίες για εμφάνιση των πιο πιθανών ταιριασμάτων, καθώς και υπολογισμό της συνολικής ομοιότητας 2 ομάδων από στοιχεία. Για τον υπολογισμό της συνολικής ομοιότητας παίρνει το κάθε στοιχείο της SO και ανακαλεί το πιο πιθανό ταίριασμα της (δηλαδή αυτό με τη μεγαλύτερη τιμή ομοιότητας), με ένα στοιχείο της PO. Ο υπολογισμός της συνολικής ομοιότητας λοιπόν μπορεί να γίνει με 3 τρόπους:

- Παίρνει το πιο πιθανό ταίριασμα για κάθε στοιχείο της SO, όπως αναφέρθηκε και παραπάνω,και υπολογίζει τον μέσο όρο των τιμών αυτών για όλα τα στοιχεία της (AverageScore - AS)

- Υπολογίζει το μέσο όρο, όπως και στον AS υπολογισμό, με τη διαφορά ότι θέτει κάποιο βάρος στα στοιχεία (WeightedAverageScore – WAS). Ο υπολογισμός αυτός χρησιμοποιείται για τον υπολογισμό συνολικής ομοιότητας δύο στοιχείων, από την ομοιότητα των επιμέρους tokens τους. Το βάρος κάθε τιμής, είναι ο αριθμός των συμβόλων που περιέχει το αντίστοιχο token του SO στοιχείου, προς τον αριθμό όλων των συμβόλων του στοιχείου αυτού.
- Από τις τιμές ομοιότητας που επιλέχτηκαν (η μεγαλύτερη τιμή για κάθε στοιχείο της SO) επιλέγει τη μεγαλύτερη, την οποία και επιστρέφει (BestSimilarity - BS). Επιστρέφει με άλλα λόγια, τη μεγαλύτερη τιμή που υπάρχει στο πίνακα του SimilarityMatrix.

Επίσης ο αλγόριθμος χρησιμοποιεί μια τεχνική, την έλεγχο βαρών (Weight_Checking - WC). Σε αυτή την τεχνική, ελέγχει όλες τις επιμέρους τιμές ομοιότητας, πριν τις συνδυάσει, σε περίπτωση που κάποια δεν είναι έγκυρη. Αν δεν είναι, κατανέμει το αντίστοιχο βάρος της στα υπολειπόμενα βάρη που αντιστοιχούν σε έγκυρες τιμές και το μηδενίζει. Ουσιαστικά αποκλείει από το συνδυασμό των τιμών που δεν είναι έγκυρες, εξασφαλίζοντας παράλληλα ότι το άθροισμα των βάρών των έγκυρων τιμών θα είναι 1. Μια τιμή δεν είναι έγκυρη είτε αν έχει υπάρξει κάποιο λάθος κατά τον υπολογισμό της (για την περίπτωση του Name Matching π.χ. μη σωστή χρήση της βιβλιοθήκης WordNet) είτε αν δεν υπάρχουν δεδομένα για τον υπολογισμό ή δεν είναι αρκετά (για την περίπτωση του domain και range matching π.χ. αν έχουν διαφορετικούς τύπους range). Έτσι θα μπορούσαμε να πούμε πως η συνάρτηση έλεγχου βαρών είναι μια λειτουργία που επεξεργάζεται τα αρχικά βάρη και τις επιμέρους ομοιότητες, και μας δίνει τα βάρη που θα χρησιμοποιηθούν τελικά, για τον υπολογισμό της ομοιότητας.

Ακόμα, υπάρχουν κάποια δεδομένα που δίνει ο χρήστης στο πρόγραμμα για μεγαλύτερη ευελιξία. Ένα από αυτά είναι ένα άνω όριο, πάνω από το οποίο θα δέχεται το πρόγραμμα ένα ταίριασμα, ενώ σε αντίθετη περίπτωση θα το απορρίπτει. Το όριο αυτό φυσικά πρέπει να κυμαίνεται μεταξύ 0 και 1 όπως και οι τιμές ομοιότητας. Στις δοκιμές μας, πειραματικά έχουμε ορίσει ως όριο την τιμή 0.7. Ακόμα, ο χρήστης δίνει το όνομα της ιδιότητας της PO, που θα χρησιμοποιηθεί σαν Default Property και θα εξαιρεθεί από τη διαδικασία ταιριάσματος.

Τέλος, να σημειώσουμε πως στο τέλος κάθε ενότητας περιγραφής του αλγορίθμου θα παρουσιάζεται και ο ψευδοκώδικας της αντίστοιχης λειτουργίας που παρουσιάστηκε στο εδάφιο αυτό.

6.2 Μεθοδολογία

Τα γενικά βήματα του αλγορίθμου του shopbot (AlgS) είναι η αρχικοποίηση των βαρών που θα χρησιμοποιηθούν μετέπειτα. Μετά, διαβάζει τις δύο οντολογίες, την SO καθώς και την PO για να μπορούμε να επεξεργαστούμε τα στοιχεία τους. Ακολουθεί ο έλεγχος και το ταίριασμα στοιχείων από τη μια οντολογία στην άλλη, και τέλος η ανάθεση-αντιγραφή τιμών από την οντολογία πηγής στην οντολογία προϊόντος.

Αλγόριθμος 1. Ο βασικός αλγόριθμος ενός ShopBot.

```
//-------------------------------------------------------------------------------------------------------------
// The main ShopBot algorithm
Begin
    Foreach Seller
        InitializeWeights()
        Get model of the source ontology
        CompareClasses(ProductOntology, SourceOntology, Weights)
        CompareProperties(ProductOntology, SourceOntology, Weights)
        AssignInstances()
    EndFor
End
//-------------------------------------------------------------------------------------------------------------
```

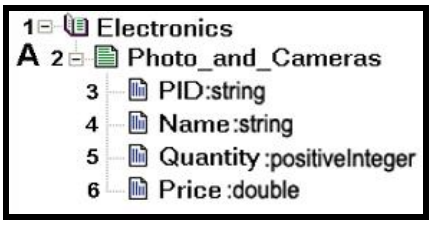
Περνώντας σε λεπτομέρειες λοιπόν, πρώτα αρχικοποιούμε τα βάρη που θα χρησιμοποιηθούν μετέπειτα για τις διάφορες ομοιότητες που θα προκύψουν από τις τεχνικές που χρησιμοποιούμε στον παρόντα αλγόριθμο. Τα βάρη αυτά είναι σταθερά και έχουν προέλθει πειραματικά. Εδώ, υπάρχει χώρος για βελτίωση των βαρών μέσω μιας συνάρτησης που τα αναθέτει αναλόγως με τις οντολογίες που χρησιμοποιούνται κάθε φορά. Πιο συγκεκριμένα χρησιμοποιούμε 2 πίνακες βαρών ένα για τα βάρη που αντιστοιχούν στις τεχνικές του Name Matching που αναφέρθηκαν προηγουμένως, καθώς και ένα για τη συνολική ομοιότητα των ιδιοτήτων (Name Matching, έλεγχος domain και range). Μετέπειτα, ο αλγόριθμος διαβάζει και ‘ανοίγει’ τις οντολογίες ώστε να επεξεργαστεί τα στοιχεία τους και να αποθηκεύσει τις κλάσεις και τις ιδιότητες τους.

Εδώ θεωρούμε πως οι οντολογίες είναι αποθηκευμένες σε ένα τοπικό χώρο και διαβάζονται ως αρχεία.

Αφού αρχικοποιηθούν τα βάρη και ανοιχτούν οι οντολογίες, ο αλγόριθμος υπολογίζει την ομοιότητα των κλάσεων και των ιδιοτήτων αποθηκεύοντας τα αποτελέσματα στον Class_SimilarityMatrix (CSM) και Property_SimilarityMatrix (PSM) αντίστοιχα.

6.2.1 Ομοιότητα κλάσεων

Για τον υπολογισμό της ομοιότητας των κλάσεων αρχικοποιείται πρώτα ο CSM. Μετά, για κάθε ζεύγος κλάσεων (μια από κάθε οντολογία) κάνει τα εξής:



Εικόνα 6.1 Παράδειγμα Οντολογίας.

Παίρνει το μονοπάτι (κλάση προς κλάση) από την προς εξέταση κλάση μέχρι τη ρίζα της οντολογίας. Έπειτα, σπάει το μονοπάτι σε επιμέρους, βασικές λέξεις που το συνθέτουν. Το σπάσιμο αυτό γίνεται με βάση κεφαλαίους χαρακτήρες, αριθμούς και σύμβολα. Για την κλάση Quantity της οντολογίας στην εικόνα 6.1, το μονοπάτι θα είναι Electronics_Photo_and_Cameras_ Quantity. Μετά, θα σπάσει το μονοπάτι στις εξής επιμέρους λέξεις: Electronics, Photo, and, Cameras και Quantity. Έτσι κάθε κλάση θα περιλαμβάνει ένα αριθμό λέξεων που την χαρακτηρίζουν. Τέλος, συγκρίνονται οι λέξεις που προέκυψαν. Η σύγκριση γίνεται με την χρήση του ClassToken_SimilarityMatrix. Για κάθε ζεύγος λέξεων, μια από κάθε κλάση, υπολογίζει τη Name Matching Similarity χρησιμοποιώντας τις τεχνικές που αναφέρθηκαν στην ενότητα 6.1. Τα αποτελέσματα-ομοιότητες των τεχνικών αυτών, συνδυάζονται μεταξύ τους, για να υπολογιστεί μια συνολική ομοιότητα για τις λέξεις αυτές. Ο συνδυασμός γίνεται χρησιμοποιώντας τα βάρη που δόθηκαν από το χρήστη, αφού περάσουν τη λειτουργία του έλεγχου βαρών (Weight_Checking). Τη συνολική ομοιότητα των δύο λέξεων την αποθηκεύει στον ClassToken_SimilarityMatrix. Υπολογίζει λοιπόν και αποθηκεύει τις τιμές του ClassToken_SimilarityMatrix για όλους τους συνδυασμούς των λέξεων που περιέχει. Για

τον υπολογισμό της συνολικής ομοιότητας δύο κλάσεων χρησιμοποιεί τον WAS του πίνακα ClassToken_SimilarityMatrix. Χρησιμοποιούμε το μέσο όρο επειδή κάθε λέξη που υπάρχει στο όνομα μιας κλάσης περιέχει πληροφορίες για την κλάση αυτή, οπότε θα πρέπει να παίζει ρόλο για τον υπολογισμό της συνολικής ομοιότητας. Ακόμα, υπάρχει περίπτωση δύο τελείως διαφορετικές κλάσεις, που δεν πρέπει να ταιριάξουν, να περιέχουν κάποιες κοινές ή παρόμοιες λέξεις (και έτσι με μια διαφορετική τεχνική όπως η BS να έχουν υψηλή τιμή ομοιότητας) αλλά θα είναι απίθανο όλες οι λέξεις τους να ταιριάζουν (οπότε με το μέσο όρο θα πέσει η τιμή ομοιότητας). Για παράδειγμα, έστω ότι έχουμε τις κλάσεις με ονόματα BusDriver και BusStation. Με την παραπάνω διαδικασία, θα παραχθεί ο παρακάτω πίνακας:

Πίνακας 6.1 – Παράδειγμα ενός SimilarityMatrix για tokens δύο κλάσεων

	Bus	**Station**
Bus	**1**	0.036
Driver	0.041	0.176

Όπως έχει ειπωθεί, θα πάρουμε τις μεγαλύτερες τιμές για κάθε στοιχείο της SO, δηλαδή της οριζόντιας οντολογίας (Bus και Station) δηλαδή 1 (η τιμή του Bus με το Bus) και 0.176 (η τιμή του Station με το Driver). Το βάρος της πρώτης τιμής θα είναι 3/10=0.3 αφού η λέξη Bus έχει 3 γράμματα ενώ όλη η λέξη BusStation 10. Το βάρος της δεύτερης με τον ίδιο υπολογισμό βγαίνει 0.7. Άρα η συνολική ομοιότητα θα είναι 0.3*1 + 0.7*0.176= 0.423. Τα βάρη τα χρησιμοποιούμε για περιπτώσεις όπως η παραπάνω, όπου οι λέξεις δεν πρέπει να έχουν υψηλή ομοιότητα άλλα έχουν κάποιες κοινές βασικές λέξεις. Σε αυτή την περίπτωση «μετράμε» το πόσο σημαντική είναι η κάθε λέξη ανάλογα με την έκταση της στη συνολική λέξη. Αν παίρναμε τον απλό μέσο όρο (AS) το πρώτο ταίριασμα θα μετρούσε το ίδιο με το δεύτερο (θα είχαμε δηλαδή υπερεκτίμηση της σημασιολογικής συμβολής του), παρόλο που η ουσία της λέξης κρύβεται στο δεύτερο ζεύγος, και το αποτέλεσμα θα ήταν (1+0.176)/2=0.58. Η συμβολή της πρώτης λέξης στην τελική ομοιότητα θα είναι ½=0.5, ενώ της δεύτερης 0.176/2=0.088. Δηλαδή, μόνο η ύπαρξη μεγάλης ομοιότητας σε μια λέξη ανεβάζει την συνολική κατά 0.5 μονάδες. Από το παράδειγμα φαίνεται πως η πρώτη περίπτωση παράγει

ικανοποιητικότερα αποτελέσματα. Την προκύπτουσα τιμή συνολικής ομοιότητας των δύο κλάσεων την αποθηκεύουμε στον CSM.

Αλγόριθμος 2. Σύγκριση κλάσεων (γενικά).

```
//-------------------------------------------------------------------------------------------------------------
Usage: ClassSimilarityMatrix, ProductOntology, SourceOntology, Weights
// Fill the ClassMatrix with the similarities of its objects
CompareClasses(ProductOntology, SourceOntology, Weights)
Input: ProductOntologyClasses, SourceOntologyClasses, Weights
Output: ClassSimilarityMatrix
Begin
        Foreach ProductClass in ProductOntologyClasses
                Foreach SourceClass in SourceOntologyClasses
                        CalculateClassSimilarity(ProductClass, SourceClass, Weights)
                        Update the ClassSimilarityMatrix
                EndFor
        EndFor
End
//-------------------------------------------------------------------------------------------------------------
```

Αλγόριθμος 3. Σύγκριση κλάσεων (για ένα ζεύγος κλάσεων που παίρνει σαν είσοδο).

```
//-------------------------------------------------------------------------------------------------------------
// Compare 2 Strings using various Similarity metrics
Function CalculateClassSimilarity
Input: The name of the HomeObject
        The name of the TargetObject
        Weights
Output: The similarity score of the two objects
Begin
        GetClassPath(HomeObject)
        GetClassPath(TargetObject)
        Split the path of the HomeObject
```

Split the path of the TargetObject
Initialize **ClassTokenSimilarityMatrix**
ForEach HomeName **in** HomeTokens
ForEach TargetName in TargTokens
Calculate the similarity metrics for HomeName and TargetName
//The metrics used in our model are: Maedche-Staab,
//Rondriguez-Engenhofer, Wu-Palmer, Lin, Jaro.
Foreach SimilarityResult
If SimilarityResult <0 Then //Especially used in semantic similarity
Reject SimilarityResult
EndIf
EndFor
Share Weights of the rejected similarity results to the remaining Weights
Foreach SimilarityMetric
Calculate the WeightedSimilarityResult
EndFor
Update the ClassTokenSimilarityMatrix with the WeightedSimilarityResult
EndFor
EndFor
Get the best similarity score
Update the ClassSimilarityMatrix
End

//----------

6.2.2 Ομοιότητα ιδιοτήτων

Για τον υπολογισμό της ομοιότητας των ιδιοτήτων αρχικοποιείται πρώτα ο PSM. Μετά, για κάθε ζεύγος ιδιοτήτων (μια από κάθε οντολογία) κάνει τα εξής βήματα: Αρχικά υπολογίζει την ομοιότητα των ονομάτων των ιδιοτήτων. Έπειτα υπολογίζει την ομοιότητα των πεδίων ορισμού τους και μετά την ομοιότητα των πεδίων τιμών τους. Πριν από τον υπολογισμό της συνολικής ομοιότητας των τριών τιμών, υπολογίζει μια ενδιάμεση τιμή. Η ενδιάμεση τιμή θα είναι το αποτέλεσμα της εξίσωσης: 0.4*

ΟμοιότηταΟνομάτων+0.6*ΟμοιότηταΠεδίουΟρισμού. Αν το αποτέλεσμα της είναι πάνω από ένα δοσμένο όριο, τότε συνεχίζεται η εκτέλεση. Αν όχι, η τιμή ομοιότητας των δύο εξεταζόμενων ιδιοτήτων θα είναι η ενδιάμεση τιμή. Στις οντολογίες, αν δύο ιδιότητες δεν έχουν κοινά ή παρόμοια πεδία ορισμού, υπάρχει πολύ μικρή πιθανότητα να ταιριάζουν μεταξύ τους. Αντίθετα, αν έχουν κοινό domain, υπάρχει περίπτωση να ταιριάζουν ή όχι αναλόγως με τις τιμές ομοιότητας ονόματος και πεδίου τιμών. Αν θέσουμε μικρό βάρος στον έλεγχο domain, τότε είναι πιθανό ιδιότητες με διαφορετικό πεδίο ορισμού να έχουν υψηλή τιμή ομοιότητας (λόγω παρόμοιου ονόματος και πεδίου τιμών). Αν θέσουμε μεγάλο βάρος στον έλεγχο domain, είναι πιθανό ιδιότητες με παρόμοιο domain να έχουν υψηλή ομοιότητα, παρόλο τον έλεγχο ονόματος και range. Έτσι χρησιμοποιούμε αυτή την ενδιάμεση τιμή. Αν οι δύο ιδιότητες έχουν παρόμοιο domain, η ενδιάμεση τιμή θα είναι πάνω από το δοσμένο όριο (αφού το βάρος 0.6 είναι αρκετά μεγάλο ώστε να επηρεάσει την τιμή), και έτσι θα συνεχιστεί η εκτέλεση παίρνοντας το μέσο όρο με τα δοσμένα βάρη(όπου για τους ίδιους λόγους δίνουν μεγαλύτερο βάρος στα ονόματα και τα πεδία τιμών απ' ότι στο πεδίο ορισμού). Αν έχουν διαφορετικό domain, τότε η ενδιάμεση τιμή είναι πολύ μικρή, και έτσι αποθηκεύεται μια μικρή τιμή ομοιότητας για τις ιδιότητες αυτές. Τέλος, εκτελείται ο έλεγχος βαρών και υπολογίζεται η συνολική ομοιότητα με βάση τα βάρη που προέκυψαν από τον έλεγχο, την οποία και αποθηκεύει στον PSM.

Αλγόριθμος 4. Σύγκριση ιδιοτήτων (γενικά).

```
//-----------------------------------------------------------------------------------------------------------
// Compute final similarity for each pair of properties and save the result to PropMatrix
Function CompareProperties
Input: ProductOntologyProperties, SourceOntologyProperties, Weights, AttributeWeights
Output: PropertySimilarityMatrix
Begin
    ForEach ProductProperty in ProductOntologyProperties
        ForEach SourceProperty in SourceOntologyProperties
            CalculatePropertyNameSimilarity()
            CalculatePropertyDomainsSimilarity()
            CalculatePropertyRangesSimilarity()
```

```
                    //We utilize attribute weights for name, domain and range similarity
                    Foreach SimilarityResult
                        If SimilarityResult <0 Then //Especially used in semantic similarity
                                Reject SimilarityResult
                        EndIf
                    EndFor
                    Share AttributeWeights of the rejected results to the remaining AttributeWeights
                    Calculate the WeightedSimilarityResult
                    Update the PropertySimilarityMatrix with WeightedSimilarityResult
                EndFor
            EndFor
End
//----------------------------------------------------------------------------------------------------
```

Η ομοιότητα των ονομάτων των ιδιοτήτων υπολογίζεται κατά ένα τρόπο παρόμοιο με την ομοιότητα των κλάσεων: Κατά πρώτον χωρίζονται τα ονόματα των ιδιοτήτων σε επιμέρους λέξεις με βάση κεφαλαίους χαρακτήρες, αριθμούς και σύμβολα ακριβώς όπως και στις κλάσεις. Οι τιμές ομοιότητας αυτών των λέξεων θα αποθηκευτούν σε ένα νέο βοηθητικό πίνακα, τον PropertyToken_SimilarityMatrix. Για την ομοιότητα μεταξύ τους χρησιμοποιείται το Name Matching και πάλι με τις επιμέρους του τεχνικές(Maedche-Staab, Rondriguez-Engenhofer, Wu-Palmer, Lin, Jaro). Αφού λοιπόν υπολογιστούν οι επιμέρους τιμές περνάνε τα βάρη από τον έλεγχο βαρών και υπολογίζεται η τελική ομοιότητα των δύο λέξεων. Η συνολική ομοιότητα ονόματος των ιδιοτήτων βγαίνει μέσω της τεχνικής WAS του PropertyToken_SimilarityMatrix. Ο λόγος που χρησιμοποιούμε την τεχνική αυτή, είναι και πάλι ο ίδιος. Εδώ να παρατηρήσουμε πως η τεχνική WAS είναι ακόμα πιο σημαντική σε αυτό το σημείο, απ΄ ότι στις κλάσεις αφού συνηθίζεται τα ονόματα των ιδιοτήτων να γράφονται όλα με ένα πρόθεμα Is ή Has μπροστά τους. Έτσι αν είχαμε την τεχνική AS η πλειοψηφία των ζευγών θα είχαν τουλάχιστον ομοιότητα 0.5 όσο άσχετες και να ήταν.

