ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΤΗΝ ΥΠΟΣΤΗΡΙΞΗ ΤΗΣ ΓΕΝΝΗΤΡΙΑΣ ΔΙΕΠΑΦΩΝ ΦΥΣΙΚΗΣ ΓΛΩΣΣΑΣ ONTONL ΚΑΙ ΕΦΑΡΜΟΓΗ ΣΕ ΒΑΣΕΙΣ MPEG-7 ΠΛΗΡΟΦΟΡΙΩΝ.

Ζώτος Αλέξανδρος

Πολυτεχνείο Κρήτης

Τμήμα Ηλεκτρονικών Μηχανικών & Μηχανικών Υπολογιστών

Μία εργασία που παρουσιάστηκε στο Πολυτεχνείο Κρήτης για την εκπλήρωση των απαιτήσεων απόκτησης Διπλώματος στο τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών

Χανιά, 2007

Αφιέρωση

Στους γονείς μου,
Χρήστο και Ελένη, που είναι πάντα δίπλα μου

ΠΕΡΙΛΗΨΗ

Σε αυτή την εργασία παρουσιάζεται η υλοποίηση του **OntoNL Framework**, ενός γενικευμένου αρχιτεκτονικού πλαισίου για τη κατασκευή και χρησιμοποίηση των διεπαφών φυσικής γλώσσας για τις αλληλεπιδράσεις χρηστών με τις βάσεις γνώσεις (knowledge bases) καθώς επίσης και μια διεπαφή αλληλεπίδρασης φυσικής γλώσσας για τις σημασιολογικές βάσεις πολυμέσων που δημιουργήθηκε χρησιμοποιώντας το πλαίσιο του **OntoNL**. Η εφαρμογή του πλαισίου του OntoNL εξετάζει μία σημασιολογική βάση πολυμέσων με το ψηφιακό οπτικοακουστικό περιεχόμενο των γεγονότων ποδοσφαίρου γενικά, που έχουν δειχθεί και αξιολογηθεί στο δεύτερο και τρίτο Annual Review of the DELOS II EU Network of Excellence (IST 507618) (http://www.delos.info/).

Το **OntoNL Framework** υλοποιεί μία πλατφόρμα λογισμικού που αυτοματοποιεί σε ένα μεγάλο βαθμό τη κατασκευή των διεπαφών φυσικής γλώσσας για τις βάσεις γνώσεων. Για να επιτευχθεί η δυνατότητα εφαρμογής και επαναχρησιμοποίησης του πλαισίου OntoNL σε πολλές διαφορετικές εφαρμογές και περιοχές, το λογισμικό χρησιμοποιεί γνώση περιοχής (domain knowledge), αλλά είναι ανεξάρτητο από τις οντολογίες περιοχών (domain ontologies).

Τα τμήματα λογισμικού του **OntoNL Framework** εξετάζουν ομοιόμορφα μια σειρά προβλημάτων στην ανάλυση της πρότασης τα οποία απαιτούν χωριστούς μηχανισμούς. Μια ενιαία αρχιτεκτονική χειρίζεται τη συντακτική και σημασιολογική ανάλυση, αλλά και τις ασάφειες που παρουσιάζονται μέσα σε αυτή. Παράλληλα, το πλαίσιο έχει κατασκευαστεί με τέτοιο τρόπο έτσι ώστε να αποφεύγονται οι εξαρτήσεις με τις βάσεις γνώσης (knowledge bases) με αποτέλεσμα να είναι επαναχρησιμοποιήσιμο σε διαφορετικές εφαρμογές με διαφορετική σημασιολογία περιοχών (domain semantics).

Αυτή η έρευνα που διεξήχθη σε αυτήν τη διατριβή υποστηρίχθηκε από το DELOS II, Network of Excellence on Digital Libraries NoE-G038-507618 / IST-507618

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να εκφράσω τις ιδιαίτερες ευχαριστίες μου στον επιβλέποντα καθηγητή μου κ. Σταύρο Χριστοδουλάκη για την επίβλεψη και καθοδήγησή του στην εκπόνηση αυτής της διπλωματικής εργασίας, καθώς και για τις σημαντικές εμπειρίες που μου προσέφερε κατά την διάρκεια της εργασίας μου στο Εργαστήριο Διανεμημένων Πληροφοριακών Συστημάτων και Εφαρμογών.
Θα ήθελα επίσης να ευχαριστήσω τους καθηγητές κκ. Λαγουδάκη Μιχάλη και Μανιά Κατερίνα για το χρόνο που θα αφιερώσουν στην ανάγνωση του κειμένου και για τις εποικοδομητικές παρατηρήσεις τους.