Αλγόριθμος 5. Σύγκριση ιδιοτήτων (Σύγκριση ονόματος).

//----------

// Compare 2 Strings (Property Names) using various Similarity metrics

Input: The name of the HomeObject, The name of the TargetObject, Weights

Output: The similarity score of the 2 objects

Begin

Split the HomeObject

Split the TargetObject

Initialize **TokenSimilarityMatrix**

ForEach HomeName in HomeTokens

ForEach TargetName in TargTokens

Calculate the similarity metrics for HomeName and TargetName

//The metrics used in our model are: Maedche-Staab,

//Rondriguez-Engenhofer, Wu-Palmer, Lin, Jaro.

Foreach SimilarityResult

If SimilarityResult <0 Then //Especially used in semantic similarity

Reject SimilarityResult

Endif

EndFor

Share Weights of the rejected similarity results to the remaining Weights

Foreach SimilarityMetric

Calculate the WeightedSimilarityResult

EndFor

Update the TokenSimilarityMatrix with the WeightedSimilarityResult

EndFor

EndFor

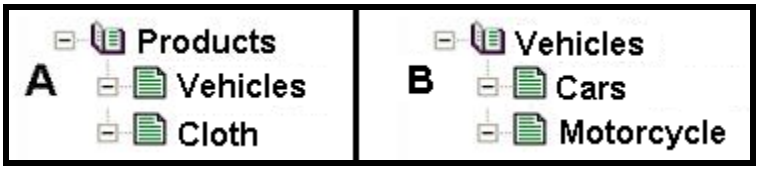
Get the best similarity score

Update the PropertySimilarityMatrix

End

//----------

Για τον υπολογισμό της ομοιότητας πεδίων ορισμού δύο ιδιοτήτων παίρνουμε αρχικά τα πεδία ορισμού (domain) για κάθε ιδιότητα. Πιο συγκεκριμένα παίρνουμε το άμεσο domain της εκάστοτε ιδιότητας (αυτή που έχει οριστεί ως domain της κατά την δημιουργία της οντολογίας) καθώς και τα έμμεσα domain (όλες τις υποκλάσεις του άμεσου domain). Και πάλι εδώ χρησιμοποιείται σαν βοήθεια το εργαλείο SimilarityMatrix και συγκεκριμένα το Domain_SimilarityMatrix το οποίο αποθηκεύει τις ομοιότητες των πεδίων ορισμού (άμεσων και έμμεσων) των ιδιοτήτων. Οι τιμές αυτές, διαβάζονται από τον CSM (να θυμίσουμε εδώ πως το πεδίο ορισμού μιας ιδιότητας είναι ένα σύνολο κλάσεων, αυτών που περιέχουν τη συγκεκριμένη ιδιότητα). Η συνολική ομοιότητα domain των ιδιοτήτων υπολογίζεται με την τεχνική BS επάνω στον πίνακα Domain_SimilarityMatrix. Δηλαδή, δεν παίρνουμε το μέσο όρο των επιμέρους ομοιοτήτων όπως κάναμε παραπάνω με τα ονόματα, για να υπολογίσουμε την συνολική ομοιότητα. Αυτό γίνεται επειδή, διαισθητικά, για να ταιριάζουν δύο ιδιότητες αρκεί να έχουν κοινό ή παρόμοιο ένα από τα domain τους, ακόμα και αν δεν ταιριάζουν τα υπόλοιπα. Και αντίστροφα, ακόμα και αν εξετάζουμε ίδιες ιδιότητες είναι απίθανο να ταιριάζουν όλα τα domain τους. Έτσι παίρνουμε σαν ομοιότητα πεδίου ορισμού των ιδιοτήτων την μεγαλύτερη ομοιότητα των ζευγών domain τους και χρησιμοποιούμε την BS τεχνική που επιστρέφει τη μεγαλύτερη τιμή ομοιότητας. Για παράδειγμα θεωρούμε τις δύο οντολογίες που εμφανίζονται στην εικόνα 6.2:



Εικόνα 6.2 - Παράδειγμα δύο Οντολογιών.

καθώς και δύο ιδιότητες, μια από κάθε οντολογία, με το ίδιο όνομα: Manufacturer. Στην οντολογία A, η ιδιότητα έχει σαν άμεσο πεδίο ορισμού την κλάση Products (και σαν έμμεσες τις Vehicles και Cloth), ενώ στην οντολογία B έχει σαν άμεσο την Vehicles και σαν έμμεσες τις Cars και Motorcycle. Παρακάτω φαίνεται ο πίνακας ομοιότητας των 6 κλάσεων:

Πίνακας 6.2 Ομοιότητα Κλάσεων

	Vehicles	**Cars**	**Motorcycle**
Products	0.3	0.2	0.34
Vehicles	1	0.76	0.72
Cloth	0.4	0.46	0.34

Οι δύο ιδιότητες θα πρέπει να ταιριάζουν απόλυτα, αφού έχουν το ίδιο όνομα και αναφέρονται σε ίδια στοιχεία. Άρα η τιμή ομοιότητας τους θα πρέπει να είναι όσο πιο κοντά στο 1 γίνεται. Παρόλα αυτά, αν πάρουμε το μέσο όρο, AS, θα παραχθεί τιμή ομοιότητας ίση με 0.82 ενώ με την τεχνική BS η τιμή θα είναι 1 (η τιμή ομοιότητας των δύο Vehicles κλάσεων).

Αλγόριθμος 6. Σύγκριση πεδίων ορισμού Ιδιοτήτων (domain).

```
//-------------------------------------------------------------------------------------------------------------
// Get a similarity value for 2 properties comparing their domains
Function CalculatePropertyDomainsSimilarity
Input: ProductOntologyProperty
       SourceOntologyProperty
Output: The similarity score of the 2 objects according to their domain
Begin
    Get domain of the ProductOntologyProperty
    Get domain of the SourceOntologyProperty
    Initialize DomainSimilarityMatrix
    ForEach ProductDomain in ProductPropertyDomains
        ForEach SourceDomain in SourcePropertyDomains
            Get the SimilarityValue for the ClassSimilarityMatrix
            Update DomainSimilarityMatrix
        EndFor
    EndFor
    Get the best similarity score from the DomainSimilarityMatrix
End
//-------------------------------------------------------------------------------------------------------------
```

Για την ομοιότητα των πεδίων τιμών των ιδιοτήτων παίρνουμε τρείς περιπτώσεις:

- **Οι δύο εξεταζόμενες ιδιότητες είναι ιδιότητες με πεδίο τιμών αντικείμενα (Object Property ή ιδιότητες αντικειμένων για συντομία):** Σε αυτή την περίπτωση παίρνουμε τα πεδία τιμών τους και ενεργούμε κατα ένα τρόπο παρόμοιο με τον υπολογισμό της ομοιότητας των domain (Αλγόριθμος 6). Δηλαδή, διαβάζουμε τις ομοιότητες των range-κλάσεων τους από τον CSM, αποθηκεύουμε σε ένα βοηθητικό πίνακα, τον Range_SimilarityMatrix και υπολογίζουμε την τελική ομοιότητα με την τεχνική BS και πάλι, για τους ίδιους λόγους.
- **Η μία ιδιότητα είναι με πεδίο τιμών αντικείμενα ενώ η άλλη είναι με πεδίο τιμών δεδομένα (Datatype Property ή ιδιότητα δεδομένων):** Εδώ, έχουμε θέσει εμπειρικά, αν το πεδίο τιμών δεδομένων της ιδιότητας είναι συμβολοσειρά (string), η ομοιότητα θα είναι 0.5. Σε αντίθετη περίπτωση θα είναι 0. Ο λόγος που ξεχωρίζουμε τις περιπτώσεις είναι γιατί υπάρχει η περίπτωση να έχει τεθεί σαν πεδίο τιμών (range) μιας ιδιότητας ένα αντικείμενο, ενώ σε μια άλλη περίπτωση, η ίδια ιδιότητα μπορεί να έχει οριστεί ως ιδιότητα δεδομένων (Datatype) απλά και μόνο λόγω έλλειψης πληροφοριών. Για παράδειγμα, στις ιδιότητες Manufacturer που αναφέρθηκαν στο παραπάνω παράδειγμα, η μια θα μπορούσε να έχει σαν πεδίο τιμών αντικείμενα (αφού μπορεί να προσφέρονται πληροφορίες, μέσω άλλων ιδιοτήτων, για τους κατασκευαστές όπως τρόπος επικοινωνίας), ενώ η άλλη να παρέχει μόνο το όνομα του κατασκευαστή σαν μια απλή ιδιότητα δεδομένων. Έτσι, υπάρχει περίπτωση να ταιριάζουν δύο ιδιότητες δεδομένων και αντικειμένων γι' αυτό και η πιθανότητα δεν είναι μηδενική. Ο λόγος που χρησιμοποιήσαμε συγκεκριμένα την τιμή 0.5 είναι εμπειρικός. Παρόλα αυτά, είναι απίθανο να ταιριάζει μια ιδιότητα αντικειμένων με μια ιδιότητα δεδομένων με τύπο δεδομένων π.χ. ακέραιο. Αυτός είναι ο λόγος που χρησιμοποιήσαμε μηδενική τιμή ομοιότητας.
- **Οι δύο εξεταζόμενες ιδιότητες είναι ιδιότητες με πεδίο τιμών δεδομένα (Datatype Property ή ιδιότητα δεδομένων):** Σε αυτή την περίπτωση συγκρίνουμε τον τύπο τους. Αν είναι ίδιοι, η ομοιότητα range θα είναι 1. Αν κάποιος από τους τύπους τους είναι συμβολοσειρά η ομοιότητα γίνεται 0.3 ενώ 0 σε οποιαδήποτε άλλη περίπτωση. Και πάλι ξεχωρίζουμε περιπτώσεις. Ο λόγος

είναι, ότι αναλόγως με το δημιουργό των οντολογιών, μια τιμή π.χ. μισθός θα μπορούσε να εμφανιστεί με τύπο ακέραιο ή σαν συμβολοσειρά. Το ίδιο συμβαίνει και με τους άλλους τύπους. Δηλαδή η συμβολοσειρά μπορεί να περιλαμβάνει τους άλλους τύπους. Έτσι πρέπει να έχει κάποια ομοιότητα ακόμα και αν δεν ταιριάζουν οι τύποι, αρκεί τουλάχιστον ένας από αυτούς να είναι με τύπο συμβολοσειρά. Σε αντίθετη περίπτωση, είναι εξαιρετικά απίθανο να εμφανιστεί π.χ. ένας μισθός με μορφή ημερομηνίας. Για αυτό δίνουμε πιθανότητα ομοιότητας 0 σ' αυτές τις περιπτώσεις. Ο λόγος που χρησιμοποιούμε την τιμή 0.3 είναι και πάλι εμπειρικός.

Αλγόριθμος 7. Σύγκριση πεδίων τιμών Ιδιοτήτων (range).

```
//-------------------------------------------------------------------------------------------------------------
// Get a similarity value for 2 properties comparing their ranges
Function CalculatePropertyRangesSimilarity
Input: ProductOntologyProperty, SourceOntologyProperty
Output: The similarity score of the 2 objects according to their range
Begin
    If ProductOntologyProperty is an ObjectProperty Then
        If SourceOntologyProperty is an ObjectProperty Then
            Get ranges of ProductOntologyProperty
            Get ranges of SourceOntologyProperty
            Initialize RangeSimilarityMatrix
            Foreach ProductRange in ProductOntologyPropertyRanges
                Foreach SourceRange in SourceOntologyPropertyRanges
                    Get the SimilarityValue fro the ClassSimilarityMatrix
                    Update RangeSimilarityMatrix
                EndFor
            EndFor
            Get the best similarity score from the RangeSimilarityMatrix
        Else
            Return -2
        EndIf
    Else
        If SourceOntologyProperty = ObjectProperty Then
```

```
                    Return -2
                Else
                    If ProductOntologyProperty.RangeType =
SourceOntologyProperty.RangeType Then
                        Return 1
                    Else
                        Return 0
                    EndIf
                EndIf
            EndIf
End
//----------------------------------------------------------------------------------------------------
```

6.2.3 Ανάθεση Στιγμιότυπων

Η πλήρης και, όσο το δυνατόν περισσότερο, σωστή ανάθεση των στιγμιότυπων από την οντολογία πηγής στην οντολογία προϊόντος είναι μια λειτουργία απαραίτητη για να επιτευχθεί η εξομάλυνση των πληροφοριών από διάφορες τοποθεσίες στον ιστό. Η ανάθεση αυτή, αποτελεί το συνδυασμό διαφόρων επιμέρους λειτουργιών. Στη συνέχεια, περιγράφουμε αυτές τις λειτουργίες καθώς και τον τρόπο που συνδέονται μεταξύ τους για να γίνει η συνολική ανάθεση.

Εισαγωγή-Αρχικοποίηση: Τώρα λοιπόν, ο αλγόριθμος έχει ολοκληρώσει τον έλεγχο για ομοιότητες και τα ταιριάσματα τα οποία έχει έτοιμα στους CSM και PSM. Σαν τελευταίο βήμα, ο αλγόριθμος έχει την ανάθεση των στιγμιότυπων της SO στην PO με την βοήθεια των δύο πινάκων.

Για να γίνει η ανάθεση, αρχικά διαβάζει τους δύο πίνακες ώστε να εξάγει τα πιθανότερα ταιριάσματα σύμφωνα με τις τιμές των δύο πινάκων. Εδώ χρησιμοποιεί την τιμή του άνω ορίου που αναφέραμε προηγουμένως. Επειδή θέλουμε την πλήρη ανάθεση των στιγμιότυπων, δηλαδή να μην υπάρξει καμία πληροφορία που να χαθεί, παίρνουμε το πιθανότερο ταίριασμα για κάθε στοιχείο της οντολογίας πηγής. Έτσι σε αυτό το σημείο θα έχουμε Ν ταιριάσματα, όπου Ν ο αριθμός των στοιχείων (κλάσεις και ιδιότητες) της οντολογίας πηγής. Σε αυτό το σημείο φαίνεται πως δεχόμαστε το ταίριασμα πολλών στοιχείων της SO σε ένα στοιχείο της οντολογίας προϊόντος, ενώ το αντίθετο το απορρίπτουμε. Το γιατί δεχόμαστε τα Ν:1 ταιριάσματα, φαίνεται αν παρατηρήσουμε τις οντολογίες στην εικόνα 6.2 με τις τιμές ομοιότητας του πίνακα 6.1. Θεωρούμε πως

πρέπει να κάνουμε ανάθεση από την οντολογία Β στην Α. Και οι τρεις κλάσεις της οντολογίας Β θα πρέπει να ταιριάξουν με την κλάση Vehicles της Α, και τελικώς, τα στιγμιότυπα και των τριών να αντιγραφούν στην Vehicles. Μπορεί δηλαδή, ο δημιουργός της οντολογίας πηγής να έχει προχωρήσει σε λεπτομερείς υποπεριπτώσεις μιας περίπτωσης την οποία έχουμε καλύψει με ένα στοιχείο της οντολογίας προϊόντος. Το αντίθετο δεν μπορεί να ισχύσει. Αν ίσχυε, θα αντιγραφόταν μια τιμή πολλαπλές φορές σε πολλαπλά σημεία στην οντολογία, πράγμα που δεν μας είναι χρήσιμο, δυσκολεύει την ανάλυση των πληροφοριών και οδηγεί σε άσκοπη κατανάλωση χώρου. Για το παραπάνω παράδειγμα, αν έπρεπε να κάνουμε ανάθεση από την οντολογία Α στη Β, και επιτρέπαμε τα 1:Ν ταιριάσματα, θα αντιγράφονταν τα στιγμιότυπα της Vehicles κλάσης της Α οντολογίας και στις τρείς κλάσεις της οντολογίας Β. Ακόμη, αν παίρναμε το πιθανότερο ταίριασμα για κάθε στοιχείο της οντολογίας προϊόντος, αντί της πηγής, στην περίπτωση που ταίριαζαν δύο, ή παραπάνω στοιχεία της πηγής που ταιριάζουν (έχουν τιμή ομοιότητας πάνω απ' το όριο) με ένα στοιχείο της PO, θα επιλέγονταν ένα ταίριασμα απ' τα δύο και έτσι θα χάναμε τα στιγμιότυπα του άλλου ή θα πήγαιναν σε λάθος θέση. Έτσι στο παράδειγμα μας, με ανάθεση από την Β στην Α, Θα παίρναμε το πιθανότερο ταίριασμα της κλάσης Vehicles της Α. Το ταίριασμα αυτό είναι με την κλάση Vehicles της Β (αφού έχουν τιμή ομοιότητας 1). Τελικώς θα αντιγράφονταν επιτυχώς τα στιγμιότυπα της κλάσης Vehicles του Β στην κλάση Vehicles του Α, αλλά τα στιγμιότυπα των κλάσεων Cars και Motorcycle θα χάνονταν. Επιλέγουμε λοιπόν Ν ταιριάσματα για τα Ν στοιχεία της οντολογίας πηγής. Έπειτα, ελέγχουμε αν η τιμή ομοιότητας για κάθε ταίριασμα είναι μεγαλύτερη ή ίση από το άνω όριο που έχει δοθεί. Αν είναι, αποθηκεύουμε το ζεύγος των στοιχείων. Αν όχι, απορρίπτουμε το ταίριασμα. Τα ταιριάσματα αυτά τα απορρίπτουμε, αφού έτσι αποφεύγουμε πολλά λανθασμένα ταιριάσματα. Για να τηρηθεί η πληρότητα της ανάθεσης (αφού με τις απορρίψεις μπορεί να έχουμε λιγότερα από Ν ταιριάσματα, κάποιο στοιχείο της οντολογίας πηγής να μην ταιριάζει με κάποιο στοιχείο της οντολογίας προϊόντος και κατά την ανάθεση να χαθούν οι τιμές του αφού δεν υπάρχει κάπου να ανατεθεί), αργότερα, θα αποθηκευτούν οι πληροφορίες των στοιχείων που δεν έχουν ταίριασμα στη Default Property. Έτσι λοιπόν, ο αλγόριθμος αποθηκεύει σε αυτή τη φάση τα πιθανότερα ταιριάσματα κλάσεων και ιδιοτήτων για μεταγενέστερη χρήση, και απορρίπτει τα υπόλοιπα.

Ανάθεση στιγμιότυπων: Το επόμενο βήμα του αλγόριθμου είναι να κάνει μια αναζήτηση στα ταιριάσματα που βρέθηκαν κατά τη φάση της αρχικοποίησης, και συγκεκριμένα στα στοιχεία της οντολογίας προϊόντος, για να βρει αν υπάρχει καμία

κλάση που να είναι υποκλάση της ProductCategory (βλέπε Παράρτημα Β). Αν βρεθεί, τότε αυτή η κλάση ή κλάσεις, αναφέρονται στα νέα προϊόντα που θέλουμε να προσθέσουμε στην οντολογία μας. Έτσι λοιπόν θα πάρουμε όλα τα στιγμιότυπα των κλάσεων αυτών, τα οποία θα προστεθούν αργότερα σαν νέα στιγμιότυπα της κλάσης Product (Παράρτημα Β). Αν δεν βρεθούν τέτοιες κλάσεις, ή αν δεν έχουν κανένα στιγμιότυπο αυτές, η εκτέλεση του αλγόριθμου θα ολοκληρωθεί εδώ. Για παράδειγμα, θεωρούμε και πάλι ανάθεση από την οντολογία Β στην Α της εικόνας 6.2. Ακόμη, θεωρούμε πως η κλάση Vehicles της Β έχει μια ιδιότητα αντικειμένων, την Manufacturer. Αυτή η ιδιότητα ενδέχεται να έχει σαν τιμές κάποια στιγμιότυπα της οντολογίας, αφού έχει πεδίο τιμών αντικείμενα. Τα στιγμιότυπα λοιπόν των κλάσεων Vehicles, Cars και Motorcycle, είναι στιγμιότυπα προϊόντων και θα αποθηκευτούν αφού ταιριάζουν με την υποκλάση Vehicles της Α. Τα στιγμιότυπα που είναι τιμές της ιδιότητας Manufacturer, αντίθετα, δεν ταιριάζουν με τις υποκλάσεις της ProductCategory και δεν θα αποθηκευτούν αφού αφορούν έμμεσα τα προϊόντα και όχι άμεσα. Σκοπός λοιπόν αυτής της φάσης είναι να βρεθούν όλα τα στιγμιότυπα, που αναφέρονται άμεσα σε προϊόντα, που περιέχει η οντολογία πηγής, και να εισαχθούν στην οντολογία προϊόντος. Το πώς θα εισαχθούν εξετάζεται παρακάτω.

Αλγόριθμος 8. Ανάθεση στιγμιοτύπων (γενικά).

```
//-------------------------------------------------------------------------------------------------------
//Get all products from source ontology and create new product instances for each of them
Algorithm AssignInstances
Input: ProductOntology, SourceOntology, SellerName
Output: Updated ProductOntology
Begin
    Get MatchedClasses()
    Get MatchedProperties()
    Foreach pair in MatchedClasses
        If pair.ProductOntologyClass is Subclass of ProductCategory class Then
            Get all instances from the SourceOntologyClass
            Foreach instance in Instances
                CreateNewProductInstance(Instance)
            EndIf
```

```
                    EndFor
                EndIf
            EndFor
End
//-------------------------------------------------------------------------------------------------------
```

Δημιουργία νέου στιγμιότυπου Προϊόντος: Για τη δημιουργία ενός νέου στιγμιότυπου προϊόντος χρησιμοποιούμε το στιγμιότυπο που εξάγαμε από την οντολογία πηγής στη προηγούμενη παράγραφο (αλγόριθμος 8).

Σαν πρώτο βήμα για τη δημιουργία, συγκρίνουμε το δοθέν στιγμιότυπο, με τα στιγμιότυπα που ήδη περιέχει η κλάση Product, για να βρεθεί αν το συγκεκριμένο προϊόν υπάρχει ήδη στην οντολογία μας. Η σύγκριση γίνεται με βάση το όνομα προϊόντος και το όνομα πωλητή. Το όνομα προϊόντος του δοθέν στιγμιότυπου είναι το ίδιο το όνομα του. Το όνομα προϊόντος ενός στιγμιότυπου της κλάσης Product το βρίσκουμε μέσω της ιδιότητας hasName. Το όνομα πωλητή του στιγμιότυπου της κλάσης Product το βρίσκουμε σαν τιμή της ιδιότητας hasSellerName. Αν δεν βρεθεί κανένα στιγμιότυπο που να έχει ίδια και τα δύο πεδία με το δοθέν στιγμιότυπο, δημιουργούμε ένα νέο στιγμιότυπο στη κλάση Product. Αυτό το στιγμιότυπο θα χρησιμοποιήσουμε στη συνέχεια για να εισάγουμε τιμές στις ιδιότητες του. Αν αντίθετα, με την αναζήτηση βρεθεί κάποιο στιγμιότυπο που να έχει ίδια και τα δύο πεδία, το χρησιμοποιούμε για να ανανεώσουμε τις τιμές του. Ο λόγος που χρησιμοποιούμε τον έλεγχο των δύο πεδίων, είναι γιατί μπορεί το ίδιο προϊόν να προσφέρεται από δύο διαφορετικές πηγές, με διαφορετικές πληροφορίες στο καθένα. Έτσι, θα πρέπει να αντιμετωπιστούν σαν δύο διαφορετικά προϊόντα. Για παράδειγμα, μπορεί μια αντιπροσωπεία να προσφέρει ένα αμάξι με μία τιμή, ενώ από μια άλλη αντιπροσωπεία να προσφέρεται με διαφορετική τιμή. Θα πρέπει να εμφανίζονται όλα τα προϊόντα. Μόνο αν έχουν το ίδιο όνομα (είναι το ίδιο προϊόν) και το ίδιο όνομα πωλητή (προσφέρεται από την ίδια πηγή) θεωρούμε ότι υπάρχει ήδη το προϊόν στην οντολογία μας και δεν χρειάζεται να δημιουργηθεί ένα καινούριο στιγμιότυπο. Σε αυτή την περίπτωση ανανεώνουμε τις τιμές του ήδη υπάρχον στιγμιότυπου. Αυτό γίνεται γιατί μπορεί να έχουν ανανεωθεί οι πληροφορίες που προσφέρει ο πωλητής για το συγκεκριμένο προϊόν, π.χ. να έχει αλλάξει η τιμή και θα πρέπει να κρατήσουμε μόνο τις τελευταίες τιμές.

Το όνομα του νέου στιγμιότυπου θα είναι της μορφής Product_ID, όπου στην θέση ID μπαίνει ο μοναδικός κωδικός. Με αυτό το τρόπο, αν υπάρχει πολλαπλές φορές ένα προϊόν, από διαφορετικούς πωλητές, εμφανίζεται πολλαπλές φορές και στην οντολογία μας, με διαφορετικό κωδικό. Σε αντίθετη περίπτωση θα έπρεπε να κρατήσουμε ένα προϊόν από ένα πωλητή και να απορρίψουμε τα άλλα των υπολοίπων, το οποίο είναι λάθος και μονομερές.

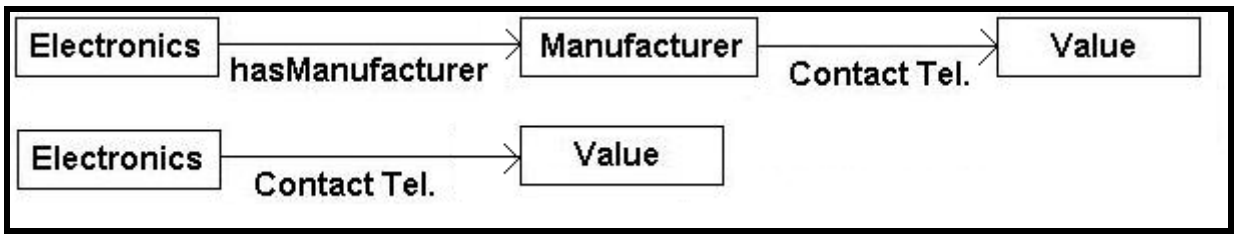
Ακόμα, στη περίπτωση που έχουμε δημιουργήσει ένα νέο στιγμιότυπο, και δεν ανανεώνουμε κάποιο παλιότερο, βάζουμε τιμές στις σταθερές ιδιότητες του, πριν προχωρήσουμε στην αντιγραφή των τιμών των ιδιοτήτων από το στιγμιότυπο. Οι ιδιότητες αυτές είναι οι hasPID όπου θέτουμε σαν τιμή τον κωδικό ID, hasName οπού θέτουμε σαν τιμή το όνομα του δοθέν στιγμιότυπου και τέλος την hasSellerName όπου θέτουμε σαν τιμή του τον πωλητή που έχει δοθεί προηγουμένως από το χρήστη.

Μέχρι τώρα λοιπόν, είμαστε στην περίπτωση όπου είτε έχει δημιουργηθεί ένα νέο στιγμιότυπο στην κλάση Product με τις τιμές των γενικών ιδιοτήτων του συμπληρωμένες και τις υπόλοιπες κενές προς συμπλήρωση, είτε υπάρχει ήδη ένα στιγμιότυπο με τις ιδιότητες του συμπληρωμένες, οι οποίες όμως θα ανανεωθούν με τις νέες τιμές των ιδιοτήτων. Σαν επόμενο βήμα παίρνουμε όλες τις ιδιότητες της κλάσης Product. Για κάθε μία από αυτές, παίρνουμε την αντίστοιχη της SO (matchProp). Αυτό γίνεται εξετάζοντας τις τα ταιριάσματα που είχαμε πάρει κατά το στάδιο της αρχικοποίησης. Όπως έχει φανεί και πιο πάνω, τα ταιριάσματα αυτά δεν είναι 1:1, δηλαδή υπάρχει η περίπτωση κάποια ιδιότητα της οντολογίας προϊόντος να μην ταιριάζει με κάποια ιδιότητα της οντολογίας πηγής. Έτσι εδώ παίρνουμε δύο περιπτώσεις:

Αν δεν βρεθεί ιδιότητα της SO που να ταιριάζει με την ιδιότητα του Product, δεν θα υπάρχει κάποια τιμή να αντιγραφεί σε αυτήν. Έτσι, τελειώνει αυτός ο κύκλος και εξετάζουμε την επόμενη ιδιότητα του Product.

Αν βρεθεί η αντίστοιχη ιδιότητα, τότε κάνουμε αναζήτηση για να βρούμε σε ποιό ακριβώς στιγμιότυπο ανήκει η συγκεκριμένη ιδιότητα matchProp. Το στιγμιότυπο αυτό το χρειαζόμαστε για να μπορέσουμε να εξάγουμε την τιμή της ιδιότητας αυτής, ώστε μετά να μπορέσουμε να την προσθέσουμε στην ιδιότητα του Product που ταιριάζει. Την αναζήτηση την κάνουμε, παρόλο που έχουμε το στιγμιότυπο του προϊόντος από την οντολογία πηγής, γιατί η ιδιότητα μπορεί να ανήκει στο στιγμιότυπο αυτό, αλλά μπορεί και να αναφέρεται έμμεσα σε αυτό χωρίς να του ανήκει. Δηλαδή, μπορεί το στιγμιότυπο προϊόντος να περιέχει μια ιδιότητα που έχει ως πεδίο τιμών ένα άλλο στιγμιότυπο το

οποίο με την σειρά του έχει την ιδιότητα που θέλουμε. Αυτό εξαρτάται από τη δομή της οντολογίας η οποία ποικίλλει αναλόγως με αυτόν που τη κατασκεύασε. Παρόλαυτα, είτε ανήκει άμεσα στο στιγμιότυπο, είτε έμμεσα, πρέπει να προστεθεί σαν ιδιότητα του νέου ή προϋπάρχοντος προϊόντος αφού αναφέρεται σε αυτό. Εδώ βλέπουμε πως η δομή των οντολογιών και το 'ύψος' που βρίσκονται οι τιμές δεν έχει σημασία για τη λειτουργία ανάθεσης τιμών, αντίθετα με την λειτουργία ταιριάσματος που πρέπει να ληφθεί υπ' όψιν η δομή.



Εικόνα 6.3. Παράδειγμα Οντολογιών.

Για παράδειγμα, θεωρούμε τις οντολογίες της εικόνας 6.3. Υποθέτουμε πως η SO είναι η πάνω, ενώ η κάτω η PO. Εδώ, επειδή οι δύο κλάσεις Electronics ταιριάζουν, παίρνουμε τα στιγμιότυπα της κλάσης της πάνω οντολογίας, και τα αντιγράφουμε στην κάτω. Έπειτα, κοιτάμε τις ιδιότητες της κλάσης Electronics στη κάτω οντολογία, δηλαδή την Contact Tel. Αυτή ταιριάζει με την Contact Tel. ιδιότητα της πάνω. Παρόλαυτα, για να μπορέσουμε να πάρουμε την τιμή της πρέπει να έχουμε το άμεσο στιγμιότυπο στο οποίο ανήκει. Εμείς έχουμε το στιγμιότυπο της Electronics κλάσης. Έτσι θα πρέπει να ταξιδέψουμε, από το αρχικό στιγμιότυπο, μέσω της hasManufacturer ιδιότητας στο αντίστοιχο στιγμιότυπο της Manufacturer κλάσης, και από κει, στην τιμή που θέλουμε μέσω της Contact Tel.

Έτσι η αναζήτηση ξεκινά από αυτό το δοθέν στιγμιότυπο, που ανήκει στην οντολογία πηγής. Αν το στιγμιότυπο αυτό έχει την matchProp ιδιότητα, η αναζήτηση τελειώνει και επιστρέφει το ίδιο το στιγμιότυπο. Αν δεν έχει τη συγκεκριμένη ιδιότητα, παίρνουμε όλες τις ιδιότητες του με πεδίο τιμών αντικείμενα, και παίρνουμε τα αντίστοιχα στιγμιότυπα στο πεδίο τιμών αυτών των ιδιοτήτων, τα οποία ελέγχουμε και αυτά με την σειρά τους αν έχουν την matchProp. Η αναζήτηση λοιπόν έχει αναδρομική λειτουργία. Την λειτουργία αυτή την χρησιμοποιούμε με το όνομα GetInstanceHavingThisProperty. Αν δεν βρεθεί

κανένα στιγμιότυπο που να περιέχει την ιδιότητα που θέλουμε, απορρίπτεται η ιδιότητα του Product και προχωράμε στην επόμενη.

Αφού ολοκληρωθεί η αναζήτηση επιτυχώς, έχει βρεθεί δηλαδή ένα στιγμιότυπο που περιέχει την matchProp ιδιότητα συνεχίζουμε στην ανάθεση των τιμών. Στην ανάθεση αυτή, παίρνουμε διάφορες περιπτώσεις:

Στη πρώτη περίπτωση, η ιδιότητα του Product που θέλουμε να προσθέσουμε την τιμή, έχει πεδίο τιμών αντικείμενα. Εδώ, αν έχει βρεθεί η ιδιότητα της πηγής που ταιριάζει με την ιδιότητα του Product, παίρνουμε πάλι άλλες δύο περιπτώσεις: την περίπτωση η matchProp να έχει πεδίο τιμών αντικείμενα, και την περίπτωση να έχει πεδίο τιμών δεδομένα. Σε αυτές τις περιπτώσεις κάνουμε ανάθεση τιμών από ιδιότητα με πεδίο τιμών αντικείμενα, σε ιδιότητα με πεδίο τιμών αντικείμενα πάλι, και ανάθεση από ιδιότητα με πεδίο τιμών δεδομένα σε ιδιότητα με πεδίο τιμών αντικείμενα, αντίστοιχα. Οι αναθέσεις αυτές περιγράφονται λεπτομερέστατα σε επόμενο εδάφιο.

Αν δεν έχει βρεθεί η ιδιότητα, δηλαδή η μεταβλητή matchProp είναι κενή, ο αλγόριθμος προσπαθεί να κάνει ανάθεση στιγμιότυπων μέσω του πεδίου τιμών της. Πιο συγκεκριμένα, παίρνει το έμμεσο και άμεσο πεδίο τιμών της ιδιότητας (τα οποία αφού βρισκόμαστε στην περίπτωση που έχει πεδίο τιμών αντικείμενα, θα είναι ένα σύνολο κλάσεων). Εδώ αξίζει να θυμίσουμε πως το άμεσο πεδίο τιμών μιας ιδιότητας είναι η κλάση που είναι ορισμένη ως πεδίο τιμών, ενώ το έμμεσο πεδίο τιμών είναι όλες οι υποκλάσεις της. Παίρνει λοιπόν το άμεσο και έμμεσο πεδίο τιμών της με την μορφή ενός συνόλου κλάσεων, και προσπαθεί να βρει όλες τις κλάσεις που ταιριάζουν με την κάθε κλάση του συνόλου αυτού. Αν δεν βρεθεί καμία κλάση, τερματίζει αυτή η ιδιότητα και περνάει στην επόμενη. Αν βρεθούν κάποιες κλάσεις, τότε παίρνει όλα τα στιγμιότυπα των κλάσεων αυτών για να τα εισάγει στην οντολογία μας. Αφού όμως παίρνουμε όλα τα στιγμιότυπα των κλάσεων αυτών, μπορεί να μην έχουν όλα κάποια σχέση με το συγκεκριμένο προϊόν που θέλουμε να εισάγουμε και να αφορούν άλλα προϊόντα που έχουν ήδη εισαχθεί ή θα εισαχθούν στη συνέχεια. Έτσι λοιπόν πρέπει να ξεκαθαρίσουμε τα στιγμιότυπα και να κρατήσουμε μόνο αυτά που σχετίζονται άμεσα ή έμμεσα με το προϊόν του οποίου τις πληροφορίες προσπαθούμε να εισάγουμε. Για να γίνει αυτό, χρησιμοποιούμε μια άλλη λειτουργία, την GetRelatedInstance. Ουσιαστικά αυτή η λειτουργία, όπως και η GetInstanceHavingThisProperty, έχει αναδρομικό χαρακτήρα και ψάχνει όλα τα στιγμιότυπα που σχετίζονται με το στιγμιότυπο προϊόντος της οντολογίας πηγής που θέλουμε να εισάγουμε. Αρχικά, ψάχνει στο σύνολο των στιγμιότυπων που βρέθηκαν πριν, αν περιέχει το δοθέν στιγμιότυπο θα εισάγουμε. Αν υπάρχει, τότε το

επιστρέφει. Αν δεν βρεθεί, τότε παίρνει όλες τις ιδιότητες του, που έχουν σαν πεδίο τιμών αντικείμενα, έπειτα παίρνει τις τιμές τους (οι οποίες θα είναι στιγμιότυπα αφού τα πεδία τιμών είναι αντικείμενα) και πραγματοποιεί τον ίδιο έλεγχο αναδρομικά. Όταν τελειώσει αυτή η διαδικασία είτε θα έχουμε το στιγμιότυπο που πρέπει να εισαχθεί ως τιμή της ιδιότητας του νέου προϊόντος, είτε θα τερματιστεί αυτός ο κύκλος, και θα περάσουμε σε επόμενη ιδιότητα.

Έτσι λοιπόν μένει να βρούμε σε ποια ακριβώς κλάση θα πρέπει να εισαχθεί το στιγμιότυπο αυτό. Αυτό μπορούμε εύκολα να το βρούμε αν πάρουμε την κλάση στην οποία ανήκει το στιγμιότυπο (κλάση της οντολογίας πηγής) και, μέσω των αποθηκευμένων ταιριασμάτων, την αντιστοιχίσουμε στην κλάση προϊόντων. Εδώ, είμαστε σίγουροι πως θα έχει κλάση-ταίρι και δεν ελέγχουμε την περίπτωση που δεν έχει ταίρι, αφού το στιγμιότυπο το οποίο ανήκει στη κλάση το πήραμε μέσω της αντίστροφης διαδικασίας. Κάποιος θα μπορούμε να αναρωτηθεί, γιατί αυτό το μπρος-πίσω στις κλάσεις και τα ταίρια τους: αφού πήραμε το στιγμιότυπο που θα εισάγουμε χρησιμοποιώντας, έχοντας σα δεδομένες τις κλάσεις την οντολογίας προϊόντος γιατί θα πρέπει να αναζητήσουμε σε ποια κλάση αντιστοιχεί το στιγμιότυπο; Αυτό γίνεται γιατί για να πάρουμε το στιγμιότυπο χρησιμοποιήσαμε ένα σύνολο κλάσεων (του πεδίου τιμών της κλάσης) χωρίς να ελέγχουμε κάθε κλάση ξεχωριστά, και δεν ξέρουμε σε ποια συγκεκριμένη από το όλο σύνολο θα πρέπει να αντιστοιχίσουμε το στιγμιότυπο. Χρησιμοποιήσαμε τις κλάσεις σαν σύνολο, για να πάρουμε ένα σύνολο στιγμιότυπων, από το οποίο ξεχωρίσαμε ένα. Δεν μπορούμε να ξέρουμε σε ποια κλάση πρέπει να εισαχθεί αυτό εκ των προτέρων, και έτσι, τη βρίσκουμε μέσω της κλάσης από την οποία προέρχεται το στιγμιότυπο. Το ταίριασμα του θα είναι και η κλάση που ψάχνουμε.

Τέλος, γίνεται η ανάθεση χρησιμοποιώντας τη λειτουργία ανάθεσης στιγμιότυπου σε κλάση, η οποία ουσιαστικά αντιγράφει το δοθέν στιγμιότυπο σε ένα νέο στη κλάση αυτή, κατά ένα τρόπο παρόμοιο με αυτόν για την δημιουργία ενός νέου προϊόντος που εξετάζουμε τώρα. Το πως το κάνει, θα εξεταστεί λεπτομερέστερα στη συνέχεια (αλγόριθμος 10). Η λειτουργία αυτή, επιστρέφει το στιγμιότυπο της κλάσης το οποίο μόλις δημιουργήθηκε. Έτσι μένει να αναθέσουμε το στιγμιότυπο αυτό ως τιμή της ιδιότητας της κλάσης Product που εξετάζουμε αυτή τη στιγμή, και ο σκοπός μας θα έχει επιτευχθεί.

Πριν όμως πραγματοποιήσουμε την ανάθεση αυτή, πρέπει να ελέγξουμε αν το στιγμιότυπο αυτό υπάρχει ήδη ως τιμή της ιδιότητας (προφανώς με παλαιότερες και ίσως ξεπερασμένες τιμές στις δικές του ιδιότητες και να το διαγράψουμε ώστε να μπορεί να εισαχθεί το νέο). Τον έλεγχο αυτό τον πραγματοποιούμε παίρνοντας όλες τις τιμές της

ιδιότητας και συγκρίνοντας τα ονόματα τους με το όνομα του νέου στιγμιότυπου. Αν βρεθεί κάποιο στιγμιότυπο από αυτή την αναζήτηση, διαγράφεται ώστε να μπορέσει να περαστεί το καινούριο. Κάποιος θα μπορούσε να αναρωτηθεί γιατί το διαγράφουμε και ξαναπερνάμε το καινούριο, και δεν ανανεώνουμε απλά τις τιμές των ιδιοτήτων του. Το στιγμιότυπο αυτό θα μπορούσε να είναι τιμή μιας ιδιότητας με πεδίο τιμών αντικείμενα και μέσω της διαγραφής να χαθεί η τιμή της. Αυτό όμως δεν υφίσταται αφού από το στιγμιότυπο αυτό ξεκινάμε την ανάθεση. Πιο συγκεκριμένα, αν φανταστούμε ένα νοητό δέντρο με τις διασυνδέσεις των στιγμιότυπων μεταξύ τους, το στιγμιότυπο προϊόντος θα βρίσκεται στην κορυφή, χωρίς «γονείς», αφού αυτό εισάγουμε πρώτο στην οντολογία προϊόντος και μετά ακολουθούμε τις ιδιότητες του για να εισάγουμε τιμές. Άρα, αποκλείεται να υπάρχει κάποια ιδιότητα που να προηγείται, και να έχει ως τιμή το στιγμιότυπο αυτό. Επίσης, αν ανανεώναμε απλά τις τιμές χωρίς το σβήσιμο-εισαγωγή, θα έπρεπε να πραγματοποιήσουμε το διπλάσιο κόπο (χρόνο) αφού θα έπρεπε να κάνουμε όλα τα προηγούμενα βήματα που κάναμε για να γεμίσουμε το νέο στιγμιότυπο με τιμές, για να ελέγξουμε την ύπαρξη ή μη των τιμών αυτών. Ο λόγος λοιπόν που δεν ανανεώνουμε και διαγράφουμε-εισάγουμε είναι η μείωση της πολυπλοκότητας, αφού θα έχουμε το ίδιο αποτέλεσμα και στις δύο περιπτώσεις. Έτσι επιλέγουμε απλά να διαγράψουμε το παλιό και να εισάγουμε το καινούριο που ούτως ή άλλως υπάρχει. Τέλος, μένει μόνο να κάνουμε την ανάθεση του νέου στιγμιότυπου στην κλάση της οντολογίας προϊόντος που ανήκει (την έχουμε βρει σε προηγούμενο βήμα).

Αφοί λοιπόν εξετάσαμε την περίπτωση που η ιδιότητα του Product που θέλουμε να προσθέσουμε την τιμή, έχει πεδίο τιμών αντικείμενα, σειρά έχει και η περίπτωση που έχει ως πεδίο τιμών δεδομένα. Σε αυτή την περίπτωση, αν δεν υπάρχει κάποια ιδιότητα που να ταιριάζει με αυτήν (δηλαδή η matchProp είναι κενή) τελειώνει αυτός ο κύκλος και περνάμε στην επόμενη ιδιότητα του Product. Σε αντίθετη περίπτωση, παίρνουμε πάλι, δύο υποπεριπτώσεις: η πρώτη είναι η περίπτωση που η matchProp έχει πεδίο τιμών αντικείμενα, όπου κάνουμε ανάθεση από ιδιότητα με πεδίο τιμών αντικείμενα, σε ιδιότητα με πεδίο τιμών δεδομένα. Στη δεύτερη περίπτωση, όπου η matchProp έχει πεδίο τιμών δεδομένα, κάνουμε ανάθεση από ιδιότητα με πεδίο τιμών δεδομένα, σε ιδιότητα με πεδίο τιμών δεδομένα. Οι αναθέσεις αυτές εξετάζονται στη συνέχεια.

Αλγόριθμος 9. Δημιουργία νέου στιγμιότυπου προϊόντος.

//----------

// Create new Product Instance and add its property values

Function CreateNewProductInstance

Input: Product Instance

Output: Updated ProductOntology

Begin

If the given product exists with the given seller

Get the Product_Instance

Else

Create Product ID

Fill the generic properties // hasName, hasPID, hasSellerName

Create new Product_Instance

Get new Product_Instance

EndIf

ForEach property in ProductProperties of Product_Instance

Get the best property match from the PropertySimilarityMatrix

If SourceMatchedProperty != null Then

GetSourceInstanceHavingThisProperty(SourceMatchedProperty)

If SourceInstance != null Then

If property is ObjectProperty Then

If SourceMatchedProperty is ObjectProperty Then

ObjectProp2ObjectPropAssign()

Else

DatatypeProp2ObjectPropAssign()

EndIf

Else

If SourceMatchedProperty is ObjectProperty Then

ObjectProp2DatatypePropAssign()

Else

DatatypeProp2DatatypePropAssign()

EndIf

EndIf

EndIf

Else If property is ObjectProperty Then

GetPropertyRanges()

Get matched classes for all ranges

Execute AssignInstance2Class() for all the matched classes containing instances

EndIf

EndFor

End

//----------

Ανάθεση στιγμιότυπου σε κλάση: Εδώ, για να γίνει η ανάθεση, θεωρούμε σαν δεδομένα ένα στιγμιότυπο της SO και μια κλάση της PO. Σκοπός αυτής της λειτουργίας είναι να δημιουργήσει ένα νέο στιγμιότυπο στην κλάση, το οποίο να έχει όλες τις πληροφορίες (όνομα, τιμές ιδιοτήτων) που έχει και το δοθέν στιγμιότυπο. Το ποια είναι τα στιγμιότυπα και οι κλάσεις και πως ακριβώς τα επιλέγουμε φαίνεται στα σημεία όπου καλείται η λειτουργία αυτή. Η διαδικασία ανάθεσης είναι παρόμοια με τη δημιουργία νέου προϊόντος που εξηγήθηκε παραπάνω (αλγόριθμος 9). Όπως και στη δημιουργία νέου προϊόντος, δημιουργούμε ένα νέο στιγμιότυπο (αφού πρώτα ελέγξουμε αν ήδη υπάρχει στην οντολογία όπου αντί να το δημιουργήσουμε το ενημερώνουμε). Έπειτα, περνάμε όλες τις ιδιότητες του τις οποίες γεμίζουμε με τις τιμές των ιδιοτήτων της οντολογίας πηγής, που ταιριάζουν σύμφωνα με την διαδικασία ταιριάσματος. Αν κάποια ιδιότητα της οντολογίας προϊόντος δεν ταιριάζει με καμία και περιέχει πληροφορία, η πληροφορία αυτή εισάγεται στην ιδιότητα προεπιλογής του προϊόντος στο οποίο αναφέρεται το στιγμιότυπο αυτό (όταν αναφέρουμε το προϊόν στο οποίο αναφέρεται ένα στιγμιότυπο εννοούμε το στιγμιότυπο της κλάσης Product το οποίο συνδέεται με το στιγμιότυπο αυτό).

Αρχικά λοιπόν, θα δημιουργηθεί το νέο στιγμιότυπο. Πριν το δημιουργήσουμε όμως, πρέπει να ελέγξουμε για τυχόν ύπαρξή του ήδη στην οντολογία προϊόντος. Ο λόγος είναι επειδή κάποια προϊόντα μπορεί να έχουν κοινό κάποιο χαρακτηριστικό-αντικείμενο π.χ. τον κατασκευαστή. Έτσι, δύο τελείως διαφορετικά προϊόντα, με διαφορετικές πληροφορίες, μπορεί να περιέχουν μια ίδια τιμή-στιγμιότυπο σε μια ιδιότητα τους. Η τιμή αυτή, αφού θα είναι ακριβώς ίδια και στα δύο στιγμιότυπα, δεν πρέπει να υπάρχει πολλαπλές φορές (άσκοπη σπατάλη χώρου καθώς και απαγόρευση από τον ορισμό των οντολογιών να υπάρχουν δύο ακριβώς ίδια στιγμιότυπα σε μια οντολογία), αλλά μια και να συνδέεται με τα δύο στιγμιότυπα μέσω των ιδιοτήτων τους. Για να ελέγξουμε λοιπόν για την ύπαρξη του στιγμιότυπου που θέλουμε να προσθέσουμε, παίρνουμε το όνομα του δοθέντος στιγμιότυπου και ελέγχουμε αν υπάρχει στην οντολογία προϊόντος. Αν δεν

βρεθεί, το δημιουργούμε με το ίδιο όνομα. Εδώ ξέρουμε πως σίγουρα υπάρχει ένα στιγμιότυπο με το όνομα του δοθέντος στιγμιότυπου (είτε υπήρχε ήδη, είτε το δημιουργούμε) οπότε το αποθηκεύουμε σε μια μεταβλητή έστω Β για μετέπειτα χρήση.

Το επόμενο βήμα είναι να πάρουμε όλες τις ιδιότητες του δοθέντος στιγμιότυπου που έχουν τιμές (κάποιες από τις ιδιότητες του μπορεί να έχουν αφεθεί κενές είτε επίτηδες είτε λόγω έλλειψης πληροφοριών και αφού ουσιαστικά ο σκοπός μας είναι η αντιγραφή των πληροφοριών, δεν μας χρησιμεύουν σε τίποτα και ελέγχουμε μόνο όσες έχουν τιμές). Για κάθε μια από τις ιδιότητες του δοθέντος στιγμιότυπου που περιέχουν κάποια πληροφορία γίνεται ο εξής κύκλος:

Παίρνουμε την αντίστοιχη ιδιότητα από την PO που ταιριάζει με αυτήν, σύμφωνα με τα αποθηκευμένα ταιριάσματα που περιγράψαμε πιο πάνω. Αν δεν υπάρχει αυτή η ιδιότητα (αν δηλαδή η ιδιότητα απ' την οποία θα αντιγραφεί η τιμή δεν έχει ταιριάξει με καμία) τότε χρησιμοποιούμε την ιδιότητα προεπιλογής. Αυτή θα είναι η ιδιότητα που θα γίνει η αντιγραφή της τιμής.

Μέχρι τώρα έχουμε την ιδιότητα του δοθέντος στιγμιότυπου, που περιέχει την τιμή που θέλουμε, καθώς και την ιδιότητα στην οποία θα αντιγραφεί-ανανεωθεί η νέα τιμή. Το επόμενο βήμα λοιπόν είναι να πάρουμε περιπτώσεις αναλόγως με το πεδίο τιμών των ιδιοτήτων, όπως ακριβώς και στην δημιουργία νέου προϊόντος. Αν λοιπόν, η δοθείσα ιδιότητα έχει πεδίο τιμών αντικείμενα, ενώ η ιδιότητα προς επικόλληση της νέας τιμής έχει πεδίο τιμών δεδομένα, κάνουμε '*ανάθεση από ιδιότητα με πεδίο τιμών αντικείμενα σε ιδιότητα με πεδίο τιμών δεδομένα'*, αν η δοθείσα έχει πεδίο τιμών αντικείμενα και η άλλη αντικείμενα, κάνουμε '*ανάθεση από ιδιότητα με πεδίο τιμών αντικείμενα σε ιδιότητα με πεδίο τιμών αντικείμενα'* , αν η δοθείσα έχει πεδίο τιμών δεδομένα και η άλλη αντικείμενα, κάνουμε '*ανάθεση από ιδιότητα με πεδίο τιμών δεδομένα σε ιδιότητα με πεδίο τιμών αντικείμενα*', αν η δοθείσα έχει πεδίο τιμών δεδομένα και η άλλη δεδομένα, κάνουμε '*ανάθεση από ιδιότητα με πεδίο τιμών δεδομένα σε ιδιότητα με πεδίο τιμών δεδομένα'* . Οι αναθέσεις αυτές, περιγράφονται παρακάτω.

Αλγόριθμος 10. Ανάθεση ενός συγκεκριμένου στιγμιοτύπου σε μια συγκεκριμένη κλάση.

***//*----------**

***//*assign an instance to a class along with their properties**

Function AssignInstance2Class

Input: Instance, Class

```
Output: Updated ProductOntology
Begin
    Name= anInstance.name
    If  Instance exists in ProductOntology Then
        If the seller is not the same as the SourceInstance Then
            Change the new Instance name to InstanceName+SellerName
            Create the new instance
        EndIf
    Else
        Create new Instance()
    EndIf
    Get New Instance()
    Foreach property in InstanceProperties
        Get the best property match from the PropertySimilarityMatrix
        If property is ObjectProperty Then
            If SourceMatchedProperty is ObjectProperty Then
                ObjectProp2ObjectPropAssign()
            Else
                ObjectProp2DatatypePropAssign()
            EndIf
        Else
            If SourceMatchedProperty is ObjectProperty Then
                DatatypeProp2ObjectPropAssign()
            Else
                DatatypeProp2DatatypePropAssign()
            EndIf
        EndIf
    EndFor
    Update ProductOntology with the new Instance
End
//-------------------------------------------------------------------------------------------------------------
```

Ανάθεση από ιδιότητα με πεδίο τιμών δεδομένα σε ιδιότητα με πεδίο τιμών δεδομένα: Στην ανάθεση αυτή, όπως και σε κάθε ανάθεση ιδιοτήτων, διαβάζουμε μια

τιμή από μια ιδιότητα που ανήκει στην οντολογία πηγής(ιδιότητα αντιγραφής), και την αντιγράφουμε σε μια ιδιότητα από την οντολογία προϊόντος(ιδιότητα επικόλλησης). Όσον αφορά το ποιες είναι αυτές οι ιδιότητες και πως επιλέχθηκαν εξηγείται στα σημεία που χρησιμοποιούμε αυτή την ανάθεση. Σε αυτή την περίπτωση θα πρέπει να διαβάσουμε την τιμή της ιδιότητας αντιγραφής και να την εισάγουμε στην ιδιότητα επικόλλησης, αφού πραγματοποιήσουμε τις κατάλληλες μετατροπές στον τύπο της τιμής.

Έτσι, αρχικά, διαβάζουμε την τιμή που περιέχει η ιδιότητα αντιγραφής. Αν περιέχει μια έγκυρη τιμή, παίρνουμε δύο περιπτώσεις: την περίπτωση που η ιδιότητα επικόλλησης είναι και ιδιότητα προεπιλογής, καθώς και την αντίθετη περίπτωση. Ο λόγος που χωρίζουμε αυτές τις δύο περιπτώσεις είναι ο εξής. Η ιδιότητα προεπιλογής μπορεί να πάρει πολλαπλές τιμές για ένα στιγμιότυπο, αφού δέχεται τιμές από πολλές ιδιότητες. Αντίθετα, οι υπόλοιπες ιδιότητες μπορούν να πάρουν αυστηρά μια τιμή. Έτσι αντιμετωπίζεται διαφορετικά στις δύο περιπτώσεις το ενδεχόμενο να πρέπει να εισαχθεί μια τιμή ενώ υπάρχει ήδη μια τιμή στην ιδιότητα επικόλλησης.

Στην περίπτωση που η ιδιότητα επικόλλησης είναι και ιδιότητα προεπιλογής, η τιμή που θα εισαχθεί δεν είναι ακριβώς η τιμή που αντιγράφηκε. Θα πρέπει να περιέχει παραπάνω πληροφορίες όπως την κλάση από την οποία προήλθε καθώς και την ιδιότητα στην οποία ανήκει. Σε αντίθετη περίπτωση θα έχουμε σκόρπιες πληροφορίες χωρίς καμία χρήση. Έτσι για παράδειγμα, μπορεί να θέλουμε να εισάγουμε τιμές που αντιστοιχούν σε διεύθυνση καταστήματος και διεύθυνση κατασκευαστή. Αν απλά βάλουμε τις τιμές θα υπάρχουν δύο διευθύνσεις που δεν θα ξέρουμε σε τι αναφέρονται και έτσι θα μας είναι άχρηστες. Θα πρέπει λοιπόν να προσθέσουμε και παραπάνω πληροφορίες. Στο κώδικα μας η τιμή που θα προσθέσουμε θα είναι της μορφής Κλάση_Ιδιότητα:Τιμή. Για το παράδειγμά μας οι τιμές που θα εισαχθούν θα είναι Κατάστημα_Διεύθυνση: *ΤιμήΔιεύθυνσης1* και Κατασκευαστής_Διεύθυνση:*ΤιμήΔιεύθυνσης2* αντίστοιχα. Έτσι θα περιέχουν αρκετές πληροφορίες ώστε να ξέρουμε που αναφέρεται η τιμή αυτή.

Πριν εισαχθεί η νέα τιμή θα πρέπει να ελέγξουμε αν υπάρχει ήδη τιμή με ίδια προέλευση με την τιμή. Την προέλευση την ελέγχουμε κοιτάζοντας τις πρώτες δύο πληροφορίες της τιμής που θα εισάγουμε δηλαδή Κλάση_Ιδιότητα. Έτσι αν προϋπάρχει κάποια τιμή που περιέχει τις ίδιες πληροφορίες, σημαίνει πως έχει προηγηθεί εισαγωγή από αυτή την ιδιότητα με παλιότερη τιμή. Διαγράφουμε λοιπόν την προϋπάρχουσα και προσθέτουμε την καινούρια.

Στην αντίθετη περίπτωση όπου η ιδιότητα επικόλλησης δεν είναι ιδιότητα προεπιλογής, ελέγχουμε αρχικά αν έχει προηγούμενη τιμή η ιδιότητα ή είναι κενή. Αν υπάρχει προηγούμενη τιμή, παίρνουμε δύο υποπεριπτώσεις:

- **Αν κάνουμε ανανέωση τιμών**: Σε αυτή τη περίπτωση αφαιρούμε όλες τις προηγούμενες τιμές, και προσθέτονται οι νεότερες.
- **Αν προσθέτουμε τώρα τιμές στις ιδιότητες για πρώτη φορά:** Εδώ, κατά πάσα πιθανότητα η τιμή που βρέθηκε είναι από άλλη ιδιότητα από ότι η νέα. Αυτό συμβαίνει γιατί, όπως εξηγήθηκε και στην εισαγωγή του αλγορίθμου, μπορεί πολλαπλές ιδιότητες της οντολογίας πηγής να ταιριάζουν με μια ιδιότητα της οντολογίας προϊόντος. Έτσι μπορεί να έχει εισαχθεί μια τιμή, και αργότερα να πρέπει να εισαχθεί μια άλλη διαφορετική τιμή από την πρώτη. Αφού όμως περιέχουν διαφορετικές πληροφορίες δεν πρέπει να χάσουμε καμία από αυτές και έτσι ουσιαστικά ενώνονται ώστε να σχηματίσουν ένα στιγμιότυπο με όνομα τις δύο τιμές δεδομένων. Μόνο αν στο προυπάρχον στιγμιότυπο η ονομασία του περιέχει την νέα τιμή που θέλουμε να εισαχθεί συμπεράνουμε πως γίνεται ανανέωση με νέες τιμές διαγράφεται η παλιά τιμή για να εισαχθεί η καινούρια. Παρόλα αυτά, όπως είπαμε μόνο στην ιδιότητα προεπιλογής επιτρέπουμε να έχει πολλαπλές τιμές. Γι' αυτό, σε όλες τις άλλες περιπτώσεις ενσωματώνουμε όλες τις τιμές σε μια. Έτσι σε αυτή τη περίπτωση, ελέγχουμε αρχικά αν στην ήδη υπάρχουσα τιμή υπάρχει σαν υπό-συμβολοσειρά η νέα τιμή. Αν υπάρχει, δεν υπάρχει λόγος να την ξαναπροσθέσουμε και έτσι τερματίζεται η λειτουργία αυτή. Αν δεν υπάρχει, ενώνουμε την παλιά με την νέα κατά τη μορφή: ΠαλιάΤιμή_ΝέαΤιμή.

Αν η ιδιότητα επικόλλησης δεν περιέχει ήδη τιμή απλά προσθέτουμε τη νέα τιμή.

Να συμπληρώσουμε πως για την εξαγωγή και εισαγωγή των τιμών πρέπει να γίνουν και οι κατάλληλες μετατροπές μεταξύ των ειδών δεδομένων. Τα δεδομένα μπορεί να έχουν μορφή ακεραίου αριθμού (integer), δεκαδικού αριθμού (float), συμβολοσειράς (string), λογικής τιμής (Boolean) ή ειδική μορφή για ημερομηνίες, ώρα κ.λ.π. Για να γίνουν οι μετατροπές, χρησιμοποιούμε ένα ενδιάμεσο στάδιο αυτό της συμβολοσειράς. Κατά την ανάγνωση των τιμών σε όποια μορφή και αν είναι τις μετατρέπουμε σε συμβολοσειρά. Αντίστοιχα, κατά την εισαγωγή των τιμών σε μια ιδιότητα πρέπει να μετατρέψουμε την τιμή, από συμβολοσειρά, στη μορφή που είναι ορισμένη η ιδιότητα. Αυτό το ενδιάμεσο στάδιο υπάρχει γιατί π.χ. δεν μπορεί να μετατραπεί ακέραιος σε ημερομηνία ενώ με το ενδιάμεσο στάδιο αυτό το πρόβλημα εξαλείφεται. Αν παρόλα αυτά υπάρξει πρόβλημα με

τις μετατροπές, και δεν μπορέσει να εισαχθεί σωστά η τιμή, εισάγεται και πάλι στην ιδιότητα προεπιλογής για να μην χαθεί.

Ανάθεση από ιδιότητα με πεδίο τιμών δεδομένα σε ιδιότητα με πεδίο τιμών αντικείμενα: Για να γίνει ανάθεση από ιδιότητα με πεδίο τιμών δεδομένα (ιδιότητα αντιγραφής) σε ιδιότητα με πεδίο τιμών αντικείμενα (ιδιότητα επικόλλησης) θα πρέπει ουσιαστικά να δημιουργηθεί ένα νέο στιγμιότυπο ως τιμή της ιδιότητας επικόλλησης με κενές ιδιότητες χρησιμοποιώντας την τιμή δεδομένων ως όνομα του στιγμιότυπου. Οι ιδιότητες του αναγκαστικά θα μείνουν κενές αφού η ιδιότητα αντιγραφής δεν περιέχει παραπάνω δεδομένα από την τιμή η οποία θα χρησιμοποιηθεί στο όνομα του νέου στιγμιότυπου. Διαβάζουμε λοιπόν αρχικά την τιμή δεδομένων την οποία όπως ειπώθηκε και παραπάνω την έχουμε μετατρέψει σε συμβολοσειρά.

Πριν γίνει η εισαγωγή, θα πρέπει να ελέγξουμε αν υπάρχει ήδη κάποια προυπάρχουσα τιμή στην ιδιότητα επικόλλησης αλλά και αν υπάρχει γενικά στην οντολογία κάποιο στιγμιότυπο με όνομα την τιμή αυτή (να θυμηθούμε πως ο ορισμός των οντολογιών δεν μας επιτρέπει να προσθέσουμε αντικείμενα που έχουν την ίδια ονομασία με ένα προυπάρχον, κάθε ονομασία στοιχείου πρέπει να είναι μοναδική).

Αν υπάρχει ήδη τιμή-στιγμιότυπο της ιδιότητας επικόλλησης, χωρίζουμε πάλι σε δύο υπό-περιπτώσεις:

- **Αν κάνουμε ανανέωση τιμών**: Σε αυτή τη περίπτωση αφαιρούμε όλες τις προηγούμενες τιμές, και προσθέτονται οι νεότερες.
- **Αν προσθέτουμε τώρα τιμές στις ιδιότητες για πρώτη φορά:** Εδώ, όπως και παραπάνω, για τους ίδιους λόγους πρέπει να συνδυάσουμε την παλιά τιμή με τη νέα. Πριν από αυτό όμως, ελέγχουμε αν η νέα είναι τμήμα της παλιάς. Αν είναι, τερματίζει η λειτουργία. Αν όχι, αλλάζουμε το όνομα του υπάρχον στιγμιότυπου σε ΠαλιόΟνομα_ΝέοΟνομα.

Όπως και παραπάνω, αν δεν υπάρχει ήδη στιγμιότυπο τιμή, προσθέτουμε τη καινούρια.

Σε όλες τις περιπτώσεις, είτε ανανεωθεί το στιγμιότυπο (το όνομα του), είτε δημιουργηθεί ένα νέο, πρέπει να υπάρχει έλεγχος για ύπαρξη του στιγμιότυπου αυτού στην οντολογία. Αυτό γίνεται γιατί δεν μπορούν να υπάρχουν δύο ίδια στιγμιότυπα με το ίδιο όνομα στην οντολογία. Αν υπάρξει κάποιο τέτοιο προϋπάρχον στιγμιότυπο, τελειώνει αυτός ο κύκλος του αλγορίθμου. Σε αντίθετη περίπτωση, συνεχίζεται η δημιουργία ή ανανέωση του στιγμιότυπου. Τέλος, το προσθέτουμε σαν τιμή της ιδιότητας.

Ανάθεση από ιδιότητα με πεδίο τιμών αντικείμενα σε ιδιότητα με πεδίο τιμών δεδομένα: Εδώ πρέπει να μετατρέψουμε τις πληροφορίες ενός στιγμιότυπου-αντικειμένου σε μια τιμή δεδομένου για να την εισάγουμε στην ιδιότητα δεδομένων. Η μετατροπή αυτή θα γίνει σε συμβολοσειρά, όπως και στις προηγούμενες αναθέσεις. Η μετατροπή αυτή πρέπει να γίνει έτσι ώστε να μην χαθεί καμία πληροφορία όπως οι τιμές των ιδιοτήτων του στιγμιότυπου. Διαβάζουμε λοιπόν το στιγμιότυπο, και φτιάχνουμε μια συμβολοσειρά με το όνομά του. Έπειτα παίρνουμε όλες τις ιδιότητες του. Για κάθε μια από αυτές παίρνουμε τις εξής περιπτώσεις: αν είναι ιδιότητα δεδομένων προσθέτουμε στην ήδη υπάρχουσα συμβολοσειρά μια άλλη της μορφής: *_ΌνομαΙδιότητας:ΤιμήΙδιότητας*. Αν η ιδιότητα είναι αντικειμένου, η συμβολοσειρά που θα προσθέσουμε θα είναι της μορφής *_ΌνομαΙδιότητας:ΕπόμενηΣυμβολοσειρά,* όπου με τον όρο *ΕπόμενηΣυμβολοσειρά* εννοούμε το αποτέλεσμα της ίδιας ακριβώς λειτουργίας που περιγράφεται με αρχικό στιγμιότυπο όμως το στιγμιότυπο-τιμή της ιδιότητας. Παρακάτω ακολουθεί ένα παράδειγμα επεξήγησης:

Έστω ότι έχουμε μια κλάση ‘αμάξι’, με ένα στιγμιότυπο FiatPunto, το οποίο πρέπει να εισαχθεί σε μια ιδιότητα δεδομένων. Η κλάση αυτή έχει δύο ιδιότητες. Η μια είναι αντικειμένου, ονομάζεται Κατασκευαστής, και έχει αντίστοιχα μια ιδιότητα δεδομένων, την Τηλέφωνο. Η άλλη, είναι ιδιότητα δεδομένων και ονομάζεται Κυβικά. Θεωρούμε τώρα πως για το στιγμιότυπο FiatPunto, η ιδιότητα Κατασκευαστής έχει τιμή το στιγμιότυπο Fiat που αυτό με την σειρά του έχει τιμή της ιδιότητας Τηλέφωνο 555, και η ιδιότητα του Κυβικά έχει τιμή 1300. Η συμβολοσειρά θα δημιουργηθεί ως εξής: Θα ξεκινήσουμε με το όνομα του στιγμιότυπου, δηλαδή η συμβολοσειρά θα είναι FiatPunto. Μετά, θα πάρουμε τις ιδιότητες του και θα ελέγξουμε αν είναι δεδομένων ή αντικειμένων. Η Κυβικά είναι δεδομένων, οπότε θα εισαχθεί με την μορφή _ Κυβικά:1300 και συνολικά η συμβολοσειρά θα έχει τώρα τη μορφή FiatPunto_ Κυβικά:1300. Έπειτα θα πάρει την επόμενη ιδιότητα του, την Κατασκευαστής, η οποία είναι δεδομένων. Έτσι θα κάνει αναδρομικά την ίδια λειτουργία: θα πάρει το όνομα του στιγμιότυπου Fiat, θα πάρει τις ιδιότητες του (την Τηλέφωνο), θα ελέγξει τι ιδιότητα είναι (δεδομένων), θα δημιουργήσει τη νέα συμβολοσειρά (_Τηλέφωνο:555) την οποία θα την προσθέσει στην ήδη υπάρχουσα δηλαδή την Κατασκευαστής και θα γίνει: Κατασκευαστής_Τηλέφωνο:555. Αφού ολοκληρωθεί η δεύτερη, θα επιστρέψει στην πρώτη και θα προσθέσει ότι βρήκε στην αρχική, οπότε τελικά θα έχουμε την FiatPunto_Κυβικά:1300_Κατασκευαστής_Τηλέφωνο:555. Με αυτό τον τρόπο θα διαβάσει

όλα τα δεδομένα του στιγμιότυπου και θα τα μετατρέψει σε μια συμβολοσειρά δίχως να χαθεί κανένα δεδομένο.

Από τη στιγμή που έχουμε τη νέα τιμή που θα εισαχθεί, τα επόμενα βήματα, για αυτή την ανάθεση είναι παρόμοια με την ανάθεση από ιδιότητα δεδομένων σε ιδιότητα δεδομένων. Και στις δύο περιπτώσεις τα βήματα είναι τα ίδια, το μόνο που αλλάζει είναι ο τρόπος που παίρνουμε τα δεδομένα. Πιο συγκεκριμένα χωρίζουμε πάλι σε δύο περιπτώσεις, την περίπτωση που η νέα τιμή πρέπει να εισαχθεί σε ιδιότητα προεπιλογής και την περίπτωση που πρέπει να εισαχθεί σε απλή ιδιότητα.

Στη πρώτη περίπτωση, θα προσθέσουμε στην αρχή της συμβολοσειράς που δημιουργήθηκε και την προέλευσή της, δηλαδή την κλάση και την ιδιότητα της οποίας τιμή είναι το στιγμιότυπο. Αν υπάρχει ήδη τιμή που έχει την ίδια προέλευση, αφαιρείται. Τέλος προστίθεται η νέα τιμή.

Στη δεύτερη περίπτωση, διαβάζεται η τιμή της ιδιότητας. Αν δεν υπάρχει, απλά προσθέτουμε αυτή που εξάγαμε στα προηγούμενα βήματα. Αν πάλι υπάρχει, ελέγχουμε αν είμαστε στην περίπτωση της ανανέωσης ή της εισαγωγής των τιμών. Στη πρώτη περίπτωση αφαιρούμε όποιες τιμές, ενώ στη δεύτερη συνδυάζουμε τις τιμές, παλιά και νέα αν φυσικά δεν υπάρχει ήδη η νέα σαν υπό-συμβολοσειρά της παλιάς.

Να σημειωθεί εδώ πως για την εισαγωγή, λόγω της μορφής των δεδομένων, δεν μπορεί να μετασχηματιστεί η νέα τιμή στο τύπο δεδομένων αν αυτός δεν είναι συμβολοσειρά. Έτσι, αν ο τύπος των δεδομένων της ιδιότητας επικόλλησης είναι κάτι διαφορετικό, δεν προστίθεται σε αυτήν η νέα τιμή αλλά πάει στην ιδιότητα προεπιλογής.

Ανάθεση από ιδιότητα με πεδίο τιμών αντικείμενα σε ιδιότητα με πεδίο τιμών αντικείμενα: Εδώ πρέπει να μεταφέρουμε τα δεδομένα από ένα στιγμιότυπο σ' ένα άλλο. Αρχικά διαβάζουμε την τιμή-στιγμιότυπο από την ιδιότητα αντιγραφής.

Έπειτα, διαβάζουμε τις τιμές-στιγμιότυπα που περιέχει η ιδιότητα επικόλλησης. Αν βρεθούν τιμές, διαγράφονται ώστε να μπορεί να εισαχθεί το νέο στιγμιότυπο.

Το επόμενο βήμα είναι να βρεθεί η κλάση στην οποία θα δημιουργηθεί το νέο στιγμιότυπο. Για να γίνει αυτό, πρέπει να πάρουμε τον τύπο του πεδίου τιμών της ιδιότητας επικολλήσεως, ο οποίος είναι μια κλάση (άμεσο πεδίο τιμών). Έπειτα παίρνουμε όλα τα έμμεσα πεδία τιμών, δηλαδή τις υπο-κλάσεις του άμεσου. Αυτό το κάνουμε γιατί αν μια κλάση έχει μια ιδιότητα, η ιδιότητα αυτή κληρονομείται και στις υπο-κλάσεις της αρχικής. Παρόλα αυτά δεν μπορούμε να ξέρουμε σε ποια απ' όλες τις κλάσεις

θα πρέπει να δημιουργηθεί το καινούριο στιγμιότυπο. Έτσι συγκρίνουμε το όνομα της κάθε κλάσης με το όνομα του νέου στιγμιότυπου-τιμής σύμφωνα με τον τρόπο σύγκρισης που χρησιμοποιήσαμε στις κλάσεις. Τέλος παίρνουμε την κλάση με το μεγαλύτερο δείκτη ομοιότητας, για υποψήφια κλάση εισαγωγής του νέου στιγμιότυπου. Αν ο δείκτης αυτός είναι μεγαλύτερος από το άνω όριο που έχει δοθεί από το χρήστη τότε αυτή είναι η κλάση που θα εισαχθεί η νέα τιμή. Αν είναι χαμηλότερος, το στιγμιότυπο δημιουργείται στην κλάση του άμεσου πεδίου τιμών.

Μέχρι αυτό το σημείο έχουμε το στιγμιότυπο-τιμή που θα εισαχθεί, καθώς και την κλάση στην οποία θα την εισάγουμε. Τώρα θα πρέπει να δημιουργηθεί το νέο στιγμιότυπο. Αυτό πετυχαίνεται χρησιμοποιώντας τη λειτουργία ανάθεσης στιγμιότυπου σε κλάση (Αλγόριθμος 10). Η λειτουργία αυτή μας δίνει το νέο στιγμιότυπο που δημιουργήθηκε με τιμές στις ιδιότητες του.

Σαν τελευταίο βήμα, μας μένει η ανάθεση του νέου στιγμιότυπου σαν τιμή στην ιδιότητα επικόλλησης.

6.3 Επίλογος

Στο παρόν κεφάλαιο εξετάσαμε με λεπτομέρεια την υλοποίηση που προτείνουμε για το σενάριο ηλεκτρονικής αγοράς. Αρχικά παρουσιάσαμε κάποια εργαλεία που υλοποιήσαμε μαζί με τον αλγόριθμο, τα οποία έχουν βοηθητικούς σκοπούς (για υπολογισμούς, αποθήκευση δεδομένων κ.λ.π.). Στη συνέχεια περιγράψαμε τα βήματα του αλγορίθμου χωρισμένα σε λειτουργίες-ενότητες. Οι ενότητες αυτές είναι απαραίτητες για καλύτερη περιγραφή και λειτουργία του αλγορίθμου. Στη συνέχεια εξετάζουμε τα αποτελέσματα της υλοποίησης καθώς και διάφορα αποτελέσματα άλλων αλγορίθμων για σύγκριση.

ΚΕΦΑΛΑΙΟ 7

ΑΠΟΤΕΛΕΣΜΑΤΑ – ΠΕΙΡΑΜΑΤΙΚΗ ΑΝΑΛΥΣΗ

7.1 Γενική Περιγραφή

Στη παρούσα εργασία υλοποιήσαμε μια δική μας πρόταση για ταίριασμα και σημασιολογική μετάφραση πληροφοριών από μια οντολογία σε μια άλλη. Ακόμη, επιλέξαμε κάποιους αλγόριθμους από το κεφάλαιο 5, και τους υλοποιήσαμε για πειραματική σύγκριση μεταξύ τους καθώς και με την δική μας υλοποίηση. Η σύγκριση μεταξύ τους θα γίνει με την χρήση των τιμών ακρίβειας-ανάκλησης (precision – recall).

Η ακρίβεια και η ανάκληση είναι δύο τιμές που χρησιμοποιούνται ευρέως για στατιστική κατηγοριοποίηση, και για μέτρηση των δυνατοτήτων των αλγόρίθμων ταιριάσματος οντολογιών. Η ακρίβεια είναι ένα μέσο μέτρησης της ακρίβειας των αποτελεσμάτων, και ορίζεται γενικά ως εξής:

$$\text{Precision} = \frac{\left|\{\text{RelevantDocuments}\} \cap \{\text{Retrieved Documents}\}\right|}{\left|\{\text{Retrieved Documents}\}\right|} \quad (7.1)$$

ή για το ταίριασμα οντολογιών συγκεκριμένα, ο αριθμός των σωστών αποτελεσμάτων που προέκυψαν από τον αλγόριθμο, προς τον αριθμό όλων των αποτελεσμάτων του αλγόριθμου.

Η ανάκληση είναι μια τιμή που δίνει το ποσοστό επιτυχημένων ταιριασμάτων στην παρούσα εκτέλεση. Ορίζεται ως:

$$\text{Recall} = \frac{\left|\{\text{RelevantDocuments}\} \cap \{\text{Retrieved Documents}\}\right|}{\left|\{\text{RelevantDocuments}\}\right|} \quad (7.2)$$

ή ο αριθμός των σωστών αποτελεσμάτων του αλγορίθμου, προς όλα τα σωστά αποτελέσματα, που θα έπρεπε να παρουσιάσει.

Μια άλλη τιμή, για μετρήσεις μεταξύ αλγορίθμων, αποτελεί η μέτρηση-F (F-Measure). Το F-Measure προτάθηκε αρχικά από τον van Rijsbergen (1979), ώστε να υπολογίζει μια τιμή που μετρά την αποτελεσματικότητα του αλγορίθμου. Για τον υπολογισμό αυτής της τιμής, χρησιμοποιείται η τιμή ακρίβειας και ανάκλησης καθώς και μια θετική τιμή β, η οποία αντιστοιχεί στο πόσες φορές σημαντικότερη είναι η τιμή της ανάκλησης, από την τιμή της ακρίβειας. Ο γενικός τύπος είναι:

$$F_{\beta} = \frac{(1+\beta^2)(\text{precision} * \text{recall})}{(\beta^2 * \text{precision} + \text{recall})} \quad (7.3)$$

Στην εργασία μας, θεωρούμε εξίσου σημαντικές και τις δύο τιμές. Έτσι το β γίνεται 1, και ο τύπος που χρησιμοποιούμε είναι:

$$F = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \quad (7.4)$$

Στη τεκμηρίωση αυτή ελέγχουμε την αποτελεσματικότητα των αλγορίθμων υπό το πρίσμα μιας ηλεκτρονικής αγοράς προϊόντων. Για την μέτρηση αυτή, χρησιμοποιούμε τρεις οντολογίες προϊόντων (μια οχημάτων και δύο βιβλίων). Ακόμα, έχουμε δημιουργήσει μια οντολογία προϊόντων η οποία θα αποτελέσει τον κοινό άξονα όλων των ταιριασμάτων (Παράρτημα Β). Θα ελέγξουμε δηλαδή τα αποτελέσματα της διαδικασίας ταιριάσματος, αυτής της οντολογίας με τις άλλες τρεις για κάθε αλγόριθμο.
Η μια (Book.owl) προήλθε από την Amazon [5]. Οι άλλες δύο (book_ontology.owl και vehicle_ontology.owl) είναι κατασκευασμένες από μας. Οι οντολογίες αυτές βρίσκονται στο CD της παρουσίασης.
Οι προαναφερόμενοι αλγόριθμοι μπορεί να παρουσιάσουν διακυμάνσεις στην απόδοσή τους για δύο κυρίως λόγους:

- Όπως έχει αναφερθεί και προηγουμένως, όλοι οι αλγόριθμοι πετυχαίνουν καλύτερα αποτελέσματα σε ορισμένες περιπτώσεις και χειρότερα σε άλλες, αναλόγως με τον τρόπο λειτουργίας τους και τις τεχνικές που χρησιμοποιούν. Συνήθως, σε άλλες μελέτες οι τιμές ακρίβειας, ανάκλησης και F-Measure δίνονται ως μια μέση τιμή των αποτελεσμάτων των αλγορίθμων σε διάφορες περιπτώσεις. Στην περίπτωση μας η παρουσίαση των αποτελεσμάτων γίνεται υπό το πρίσμα της ηλεκτρονικής αγοράς αυστηρά, και έτσι χρησιμοποιούνται μόνο οντολογίες προϊόντων. Χρησιμοποιούμε δηλαδή μια εξειδικευμένη περίπτωση και όχι ένα γενικό σύνολο.
- Ακόμα, οι οντολογίες πηγής αναφέρονται κυρίως σε μια ειδική υποκατηγορία προϊόντων. Έτσι, περιέχουν ένα μικρό αριθμό οντοτήτων, το οποίο συνεπάγεται και μεγάλη απόκλιση τιμών σε ενδεχόμενο λάθος. Για παράδειγμα, θεωρούμε μια οντολογία η οποία περιέχει τρεις κλάσεις που πρέπει να ταιριάξουν. Από τον τύπο της ακρίβειας συμπεραίνουμε πως ένα ενδεχόμενο λανθασμένο ταίριασμα ενός αλγόριθμου ρίχνει την τιμή της κατά 33% περίπου. Αυτό έχει σαν αποτέλεσμα την εμφάνιση γενικά, μικρότερων τιμών σε σχέση με άλλες μελέτες ως προς τους συγκεκριμένους αλγόριθμους.

Το κεφάλαιο αυτό είναι χωρισμένο σε τρεις κατηγορίες: στην παράγραφο 7.2 παρουσιάζουμε λεπτομερέστερα τον τρόπο υλοποίησης των αλγορίθμων, καθώς και τις τιμές που χρησιμοποιήσαμε για την εκτέλεσης τους (βάρη, άνω και κάτω όρια κ.λ.π.). Στην παράγραφο 7.3 παρουσιάζονται τα αποτελέσματα. Πιο συγκεκριμένα, στην 7.3.1 εμφανίζονται τα αποτελέσματα της διαδικασίας ταιριάσματος των οντολογιών. Αξιολογούνται, σχολιάζονται και συγκρίνονται μεταξύ τους. Στην 7.3.2 παρουσιάζονται τα αποτελέσματα της διαδικασίας αντιγραφής των πληροφοριών από την μια οντολογία στην άλλη, έχοντας ως δεδομένα τα αποτελέσματα του ταιριάσματος.

7.2 Συμβάσεις και δεδομένα αλγορίθμων

Στη συνέχεια, παρουσιάζουμε τις διάφορες συμβάσεις που έγιναν για την υλοποίηση του αλγόριθμου AlgS. Ακόμα, εμφανίζονται και τα βάρη που χρησιμοποιήθηκαν για κάθε τεχνική, τα οποία προήλθαν μέσω πειραματικής μελέτης.

Όπως είπαμε και στο κεφάλαιο 6, για τον υπολογισμό των τιμών ομοιοτήτων των κλάσεων χρησιμοποιούνται οι μετρικές Maedche-Staab, Rondriguez-Engenhofer, Wu-Palmer, Lin και Jaro. Σε αυτές τα βάρη που χρησιμοποιούνται για την τελική τιμή θα είναι 0.2, 0.2, 0.4, 0.1 και 0.1 αντίστοιχα. Ακόμα, για τις ιδιότητες χρησιμοποιούμε ακριβώς τα ίδια βάρη για την ομοιότητα ονομάτων. Ακόμα, για το συνδυασμό των τιμών από την ομοιότητα ονόματος, domain και range χρησιμοποιούνται τα βάρη 0.5, 0.22 και 0.28 αντίστοιχα. Να θυμίσουμε εδώ, πως για τον τελικό υπολογισμό χρησιμοποιείται μια ενδιάμεση τιμή, η οποία υπολογίζεται. Αν η τιμή αυτή είναι πάνω από ένα Threshold, τότε υπολογίζεται η συνολική ομοιότητα, αλλιώς θεωρείται ως ομοιότητα η τιμή αυτή. Το Threshold που χρησιμοποιήσαμε στις δοκιμές είναι 0.5185. Τέλος για την διαδικασία του Assign η ιδιότητα προεπιλογής θα είναι η ‘hasDescription’.

Τα όρια πετυχημένων ταιριασμάτων εδώ θα είναι 0.716 και 0.7086 για τις κλάσεις και τις ιδιότητες αντίστοιχα.

7.3 Αποτελέσματα

7.3.1 Διαδικασία Ταιριάσματος Οντολογιών

Σε αυτή την ενότητα παρουσιάζουμε τις επιδόσεις και τα αποτελέσματα των πειραματικών μας δοκιμών. Τα εξετάζουμε για κάθε οντολογία ξεχωριστά, ώστε να μπορέσουμε να αναλυθούν και να σχολιαστούν. Τα σωστά ταιριάσματα με τα οποία συγκρίνουμε τα αποτελέσματα των αλγορίθμων έχουν προκύψει μέσω ελέγχου. Ακόμα να σημειώσουμε πως κάποιοι αλγόριθμοι παρουσιάζουν αποτελέσματα για κλάσεις και

ιδιότητες ενώ κάποιοι άλλοι για κλάσεις μόνο. Έτσι, θα εμφανίσουμε αποτελέσματα για κάθε οντολογία υπό δύο οπτικές γωνίες: για κλάσεις και ιδιότητες στη μια και για κλάσεις μόνο στην άλλη.

Για την οντολογία Book (πηγή Amazon) τα αποτελέσματα (κλάσεις μόνο) είναι τα εξής:

Πίνακας 7.1 – Αποτελέσματα αλγορίθμων για την οντολογία Book Amazon (κλάσεις μόνο).

BOOK AMAZON			
	PRECISION	**RECALL**	**FMEASURE**
AlgS	1	1	1
AnchorPrompt	1	1	1
Cupid	0.5	1	0.66
NOM	1	1	1
CROSI	1	1	1

Για την ίδια οντολογία, τα αποτελέσματα για κλάσεις και ιδιότητες είναι:

Πίνακας 7.2 - Αποτελέσματα αλγορίθμων για την οντολογία Book Amazon (κλάσεις και ιδιότητες).

BOOK AMAZON			
	PRECISION	**RECALL**	**FMEASURE**
AlgS	1	1	1
NOM	0.75	0.75	0.75
CROSI	1	1	1

Η οντολογία Book, έχει πολύ μικρό αριθμό οντοτήτων. Έτσι, τα σωστά ταιριάσματα θα πρέπει να είναι τα: Book-Book, όσον αφορά τις κλάσεις και τα: price-hasPrice, title-hasTitle, hasAuthors-hasAuthor, για τις ιδιότητες.

Στις κλάσεις λοιπόν θα πρέπει να ταιριάξουν οι δύο κλάσεις Book μόνο, ενώ οι άλλες κλάσεις όχι (έχουν μικρή συσχέτιση μεταξύ τους). Παρατηρούμε πως οι κλάσεις Book έχουν ακριβώς ίδια ονοματολογία, άρα και υψηλή λεξικογραφική και σημασιολογική ομοιότητα. Αυτός είναι ο λόγος που σχεδόν όλοι οι αλγόριθμοι παρουσιάζουν τόσο μεγάλες τιμές στον πρώτο πίνακα.

Ο μοναδικός αλγόριθμος που παρουσιάζει χαμηλή απόδοση είναι ο Cupid. Πραγματοποιεί ένα λάθος ταίριασμα, και συγκεκριμένα τις κλάσεις Author-Manufacturer.

Αφού όμως από τα δύο αποτελέσματα του (το σωστό για τις δύο Book κλάσεις, και το λανθασμένο που αναφέρθηκε νωρίτερα) το ένα είναι σωστό, τότε προκύπτει τιμή ακρίβειας πολύ χαμηλή, δηλαδή 0.5. Το λάθος αυτό προκύπτει από την σχεδόν αποκλειστική χρήση της σημασιολογικής ομοιότητας και της απουσίας ισχυρών λεξικογραφικών μεθόδων.

Όσον αφορά τις ιδιότητες, και εδώ τα ταιριάσματα έχουν υψηλή λεξικογραφική και σημασιολογική ομοιότητα και έτσι έχουμε υψηλές τιμές. Στον NOM, έχουμε δύο λάθη: κατά πρώτον ταιριάζει λανθασμένα την ιδιότητα publisher με την hasAuthor. Το λάθος αυτό οφείλεται έμμεσα στην αποκλειστική χρήση της μετρικής του Maedche-Staab για τον υπολογισμό της ομοιότητας ονομάτων. Αυτό έχει σαν συνέπεια την χρήση της τεχνικής με χαμηλό βάρος (να θυμηθούμε πως για την ομοιότητα των ιδιοτήτων, παίζει μεγαλύτερο ρόλο ο έλεγχος ονόματος, domain και range), σε σχέση με τους άλλους δύο ελέγχους, για να αποφευχθούν λανθασμένα αποτελέσματα. Έτσι, αυτές οι δύο ιδιότητες παίρνουν υψηλό ποσοστό ομοιότητας αφού έχουν ακριβώς ίδια domain και range. Το δεύτερο λάθος του, είναι ότι χάνει το σωστό ταίριασμα των ιδιοτήτων price και hasPrice. Το λάθος αυτό συμβαίνει για τον ίδιο λόγο με το προηγούμενο, αλλά έχουμε αντίθετα αποτελέσματα (δεν έχουν υψηλή ομοιότητα στα domain και range τους).

Για την οντολογία Vehicle τα αποτελέσματα (κλάσεις μόνο) θα είναι:

Πίνακας 7.3 – Αποτελέσματα αλγορίθμων για την οντολογία Vehicle (κλάσεις μόνο).

VEHICLE ONTOLOGY			
	PRECISION	**RECALL**	**FMEASURE**
AlgS	1	1	1
AnchorPrompt	1	0.33	0.5
Cupid	1	1	1
NOM	1	0.33	0.5
CROSI	1	1	1

Για την ίδια οντολογία, τα αποτελέσματα για κλάσεις και ιδιότητες είναι:

Πίνακας 7.4 - Αποτελέσματα αλγορίθμων για την οντολογία Vehicle (κλάσεις και ιδιότητες).

VEHICLE ONTOLOGY			
	PRECISION	**RECALL**	**FMEASURE**

AlgS	0.86	1	0.923
NOM	1	0.16	0.28
CROSI	0.66	1	0.8

Τα σωστά ταιριάσματα εδώ, όσον αφορά τις κλάσεις είναι: Vehicle-Vehicle, Car-Vehicle, Motorcycle-Vehicle. Για τις ιδιότητες είναι: Country-hasOriginCountry, Price-hasPrice, Manufacturer-hasManufacturer.

Και εδώ, παρατηρούμε πως ο NOM αλγόριθμος παρουσιάζει χαμηλά ποσοστά απόδοσης. Όπως ειπώθηκε και παραπάνω, χρησιμοποιεί μια μόνο μετρική (Maedche-Staab - λεξικογραφική) για τον υπολογισμό των ομοιοτήτων, με αποτέλεσμα να έχει διάφορα λάθη. Έτσι ταιριάζει τις δύο Vehicle κλάσεις αφού έχουν μεγάλη λεξικογραφική ομοιότητα. Χάνει όμως τις άλλες δύο αφού, παρόλη την μεγάλη σημασιολογική ομοιότητα τους δεν έχουν μεγάλη λεξικογραφική. Έτσι, πετυχαίνει ανάκληση 1 στα 3 ή 0.33. Στις ιδιότητες, αυξάνονται τα λάθος αποτελέσματα, για τον ίδιο λόγο, καθώς και γιατί χρησιμοποιεί τα ήδη λανθασμένα ταιριάσματα των κλάσεων για να υπολογίσει ομοιότητες στις ιδιότητες.

Για τον ίδιο ακριβώς λόγο (αποκλειστική χρήση της λεξικογραφικής τεχνικής Maedche-Staab) έχουμε ίδια αποτελέσματα και στον Anchor-Prompt.

Όσον αφορά τις ιδιότητες, παρατηρούμε πως η υλοποίηση μας παράγει τα καλύτερα αποτελέσματα. Στον CROSI, τα λανθασμένα αποτελέσματα οφείλονται στα βάρη που τίθενται στον υπολογισμό των ιδιοτήτων (έλεγχοι ονόματος, domain και range). Όπως σχολιάσαμε και στην ανάλυση της μεθοδολογίας της δική μας προσέγγισης, θα πρέπει να υπάρχει ένας έλεγχος στον οποίο, αν τα domain δύο ιδιοτήτων παρουσιάζει υψηλή τιμή ομοιότητας, να έχει χαμηλό βάρος η τιμή αυτή, ενώ αντίθετα υψηλό. Δηλαδή, αν τα domain των ιδιοτήτων δεν ταιριάζουν, δεν πρέπει σίγουρα να ταιριάξουν οι κλάσεις. Αν τα domain ταιριάζουν, ο έλεγχος για την ομοιότητα των κλάσεων θα πρέπει να μεταφερθεί κυρίως στα ονόματα και τα range τους. Ο CROSI δεν έχει κανένα τέτοιο έλεγχο και έτσι παρουσιάζει λανθασμένα αποτελέσματα.

Για την οντολογία Book τα αποτελέσματα (κλάσεις μόνο) θα είναι:

Πίνακας 7.5 – Αποτελέσματα αλγορίθμων για την οντολογία Book (κλάσεις μόνο).

BOOK ONTOLOGY			
	PRECISION	**RECALL**	**FMEASURE**
AlgS	1	1	1

AnchorPrompt	1	1	1
Cupid	0.66	1	0.8
NOM	1	1	1
CROSI	1	1	1

Για την ίδια οντολογία, τα αποτελέσματα για κλάσεις και ιδιότητες είναι:

Πίνακας 7.6 - Αποτελέσματα αλγορίθμων για την οντολογία Book (κλάσεις και ιδιότητες).

BOOK ONTOLOGY			
	PRECISION	**RECALL**	**FMEASURE**
AlgS	1	1	1
NOM	0.875	0.66	0.75
CROSI	1	1	1

Τα σωστά αποτελέσματα εδώ θα είναι (κλάσεις): Book-Book Shipment-Shipment. Ιδιότητες: Price-hasPrice, title-hasTitle, hasAuthor-hasAuthor, hasShipment-hasShipment, Cost-hasCost, Region-hasRegion, Discount-hasDiscount.

Και εδώ, έχουμε λάθη παρόμοια με την οντολογία Book της Amazon. Οι λόγοι που ο NOM έχει διαφορετικές τιμές ακρίβειας και ανάκλησης, είναι κατά πρώτον η ύπαρξη περισσότερων σωστών ταιριασμάτων. Έτσι, ένα ενδεχόμενο λάθος δεν έχει τόσο μεγάλη μείωση όσο στην πρώτη οντολογία. Παρόλα αυτά, παράγει και περισσότερα λάθη ο αλγόριθμος για τους λόγους που αναπτύξαμε. Έτσι, συνολικά θα έχει ίδια τιμή F-measure.

Στους παρακάτω πίνακες, εμφανίζονται οι μέσες τιμές των αποτελεσμάτων όλων των αλγορίθμων:

Πίνακας 7.7 – Συνολικά αποτελέσματα (κλάσεις).

	PRECISION	**RECALL**	**FMEASURE**
AlgS	1	1	1
AnchorPrompt	1	0.7766	0.874
Cupid	0.72	1	0.837
NOM	1	0.7766	0.874
CROSI	1	1	1

Πίνακας 7.8 – Συνολικά αποτελέσματα (κλάσεις και ιδιότητες).

	PRECISION	RECALL	FMEASURE
AlgS	0.95	1	0.97
NOM	0.875	0.52	0.65
CROSI	0.886	1	0.94

Συνοψίζοντας λοιπόν, παρατηρούμε πως για το σενάριο ηλεκτρονικής αγοράς, όπου οι οντολογίες κατά βάση δεν έχουν πολύ μεγάλο βάθος, και αφορούν προϊόντα, οι αλγόριθμοι οι οποίοι είναι κατάλληλοι, θα πρέπει να περιέχουν ένα σύνολο μετρικών ώστε η μια να επικαλύπτει τα λάθη της άλλης. Από τους παραπάνω πίνακες φαίνεται πως οι δύο αλγόριθμοι με τις καλύτερες επιδόσεις είναι ο δικός μας καθώς και ο CROSI. Και οι δύο παρουσιάζουν πολύ καλά αποτελέσματα ως προς τις κλάσεις. Αυτό οφείλεται στην χρήση πολλαπλών μετρικών. Ακόμα, η χρήση του μονοπατιού στον υπολογισμό των κλάσεων, που χρησιμοποιήθηκε στη δική μας υλοποίηση, εκτός από σημασιολογική και λεξικογραφική ομοιότητα, χρησιμοποιεί και το περιεχόμενο των κλάσεων (που εξάγεται έμμεσα μέσα από τις σχέσεις των κλάσεων μεταξύ τους). Έτσι, θεωρούμε πως θα μπορούσε να παρουσιάσει παρόμοιες επιδόσεις και σε άλλες περιπτώσεις ταιριάσματος οντολογιών.

Για τις ιδιότητες, η χρήση διαφόρων μετρικών για τις συγκρίσεις ονόματος καθώς και η χρήση σωστών ταιριασμάτων των κλάσεων (που χρησιμοποιούνται για τους ελέγχους domain και range) οδηγούν σε αρκετά ικανοποιητικά αποτελέσματα. Επίσης, η χρήση της ενδιάμεσης τιμής κατά τον συνδυασμό των ελέγχων, απορρίπτει πολλά λανθασμένα αποτελέσματα, αλλά δεν επηρεάζει τον έλεγχο των υπολοίπων.

7.3.2 Διαδικασία Αντιγραφής Τιμών από μια οντολογία πηγής, στην οντολογία προϊόντος.

Εδώ εξετάζουμε την διαδικασία αντιγραφής τιμών (σημασιολογική μετάφραση). Η διαδικασία αυτή, εξαρτάται κυρίως από τα αποτελέσματα της προηγούμενης διαδικασίας, του ταιριάσματος οντολογιών. Χρησιμοποιεί τα αποτελέσματα για να αντιγράψει τιμές/στιγμιότυπα από μια κλάση σε μια άλλη με την οποία ταιριάζει. Έτσι, ως επί το πλείστον, τα αποτελέσματα και το ποσοστό επιτυχίας της διαδικασίας αυτής αποτελεί το ποσοστό επιτυχίας της διαδικασίας ταιριάσματος.

Να σημειωθεί εδώ, πως δεν υπάρχει προς το παρόν αντικειμενικός τρόπος αξιολόγησης αυτής της διαδικασίας. Αυτό γίνεται για τρεις κυρίως λόγους:

- Κατά πρώτον, χρησιμοποιούμε την ιδιότητα προεπιλογής ώστε να είναι σίγουρη η μεταφορά όλων των δεδομένων. Όποια τιμή δεν μπορεί να μπει πουθενά, ή για κάποιο λόγο δεν έγινε σωστά η αντιγραφή στη θέση όπου ταιριάζει, θα γραφτεί στη ιδιότητα προεπιλογής. Έτσι, δεν υπάρχει απώλεια δεδομένων. Η μεταφορά των δεδομένων γίνεται πάντα 100% και έτσι δεν μπορούμε να μετρήσουμε την αποτελεσματικότητα της λειτουργίας ως ποσοστό των πληροφοριών που μεταφέρθηκαν.
- Ένας δεύτερος λόγος, είναι η φύση της διαδικασίας. Δεν παράγει κάποια αριθμητικά αποτελέσματα, τα οποία μπορούν να μετρηθούν αργότερα, αλλά εκτελεί κάποιες συγκεκριμένες λειτουργίες.
- Τέλος, δεν υπάρχει κάποια άλλη παρόμοια προηγούμενη έρευνα πάνω σε αυτή τη λειτουργία, που να υπέπεσε στην αντίληψή μας. Έτσι, δεν υπάρχει κάτι για να γίνει μια σύγκριση με την λειτουργία που αναπτύξαμε.

Ο μόνος τρόπος που απομένει να ελέγξουμε την επιτυχία της διαδικασίας αυτής, ανεξάρτητα από την λειτουργία του ταιριάσματος, είναι η μέτρηση των περιπτώσεων τις οποίες καλύπτει η διαδικασία (με τον όρο περιπτώσεις εννοούμε αν έχουμε αντιγραφή τιμής από τύπο float, σε τύπο object, αν θα πρέπει να μπουν τιμές από δύο διαφορετικές κλάσεις σε μια κ.λ.π.). Στο κεφάλαιο 6, παρουσιάζονται οι περιπτώσεις τις οποίες καλύπτει η διαδικασία αυτή. Γνώμη μας αποτελεί πως οι περιπτώσεις που αντιμετωπίζονται στη λειτουργία που υλοποιήσαμε καλύπτουν ένα αρκετά μεγάλο φάσμα (αν όχι όλο). Στόχος μελλοντικής μας μελέτης αποτελεί ένας τρόπος αντικειμενικής αποτίμησης της αποτελεσματικότητας της λειτουργίας αυτής.

7.4 Επίλογος

Σε αυτό το κεφάλαιο εξετάσαμε τα πειράματα και τα αποτελέσματα της μελέτης που έγινε. Αρχικά, μιλήσαμε για τους τρόπους μέτρησης της αποτελεσματικότητας των αλγόρίθμων (τιμές precision, recall, f-measure). Αναφέραμε πως ακριβώς θα γίνει η σύγκριση των αλγορίθμων που χρησιμοποιήσαμε.

Στη συνέχεια εξετάσαμε με λεπτομέρεια τους τρόπους υλοποίησης των αλγορίθμων που χρησιμοποιήσαμε. Πιο συγκεκριμένα εξετάσαμε τις συμβάσεις που κάναμε (λόγω έλλειψης λεπτομερειών σε άλλες μελέτες επάνω σε αυτούς) καθώς και τα βάρη και όρια που χρησιμοποιήθηκαν. Οι τιμές αυτές, προήλθαν από πειραματική μελέτη διαφόρων τιμών.

Στη συνέχεια παρουσιάσαμε τα αποτελέσματα των αλγόρίθμων, όπου έγιναν φανερά τα προβλήματα των αλγορίθμων:

Ο Anchor-Prompt, στηρίζεται αποκλειστικά σε λεξικογραφική ομοιότητα για τον υπολογισμό των αγκυρών. Έτσι χάνει πολλά ταιριάσματα. Ακόμα, η χρήση αλυσίδας είναι άχρηστη σε μικρές οντολογίες και έτσι ουσιαστικά στηρίζεται σε μια λεξικογραφική μετρική για τον υπολογισμό των αποτελεσμάτων.

Το κυριότερο μειονέκτημα του CROSI αποτελεί η έλλειψη ελέγχου κατά τον συνδυασμό τιμών (ομοιότητα ονόματος, domain και range) στις ιδιότητες.

Ο Cupid χρησιμοποιεί μια μόνο μετρική λεξικογραφικής ομοιότητας η οποία παρουσιάζει και μικρά ποσοστά επιτυχίας. Έτσι έχει δοθεί μεγάλο – σχεδόν αποκλειστικό- βάρος στην σημασιολογική ομοιότητα. Παρόλα αυτά, όπως έχει ειπωθεί και σε προηγούμενο κεφάλαιο, η χρήση μιας μετρικής ελλοχεύει πολλά προβλήματα. Αντίθετα, η χρήση πολλαπλών μετρικών, εμφανίζει ικανοποιητικότερα αποτελέσματα, λόγω της άλληλο-επικάλυψης των λανθασμένων αποτελεσμάτων.

Τέλος, ο NOM χρησιμοποιεί και αυτός αποκλειστικά μια λεξικογραφική μετρική για τον υπολογισμό μιας αρχικής μέτρησης, το οποίο οδηγεί σε πολλά λάθη. Ακόμα, χρησιμοποιεί κάποια χαρακτηριστικά των οντοτήτων τα οποία είναι άχρηστα στο σενάριο που εξετάζουμε σε αυτή την εργασία.

Η μελέτη αυτών των προβλημάτων μας βοήθησε καθοριστικά στην υλοποίηση του δικού μας αλγορίθμου ο οποίος τα αντιμετωπίζει και παράγει ικανοποιητικότερα αποτελέσματα από τους υπόλοιπους, όπως φαίνεται και παραπάνω.

Τέλος, μιλήσαμε για την διαδικασία της σημασιολογικής μετάφρασης πληροφοριών από μια οντολογία στην άλλη. Μιλήσαμε για τα προβλήματα και την αδυναμία εύρεσης μιας αντικειμενικής μεθόδου αξιολόγησης της. Προτείναμε ένα τρόπο αξιολόγησης, τον οποίο σκοπεύουμε να αναπτύξουμε σε μελλοντικές μελέτες, για αντικειμενική μελέτη της επίδοσης της διαδικασίας αντιγραφής.

ΚΕΦΑΛΑΙΟ 8

ΣΥΜΠΕΡΑΣΜΑΤΑ - ΠΡΟΤΑΣΕΙΣ

8.1 Εισαγωγή

Σε αυτό το σημείο θα παρουσιάσουμε κάποιες σκέψεις που είχαμε κατά την διάρκεια της κατασκευής και μελέτης των αποτελεσμάτων του αλγορίθμου μας. Τα συμπεράσματα αυτά αφορούν κυρίως τα χαρακτηριστικά ενός πετυχημένου αλγόριθμου υπό το πρίσμα της ηλεκτρονικής αγοράς. Ακόμα, θα περιγράψουμε προτάσεις για και κατευθύνσεις μελλοντικής έρευνας.

8.2 Τελικά Συμπεράσματα

Όπως φάνηκε και σε προηγούμενη ενότητα, (§ 7.3) ο αλγόριθμος μας παρουσιάζει ικανοποιητικά αποτελέσματα στο σενάριο του ηλεκτρονικού εμπορίου, για τα δεδομένα που χρησιμοποιήσαμε. Παρατηρήθηκαν επίσης ικανοποιητικότερα αποτελέσματα από τους υπολοίπους αλγόριθμους, οι οποίοι περιέχουν διάφορα μειονεκτήματα, όσον αφορά τη λειτουργία ενός shopbot, τα οποία σχολιάστηκαν προηγουμένως.

Μέσα από αυτά τα αποτελέσματα, γίνεται φανερό πως η χρήση πολλαπλών μετρικών ομοιότητας αυξάνει την απόδοση των αλγορίθμων. Όπως έχει ειπωθεί και παραπάνω, καθώς και σε άλλες μελέτες, με πολλαπλές μετρικές επικαλύπτονται τα αρνητικά και οι αδυναμίες της κάθε ατομικής μετρικής και έτσι, αυξάνεται ο βαθμός επιτυχίας των αλγορίθμων. Παρόλα αυτά, θα πρέπει να επιλεχτούν μετρικές από όσο το δυνατόν μεγαλύτερο φάσμα. Θα πρέπει δηλαδή να εξετάζουν την ομοιότητα μελετώντας διαφορετικά χαρακτηριστικά των οντοτήτων. Ακόμα, οι μετρικές αυτές θα πρέπει να περιέχουν έλεγχο σημασιολογικής και λεξικογραφικής ομοιότητας.

Ένα άλλο χαρακτηριστικό που αυξάνει την επίδοση των αλγορίθμων, είναι η χρήση της ενδιάμεσης τιμής κατά τον συνδυασμό διαφόρων ελέγχων πάνω στις ιδιότητες. Να θυμηθούμε πως για τον έλεγχο ιδιοτήτων στη δική μας υλοποίηση, χρησιμοποιήσαμε ελέγχους ονόματος, πεδίου τιμών και ορισμού. Στο συνδυασμό τους, χρησιμοποιήσαμε

μια ενδιάμεση τιμή, η οποία εξαρτιόταν κατά μεγάλο βαθμό από το πεδίο ορισμού. Αν η τιμή αυτή ήταν αρκετά μεγάλη (δηλαδή αν υπήρχε μεγάλη ομοιότητα στο πεδίο ορισμού), ο συνδυασμός γινόταν κανονικά (δίνοντας βάρος στα άλλα χαρακτηριστικά). Αν ήταν μικρή, απορρίπτονταν αυτόματα το ταίριασμα αυτό. Με την ενδιάμεση τιμή, δεν έχουμε τη στατική ανάθεση βαρών, αλλά ο τρόπος υπολογισμού αλλάζει κατά περίπτωση. Έτσι, η μικρή ομοιότητα χαρακτηριστικών (όπως το πεδίο ορισμού) που μπορεί να είναι κοινά σε πολλές οντότητες, συνεπάγεται και μη ομοιότητα των οντοτήτων που τις περιέχουν, αλλά η ομοιότητα τους δεν συνεπάγεται σίγουρα την ομοιότητα των οντοτήτων. Με αυτό τον τρόπο, πετυχαίνουμε περισσότερα σωστά αποτελέσματα χρησιμοποιώντας αυτή τη τεχνική για τον υπολογισμό ομοιότητας τέτοιων χαρακτηριστικών.

Ένα άλλο συμπέρασμα που εξάγεται είναι πως η χρήση μιας ιδιότητας προεπιλογής κατά την δημιουργία της οντολογίας αποθήκευσης πληροφοριών είναι καταλυτική. Η μεταφορά πληροφοριών από την οντολογία πηγής στην οντολογία προϊόντος εξαρτάται από πολλούς παράγοντες. Ο κυριότερος είναι η διαδικασία ταιριάσματος. Αν η διαδικασία αυτή, δεν βρει κανένα ταίριασμα για τα δεδομένα αυτά, δεν θα γίνει η μεταφορά και θα χαθούν. Ακόμα, οι τύποι των πεδίων τιμών επηρεάζουν πολύ τη μεταφορά. Για παράδειγμα, θεωρούμε μια ιδιότητα Α με τύπο πεδίου τιμών A_1 και μια άλλη ιδιότητα Β της άλλη οντολογίας με τύπο πεδίου τιμών B_1. Η πληροφορία που θα μεταφερθεί από την Β στην Α θα πρέπει να ‘μεταφραστεί’ από τον τύπο B_1 στον τύπο A_1. Η μετάφραση αυτή δεν είναι σίγουρη, λόγω κυρίως ασυμβατότητας των τύπων. Για να μην χαθούν αυτά τα δεδομένα θα πρέπει να υπάρχει η ιδιότητα προεπιλογής, ώστε να αποθηκεύονται σε τέτοιες περιπτώσεις.

Άλλα χαρακτηριστικά που θα πρέπει να έχει η οντολογία προϊόντος είναι η χρήση όσο το δυνατόν κοινής ορολογίας και καλή ομαδοποίηση των διαφορετικών κατηγοριών, ώστε να διευκολυνθεί η διαδικασία ταιριάσματος.

Ως τελικό συμπέρασμα, θεωρούμε πως ο έλεγχος διαφορετικών χαρακτηριστικών των οντοτήτων (περιεχόμενο, σημασιολογική και λεξικογραφική ομοιότητα για τις κλάσεις και έλεγχος πεδίου τιμών και ορισμού καθώς και ο σημασιολογικός και λεξικογραφικός έλεγχος) αυξάνει την αποτελεσματικότητα της λειτουργίας των αλγορίθμων. Ακόμα, η σωστή αξιολόγηση των χαρακτηριστικών και της επίδρασης που έχουν στην ομοιότητα των οντοτήτων που ανήκουν οδηγεί σε καλύτερες επιδόσεις.

8.3 Μελλοντικές Κατευθύνσεις

Τα πειράματα που έγιναν στη παρούσα εργασία είναι κυρίως συγκριτικά, χωρίς όμως να μας επιτρέπουν τον υπολογισμό μιας αντικειμενικής τιμής που αντιστοιχεί στην επίδοση των αλγορίθμων. Δηλαδή, δεν μπορούμε να βγάλουμε συμπεράσματα όσον αφορά τις επιδόσεις των αλγορίθμων γενικά, σε μια πληθώρα των περιπτώσεων, αλλά μόνο για το συγκεκριμένο σενάριο, για την βελτίωση της λειτουργίας ενός shopbot. Ο κύριος λόγος είναι η έλλειψη πειραματικών δεδομένων, προσανατολισμένων στο περιβάλλον της ηλεκτρονικής αγοράς. Παρόλα αυτά, στο συγκεκριμένο σενάριο ο αλγόριθμος παρουσίασε ικανοποιητικά αποτελέσματα σε σύγκριση με τους υπολοίπους. Μελλοντικό μας στόχο αποτελεί ο πειραματισμός και σε άλλες οντολογίες, όπως οι UnspscOWL & eclassOWL στα οποία θα δοκιμαστεί ο αλγόριθμος μόλις είναι διαθέσιμες.

Πειραματισμός ακόμα χρειάζεται, και για διαφορετικές οντολογίες για την αποθήκευση των πληροφοριών (PO). Στην παρούσα μελέτη χρησιμοποιήσαμε μια οντολογία που κατασκευάσαμε. Η κατασκευή της PO δεν υπόκεινται σε συμβάσεις και έτσι, μπορεί να δημιουργηθούν με τελείως διαφορετική δομή και οργάνωση από το κάθε κατασκευαστή. Η δομή αυτή δεν θα είναι πάντα η βέλτιστη. Μέσω του πειραματισμού, ελπίζουμε στην εξαγωγή συμπερασμάτων, για τη δημιουργία κάποιων κανόνων και συμβάσεων για τη κατασκευή και χρήση τους, ώστε να βελτιστοποιηθούν τα αποτελέσματα της οντότητας αγορών .

Μια άλλη μελλοντική κατεύθυνση της έρευνας αποτελεί η ανάθεση βαρών. Στην παρούσα εργασία χρησιμοποιήσαμε σταθερές τιμές βαρών οι οποίες προέκυψαν από πειραματική μελέτη, με σκοπό την αύξηση των τιμών ακρίβειας και ανάκλησης. Παρόλα αυτά, οι τιμές οπού σε μια εφαρμογή-ταίριασμα είναι βέλτιστες (παράγουν τα καλύτερα αποτελέσματα) μπορεί σε άλλη εφαρμογή να έχουν αντίθετο αποτέλεσμα. Στην εργασία μας χρησιμοποιήσαμε τιμές οι οποίες παράγουν όχι μεν τα βέλτιστα, αλλά ικανοποιητικά αποτελέσματα σε όλα τα ταιριάσματα. Με την χρήση κάποιων μεθόδων, όπου τα βάρη ανατίθενται δυναμικά αναλόγως με την εφαρμογή, και όχι στατικά όπως εδώ, θα έχουμε και μεγαλύτερη αύξηση της αποτελεσματικότητας των αλγορίθμων.

Ένα άλλο σημείο που θα μας προβληματίσει στο μέλλον είναι η ανάθεση τιμών ομοιότητας για τον έλεγχο των range των ιδιοτήτων. Στην παρούσα εργασία, ο έλεγχος γίνεται ανά περίπτωση. Αν οι δύο ιδιότητες είναι και οι δύο τύπου αντικειμένων ο έλεγχος γίνεται κανονικά. Σε περιπτώσεις όμως που έχουμε, για παράδειγμα μια με τύπο δεδομένων ακέραιο και μια με τύπο δεδομένων συμβολοσειρά, ο έλεγχος είναι

δύσκολος. Δεν μπορούμε να ποσοτικοποιήσουμε την ομοιότητα. Το μόνο που μπορούμε να κάνουμε είναι να αποφασίσουμε αν είναι ίδιοι οι τύποι (τιμή 1) ή διαφορετικοί (τιμή 0). Παρόλα αυτά, θα μπορούσε μια ιδιότητα, με τα ίδια ακριβώς δεδομένα, να οριστεί και με τους δύο τρόπους. Έτσι θα πρέπει να υπάρχει κάποια ιδιότητα. Στην εργασία μας, για να λύσουμε αυτό το πρόβλημα, δώσαμε κάποιες σταθερές τιμές ομοιότητας για κάθε περίπτωση, οι οποίες προέκυψαν μέσα από πειραματισμό. Για το παράδειγμα αυτό δόθηκε η τιμή 0.45. Στο μέλλον, σκοπεύουμε να μελετήσουμε αυτό το φαινόμενο, για βελτιστοποίηση του ελέγχου range.

8.4 Επίλογος

Σε αυτό το τελευταίο κεφάλαιο παρουσιάσαμε κάποια τελικά συμπεράσματα. Είδαμε κάποιες περιπτώσεις ταιριάσματος, και εξηγήσαμε τον τρόπο που αντιμετωπίστηκαν αυτές στην δική μας υλοποίηση. Ακόμη προσφέραμε κάποιες συμβουλές ως προς τη δημιουργία οντολογίας προϊόντος για αποθήκευση πληροφοριών από άλλες. Τέλος, παρουσιάσαμε κάποιες αλλαγές και κατευθύνσεις, που θα βελτιώσουν τη λειτουργία και την επίδοση του αλγορίθμου.

ΠΑΡΑΡΤΗΜΑ Α

Κώδικας Προγράμματος

Ο κώδικας του προγράμματος που υλοποιεί την μεθοδολογία που παρουσιάσαμε στην παρούσα μελέτη περιέχεται στο συνοδευτικό CD-Rom.

ΠΑΡΑΡΤΗΜΑ Β

Οντολογία Προϊόντος (στόχου) που χρησιμοποιήσαμε

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:p2="http://swrl.stanford.edu/ontologies/built-ins/3.4/swrlxml.owl#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:tbox="http://localhost:8080/ontologies/built-ins/3.3/tbox.owl#"
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:temporal="http://localhost:8080/ontologies/built-ins/3.3/temporal.owl#"
    xmlns:p6="http://swrl.stanford.edu/ontologies/built-ins/3.4/swrlm.owl#"
    xmlns:p3="http://swrl.stanford.edu/ontologies/built-ins/3.3/swrlx.owl#"
    xmlns:swrlx="http://localhost:8080/ontologies/built-ins/3.3/swrlx.owl#"
    xmlns:swrlm="http://localhost:8080/ontologies/built-ins/3.4/swrlm.owl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:p7="http://swrl.stanford.edu/ontologies/built-ins/3.3/temporal.owl#"
    xmlns:p5="http://swrl.stanford.edu/ontologies/built-ins/3.3/tbox.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:abox="http://localhost:8080/ontologies/built-ins/3.3/abox.owl#"
    xmlns:swrlxml="http://localhost:8080/ontologies/built-ins/3.4/swrlxml.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:sqwrl="http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl#"
    xmlns:p4="http://swrl.stanford.edu/ontologies/built-ins/3.3/abox.owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns="http://www.owl-ontologies.com/Ontology1237901299.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:swrla="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#"
  xml:base="http://www.owl-ontologies.com/Ontology1237901299.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/built-ins/3.3/tbox.owl"/>
```

```
    <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/built-
ins/3.3/swrlx.owl"/>
    <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/built-
ins/3.3/temporal.owl"/>
    <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/built-
ins/3.4/swrlm.owl"/>
    <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/built-
ins/3.4/swrlxml.owl"/>
    <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/3.3/swrla.owl"/>
    <owl:imports rdf:resource="http://sqwrl.stanford.edu/ontologies/built-
ins/3.4/sqwrl.owl"/>
    <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/built-
ins/3.3/abox.owl"/>
  </owl:Ontology>
  <owl:Class rdf:ID="USA">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Region"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Vehicle">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="ProductCategory"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Matchmaker">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Mediator"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Ticket">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#ProductCategory"/>
    </rdfs:subClassOf>
  </owl:Class>
```

```
<owl:Class rdf:about="#ProductCategory">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:ID="Sport"/>
        <owl:Class rdf:ID="TravelPackage"/>
        <owl:Class rdf:about="#Ticket"/>
        <owl:Class rdf:ID="Service"/>
        <owl:Class rdf:about="#Vehicle"/>
        <owl:Class rdf:ID="Cloth"/>
        <owl:Class rdf:ID="Electronic"/>
        <owl:Class rdf:ID="Book"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Service">
  <rdfs:subClassOf rdf:resource="#ProductCategory"/>
</owl:Class>
<owl:Class rdf:about="#Region">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:ID="Worldwide"/>
        <owl:Class rdf:ID="Asia"/>
        <owl:Class rdf:about="#USA"/>
        <owl:Class rdf:ID="Europe"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Mediator">
```

```
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Market_Entity"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:about="#Sport">
  <rdfs:subClassOf rdf:resource="#ProductCategory"/>
 </owl:Class>
 <owl:Class rdf:ID="Manufacturer"/>
 <owl:Class rdf:about="#TravelPackage">
  <rdfs:subClassOf rdf:resource="#ProductCategory"/>
 </owl:Class>
 <owl:Class rdf:ID="Product"/>
 <owl:Class rdf:about="#Asia">
  <rdfs:subClassOf rdf:resource="#Region"/>
 </owl:Class>
 <owl:Class rdf:ID="Shipment"/>
 <owl:Class rdf:ID="Africa">
  <rdfs:subClassOf rdf:resource="#Region"/>
 </owl:Class>
 <owl:Class rdf:about="#Cloth">
  <rdfs:subClassOf rdf:resource="#ProductCategory"/>
 </owl:Class>
 <owl:Class rdf:ID="Seller">
  <rdfs:subClassOf rdf:resource="#Market_Entity"/>
 </owl:Class>
 <owl:Class rdf:about="#Book">
  <rdfs:subClassOf rdf:resource="#ProductCategory"/>
 </owl:Class>
 <owl:Class rdf:about="#Europe">
  <rdfs:subClassOf rdf:resource="#Region"/>
 </owl:Class>
 <owl:Class rdf:about="#Worldwide">
  <rdfs:subClassOf rdf:resource="#Region"/>
 </owl:Class>
```

```
<owl:Class rdf:about="#Electronic">
  <rdfs:subClassOf rdf:resource="#ProductCategory"/>
</owl:Class>
<owl:Class rdf:ID="Broker">
  <rdfs:subClassOf rdf:resource="#Mediator"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="hasShipment">
  <rdfs:range rdf:resource="#Shipment"/>
  <rdfs:domain rdf:resource="#Product"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasManufacturer">
  <rdfs:range rdf:resource="#Manufacturer"/>
  <rdfs:domain rdf:resource="#Product"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasProduct">
  <rdfs:domain rdf:resource="#Market_Entity"/>
  <rdfs:range rdf:resource="#Product"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasCategory">
  <rdfs:domain rdf:resource="#Product"/>
  <rdfs:range rdf:resource="#ProductCategory"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasProvider">
  <rdfs:range rdf:resource="#Market_Entity"/>
  <rdfs:domain rdf:resource="#Product"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasRegion">
  <rdfs:domain rdf:resource="#Shipment"/>
  <rdfs:range rdf:resource="#Region"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="hasSellerName">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Product"/>
</owl:DatatypeProperty>
```

```
<owl:DatatypeProperty rdf:ID="hasRelevance">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  <rdfs:domain rdf:resource="#Product"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasColor">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Cloth"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasResponseTime">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  <rdfs:domain rdf:resource="#Market_Entity"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasSubCategory">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#ProductCategory"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasPID">
  <rdfs:domain rdf:resource="#Product"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="isManufacturer">
  <rdfs:domain rdf:resource="#Seller"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasAddress">
  <rdfs:domain rdf:resource="#Manufacturer"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasID">
  <rdfs:domain rdf:resource="#Market_Entity"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasPrice">
  <rdfs:domain rdf:resource="#Product"/>
```

```
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:ID="hasModel">
  <rdfs:domain rdf:resource="#Electronic"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:ID="hasNickname">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Manufacturer"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:ID="hasName">
  <rdfs:domain rdf:resource="#Product"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:ID="hasTitle">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Book"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:ID="hasPenalty">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  <rdfs:domain rdf:resource="#Market_Entity"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:ID="hasTimeValidity">
  <rdfs:domain rdf:resource="#Product"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:ID="hasDescription">
  <rdfs:domain rdf:resource="#Product"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:ID="hasCurrency">
  <rdfs:domain rdf:resource="#Product"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </owl:DatatypeProperty>
```

```
<owl:DatatypeProperty rdf:ID="hasReputation">
  <rdfs:domain rdf:resource="#Market_Entity"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasAuthor">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Book"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasOriginCountry">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Product"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasCost">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  <rdfs:domain rdf:resource="#Shipment"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasCC">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  <rdfs:domain rdf:resource="#Vehicle"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="isMotorcycle">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
  <rdfs:domain rdf:resource="#Vehicle"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasDiscount">
  <rdfs:domain rdf:resource="#Product"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasSize">
  <rdfs:domain rdf:resource="#Cloth"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<swrl:Imp rdf:ID="Rule-1">
  <swrl:head>
```

```
      <swrl:AtomList>
       <rdf:first>
        <swrl:BuiltinAtom>
         <swrl:builtin rdf:resource="http://sqwrl.stanford.edu/ontologies/built-
ins/3.4/sqwrl.owl#last"/>
         <swrl:arguments>
          <rdf:List>
           <rdf:first>
            <swrl:Variable rdf:ID="x"/>
           </rdf:first>
           <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
          </rdf:List>
         </swrl:arguments>
        </swrl:BuiltinAtom>
       </rdf:first>
       <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
      </swrl:AtomList>
    </swrl:head>
    <swrla:isRuleEnabled rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</swrla:isRuleEnabled>
    <swrl:body>
      <swrl:AtomList>
       <rdf:first>
        <swrl:ClassAtom>
         <swrl:argument1 rdf:resource="#x"/>
         <swrl:classPredicate rdf:resource="#Product"/>
        </swrl:ClassAtom>
       </rdf:first>
       <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
      </swrl:AtomList>
    </swrl:body>
  </swrl:Imp>
</rdf:RDF>
```

ΑΝΑΦΟΡΕΣ / ΒΙΒΛΙΟΓΡΑΦΙΑ

1. “Intelligent agent”, http://en.wikipedia.org/wiki/Intelligent_agent
2. “Ontology (information science)”, http://en.wikipedia.org/wiki/Ontology_(information_science)
3. “Semantic Web”, http://en.wikipedia.org/wiki/Semantic_Web
4. “Τι είναι και πώς λειτουργεί το ECRM (e-customer relationship management)”, http://www.eeei.gr/interbiz/articles/ecrm.htm
5. Amazon, http://www.amazon.com/
6. G. Antoniou, F. van Harmelen, “A Semantic Web Primer”, MIT Press, Massachusetts, 2004.
7. J. R. Aumann, "game theory", The New Palgrave: A Dictionary of Economics, 2, pp. 460–82
8. D. Aumueller, Do H., S. Massmann, E. Rahm, “Schema and Ontology Matching with COMA++”, SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data (2005), pp. 906-908
9. D. Beneventano, D. Montanari, “Ontological Mappings of Product Catalogues”,. 3rd International Workshop on Ontology Matching
10. W. L. Bethea, C. R. Fink, J. S. Beecher-Deighan, “JHU/APL Onto-Mapology Results for OAEI 2006”, Ontology Matching, volume 225 of CEUR Workshop Proceedings, CEUR-WS.org, *(*2006*)*
11. J. Bock, J. Hettenhausen, “MapPSO Results for OAEI 2008”, 3rd International Workshop on Ontology Matching
12. D. Brickley, R. Guha(eds), “RDF Vocabulary Description Language 1.0: RDF Schema”, W3C Recommendation, 10 February 2004. Latest version available at http://www.w3.org/TR/rdf-schema/
13. A. Budanitsky, G. Hirst, “A survey of approaches to automatic schema matching”, VLDB Journal: Very Large Data Bases, Vol. 10, No. 4. (2001), pp. 334-350.

14. S. Castano, A. Ferrara, G. Messa, “Results of the HMatch Ontology Matchmaker in OAEI 2006”, International Workshop on Ontology Matching co-located with the 5th International Semantic Web Conference ISWC-2006, November 5, Georgia, USA 2006
15. D. Chakrabarty, G. Goel, V. V. Vazirani, L. Wang, C. Yu, “Efficiency, Fairness and Competitiveness in Nash Bargaining Games”, Internet and Network Economics, pg. 498-505, Dec. 2008
16. M. S. Chaves, Strube V. L. de Lima, ”Looking for Similarity among Ontological Structures”, Workshop on Tagging and Shallow Processing of Portuguese, 2003.
17. D. K. W. Chiu, J. K. M. Poon, W. C. Lam, C. Y. Tse, W. H. T. Siu, Poon W.S., “How Ontologies Can Help in an E-marketplace”, European Conference on Information Systems 2005 (ECIS 2005), May 2005
18. W. W. Cohen, Hirsh H., “Joins that generalize: Text classication using whirl”, In Proc. of the Fourth Int. Conf. on Knowledge Discovery and Data Mining (KDD-98), 1998.
19. D. S. Day, M. B. Villain, “Phrase Parsing with Rule Sequence Processors: an Application to the Shared CoNLL Task”, Proc. of CoNLL-2000 and LLL-2000
20. P. Dempster, G. Schafer, “Nonlinear oscillations and boundary-value problems for Hamiltonian systems”. Arch. Rat. Mech. Anal. 78 (1982) 315–333
21. T. Dietterich, “Machine Learning”, Nature Encyclopedia of Cognitive Science, London: Macmillan, 2003
22. Hong-Hai Do, E. Rahm, “COMA - A system for flexible combination of schema matching approaches”, VLDB 2002
23. A. Doan, et al., “Learning to map between ontologies on the semantic web. 11th International World Wide Web Conference. 2002. New York, NY, USA: ACM Press: p. 662-673
24. A. Doan, P. Domingos, A. Levy, “Learning source descriptions for data integration”, Proc. the 3rd Int. Workshop on the Web and Databases (WebDB-2000) , Dallas, May 18--19, 2000, pp.81--86

25. P. Domingos, M. Pazzani, “Beyond independence: Conditions for the optimality of the simple Bayesian classier”, In Proceedings of the Thirteenth International Conference on Machine Learning, pages 105{112, Bari, Italy, 1996. Morgan Kaufmann.
26. M. Ehrig, Y. Sure, “Ontology Mapping - An Integrated Approach”, ESWS pp. 76-91
27. J. Euzenat, P. Valtchev, “Similarity-based ontology alignment in OWL-lite”, In Proc. 15th ECAI, Valencia (ES), 2004, 333-337.
28. D. Fensel, “Ontologies and Electronic Commerce”, IEEE Intelligent Systems, v.16 n.1, p.8-14, January 2001
29. B. Fries, M. Klusch, K. Sycara, “Automated Semantic Web Service Discovery with OWLS-MX”, International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Japan, 2006
30. F. Giunchiglia, P. Shvaiko, M. Yatskevich, “S-Match: An algorithm and an implementation semantic matching”, European Semantic Web Symposium, LNCS 3058, pp. 61-75, (2004)
31. A. Gomez-Perez, M. Fernandez-Lopez, O. Corcho, “Ontological Engineering”, Springer, 2004.
32. O. Gotoh, “An Improved Algorithm for Matching Biological Sequences”, Journal of Molecular Biology, vol. 162, pp 705-708, 1981.
33. P. Hayes(ed.), “RDF Semantics”, W3C Recommendation 10 February 2004. Latest version is available at http://www.w3.org/TR/rdf-mt/
34. A. Heß, “An Iterative Algorithm for Ontology Mapping Capable of Using Training Data”, Proc. 3rd ESWC, LNCS 4011:19-33, 2006
35. G. Hirst, S. St-Onge, “Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms”, In C. Fellbaum, editor, WordNet: An Electronic Lexical Database, chapter 13, pp. 305–332, MIT Press, 1998.
36. W. Hu, N. Jian, Y. Qu, Y. Wang, “GMO: A Graph Matching for Ontologies”, Proceedings of the K-CAP Workshop on Integrating Ontologies, 2005, pp. 41-48
37. Jaccard, "The distribution of the flora of the alpine zone", New Phychologist, vol 11, pp 37-50, 1912

38. M. A. Jaro, “Advances in record linking methodology as applied to the 1985 census of Tampa Florida”, Journal of the American Statistical Society, vol. 64, pp 1183-1210, 1989.

39. Y. R. Jean-Mary, M. R. Kabuka, “ASMOV Results for OAEI 2007”, Proceedings of the 2nd Ontology Matching Workshop; 2007; Busan

40. N. Jian, W. Hu, G. Cheng, Y. Qu, “Falcon-AO: Aligning Ontologies with Falcon”, Proc. of the K-CAP workshop on Integrating Ontologies. (2005) 85–91

41. J. J. Jiang, D. W. Conrath, “Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy“, In Proceedings of International Conference Research on Computational Linguistics (ROCLING X), Taiwan, 1997

42. Y. Kalfoglou, B. Hu, “Crosi mapping system (cms) results of the 2005 ontology alignment contest”, Proceedings of K-Cap'05 Integrating Ontologies workshop, 2005, pp. 77-85.

43. Y. Kalfoglou, M. Schorlemmer, “IF-Map: An Ontology-Mapping Method based on Information-Flow Theory”, Lecture Notes in Computer Science, vol. 2800. Springer-Verlag. pp. 98-127

44. M. Klusch, “Information Agent Technology for the Internet: A Survey”, Data and Knowledge Engineering, Volume 36, Number 3, March 2001.

45. K. Kolomvatsos, S. Hadjiefthymiades, “Implicit Deadline Calculation for Seller Agent Bargaining in Information Marketplaces”, Complex, Intelligent and Software Intensive Systems, 2008. CISIS 2008. International Conference on**,** 4-7 March 2008, p. 184-190

46. M. Lesk, “Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone”, In Proceedings of the 5th annual international conference on Systems documentation, pp 24–26, ACM Press, 1986.

47. V. Levenshtein, “Binary Codes Capable of Correcting Deletions, Insertions, and Reversals”, Soviet Physics Doklady, Volume 10, Issue 8, Pages 707-710

48. Y. Li, Z. A. Bandar, D. McLean, “An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources”, IEEE

Transactions on Knowledge and Data Engineering, vol . 15, No. 4, JULY/AUGUST 2003.

49. D. Lin, “An Information-Theoretic Definition of Similarity”, In Proceedings of the 15th International Conf. on Machine Learning, pp 296–304. Morgan Kaufmann, San Francisco, CA, 1998.

50. Lucene Search Engine, http://lucene.apache.org/java/docs/, 2006

51. J. Madhavan, P. A. Bernstein, E. Rahm, “Generic Schema Matching with Cupid”, The VLDB Journal (2001), pp. 49-58.

52. A. Maedche, S. Staab, “Comparing Ontologies— Similarity Measures and a Comparison Study”, Institute AIFB, University of Karlsruhe, Internal Report, 2001.

53. A. Maedche, S. Staab, ”Measuring Similarity between Ontologies”, Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, Ontologies and the Semantic Web, pp 251-263, 2002.

54. B. Magnini, L. Serafini, M. Speranza, “Semantic Coordination for Document Retrieval”, DBLP:journals/ki/MagniniSS04, 18(4):18-23, 2004

55. S. Melnik, H. Garcia-Molina, E. Rahm, "Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching", Data Engineering, 2002. Proceedings. 18th International Conference on In Data Engineering, 2002. Proceedings. 18th International Conference on (2002), pp. 117-128

56. S. Melnik, H. Garcia-Molina, E. Rahm, "Similarity Flooding: A Versatile Graph Matching Algorithm (Extended Technical Report)", vol. 2003, 2001

57. T. M. Mitchell, “The Discipline of Machine Learning“, Machine Learning Department technical report CMU-ML-06-108, Carnegie Mellon University

58. P. Mitra, M. Kersten, G. Wiederhold, “A Graph Oriented Model for Articulation of Ontology Interdependencies”, In *Advances* in DatabaseTechnology -*EDBT,*2000.

59. P. Mitra, N. F. Noy, A. R. Jaiswal, “Ontology Mapping Discovery with

Uncertainty", Fourth International Semantic Web Conference (ISWC 2004). November, 7th 2005, Galway, Ireland

60. A. E. Monge, C. P. Elkan, "The Field Matching Problem: Algorithms and Applications", In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996.

61. M. Nagy, M. Vargas-Vera, E. Motta, "DSSim-ontology mapping with uncertainty", Proc. 1st Ontology matching workshop, Athens (GA US), pp115-123, 2006

62. S. B. Needleman, C. D. Wunch, "Needleman-Wunch Algorithm for Sequence Similarity Searches", Journal of Mol. Biology, vol. 48, pp 443-453, 1970

63. N. F. Noy, M. A. Musen, "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment", AAAI (2000), pp. 450-455

64. N. Noy, "Ontology Mapping and Alignment", Tutorial, The Third Summer School on Ontological Engineering and the Semantic Web (SSSW'05), 2005

65. N. Noy, "Semantic Interoperability", http://carbon.videolectures.net/2008/active/iswc08_karlsruhe/hendler_ittsw/iswc08_euzenat_noy_si_01.ppt

66. N.F. Noy, , "Anchor-PROMPT: Using Non-Local Context for Semantic Matching ", In IJCA '01: IIn Proceedings of the workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence (4-5 August 2001), pp. 63-70.

67. B. Omelayenko, "Integration of Product Ontologies for B2B Marketplaces: A Preview", ACM SIGecom Exchanges, vol. 2 Issue 1, Dec. 2000, pp. 19-25 (plus citation page).

68. B. Omelayenko, "Ontology Integration Tasks in Business-to-Business E-commerce", Proceedings of the Fourteenth International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems. Springer, Budapest, Hungary (2001): 119-124.

69. P. Pantel, D. Lin, "Discovering word senses from text", Proceedings of 2002 ACMSIGKDD Conference on Knowledge Discovery and

DataMining, 2002, pp. 613–619.

70. T. B. Passin, “Explore’s Guide to the Semantic Web”, Manning Publications, 2004.

71. P. Patel-Schneider, P. Hayes, I. Horrocks(eds.), “OWL Web Ontology Language Semantics and Abstract Syntax”, W3C Recommendation 10 February 2004. Latest version is available at http://www.w3.org/TR/owl-semantics/

72. S. Patwardhan, “Incorporating Dictionary and Corpus Information into a Context Vector Measure of Semantic Relatedness”, Msc Thesis 2003, University of Minnesota.

73. S. Patwardhan, S. Banerjee, T. Pedersen, “Using Measures of Semantic relatedness for Word Sense Disambiguation”, In Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics, pp 241–257, Mexico City, February 2003.

74. T. Pedersen, S. Banerjee, S. Patwardhan, “Maximizing Semantic Relatedness to Perform Word Sense Disambiguation”, 2005.

75. R. Rada, H. Mili, E. Bicknell, M. Blettner, ”Development and Application of a Metric on Semantic Nets”, IEEE Transactions on Systems, Man, and Cybernetics, vol. 19, No 1, pp 17–30, 1989

76. A. Reitano, “Shopping robots and e-commerce”, International Conference on Computer Systems and Technologies - CompSysTech’07, University of Rousse, Bulgária, 2007

77. P. Resnik, “Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language”, Journal of Artificial Intelligence Research vol. 11, pp 95-130, 1999.

78. P. Resnik, “Using Information Content to Evaluate Semantic Similarity in a Taxonomy”, In Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal, 1995.

79. M. A. Rodrıguez, M. J. Egenhofer, “Determining Semantic Similarity among Entity Classes from Different Ontologies”, IEEE Transactions on Knowledge and Data Engineering, vol. 15, No. 2, MARCH/APRIL 2003.

80. F. Sebastiani, “Machine learning in automated text categorization”, ACM Comuting Surveys, 34(1), 2002, 1–47.

81. N. A. L. Seco, “Computational Models of Similarity in Lexical Ontologies”, Msc Thesis, 2005, University College, Dublin.

82. N. Seco, T. Veale, J. Hayes, “An Intrinsic Information Content Metric for Semantic Similarity in WordNet”, In Proceedings of ECAI’2004, the 16th European Conference on Artificial Intelligence, 2004.

83. S. Sosnovsky, “Integration of Overlay UM for Close Domains Based on Domain Ontology Mapping”, http://www.sis.pitt.edu/~paws/assets/ppt/L3S-report20060918.ppt

84. G. Shafer, “Dempster–Shafer theory”, 2002

85. P. Shvaiko, J. Euzenat, ” A Survey of Schema-based Matching Approaches”, Journal on Data Semantics, Vol. 4, pp. 146-171.

86. U. Straccia, R. Troncy, “oMAP: Combining Classifiers for Aligning Automatically OWL Ontologies”, LNCS, vol. 3806. Springer. pp. 133-147

87. J. Tang, J. Li, B. Liang, X. Huang, W. Li, K. Wang, “Using Bayesian decision for ontology mapping”, Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 4, No. 4. (December 2006), pp. 243-262

88. K.M. Ting, I. H. Witten, “Issues in Stacked Generalizations”,

89. A. Tversky, “Features of similarity”, Psychological Review, vol. 4, pp 327–352, 1977.

90. W3C - The World Wide Web Consortium, www.w3.org/

91. J. Wang, Z. Ding, C. Jiang, “GAOM: Genetic Algorithm based Ontology Matching”, Proceedings of IEEE Asia-Pacific Conference on Services Computing, 2006

92. W. E. Winkler, "String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage", Proceedings of the Section on Survey Research Methods, American Statistical Assn., pp 354-359, 1990

93. D.H. Wolpert, “STACKED GENERALIZATION”, Neural Net-works5 (2

)(1992) 241-260

94. WordNet, A lexical database for the English Language, http://wordnet.princeton.edu.

95. Z. Wu, M. Palmer, “Verb Semantics and Lexical Selection”, Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics. Las Cruces, New Mexico.

96. XML 2000 Conference, 3-8 December 2000, Washington DC, http://www.gca.org/attend/2000_conferences/XML_2000/default.htm.

97. M. Yatskevich, F. Giunchiglia, P. Shvaiko, “Semantic Matching: Algorithms and Implementation”, Journal on Data Semantics (2007) 4601:1–38.

98. A. Zavaracky, “Glossary-Based Semantic Similarity in the WordNet Ontology”, Msc thesis, 2003, University College Dublin.

99. Κ. Κολομβάτσος, “Συγκριτική Αξιολόγηση Μεθόδων Λεξικογραφικής και Σημασιολογικής Ομοιότητας”