Επίσης, οφείλω ένα ιδιαίτερο ευχαριστώ στη Νατάσα Καραναστάση, η οποία ήταν υπεύθυνη για το σχεδιασμό της αρχιτεκτονικής του OntoNL, για την συνεργασία και την αρωγή που μου προσέφερε στην υλοποίηση της εργασίας αυτής καθώς και τις συμβουλές της σε τεχνικά θέματα.

Ένα μεγάλο ευχαριστώ σε όλα τα μέλη του εργαστηρίου και ιδιαίτερα στους Γιολδάση Νεκτάριο , Κοτόπουλο Γιάννη και Μυλωνάκη Μανόλη για τη βοήθεια και συμπαράσταση που μου έδειξαν όλο το καιρό που ήμουνα στο εργαστήριο.

Τέλος, (ας μου επιτραπεί το πιο “χαλαρό” ύφος σε αυτή την παράγραφο), ευχαριστώ όλους τους φίλους μου στα Χανιά, τις παλιοσειρές, κυρίως για τη κατανόηση τους όλα αυτά τα χρόνια που δεν ήταν πάντα εύκολα αλλά μαζί τους έγιναν παραπάνω από ευχάριστα. Ευχαριστώ όλα τα γήπεδα 5x5 και όλα τα σουβλατζίδικα της γειτονιάς γιατί χωρίς αυτά η επιβίωση στα Χανιά θα ήταν πολύ πιο δύσκολη.

ΔΗΜΟΣΙΕΥΣΕΙΣ

Μέρος της δουλειάς που περιγράφεται σε αυτή τη διπλωματική εργασία έχει δημοσιευτεί στα ακόλουθα Conference Proceedings:

- A. Karanastasi, A. Zotos, S. Christodoulakis: "The OntoNL Framework for Natural Language Interface Generation and a Domain-specific Application", in Proceedings of the DELOS Conference on Digital Libraries, 13-14 February, Tirrenia, Pisa, Italy 2007
- A. Karanastasi, A. Zotos, S. Christodoulakis: "User Interactions with Multimedia Repositories using Natural Language Interfaces: an Architectural Framework and its Implementation", in the Journal of Digital Information Management (JDIM), Volume 4 Issue 4, December 2006
- A. Karanastasi, A. Zotos, S. Christodoulakis: "User Interactions with Multimedia Repositories using Natural Language Interfaces: an Architectural Framework and its Implementation", in Proceedings of the Fourth Special Workshop on Multimedia Semantics (WMS06), June 19-21, 2006

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΕΦΑΛΑΙΟ 1
ΕΙΣΑΓΩΓΗ

1.1 Εισαγωγή - Αναγκαιότητα

Ο στόχος της επεξεργασίας της φυσικής γλώσσας (Natural Language Processing ή αλλιώς NLP) είναι να σχεδιάσει και να κατασκευάσει λογισμικό που θα αναλύει και θα καταλαβαίνει γλώσσες που οι άνθρωποι χρησιμοποιούν φυσικά, έτσι ώστε τελικά ο χρήστης να μπορεί να απευθυνθεί στον υπολογιστή του με τον ίδιο τρόπο που θα απευθυνόταν σε ένα άλλο άτομο.

Αυτός ο στόχος βέβαια δεν είναι εύκολο να επιτευχθεί. Το να καταλαβαίνει κάποιος τη γλώσσα σημαίνει μεταξύ άλλων, να γνωρίζει ποιες έννοιες μία λέξη ή φράση αντιπροσωπεύουν και να ξέρει πως να συνδέσει αυτές τις έννοιες μαζί με τρόπο τέτοιο ώστε το τελικό αποτέλεσμα να έχει νόημα. Αποτελεί σχήμα οξύμωρο το γεγονός ότι η φυσική γλώσσα, το σύστημα συμβόλων που είναι το ευκολότερο για τους ανθρώπους να καταλάβουν, είναι το πιο δύσκολο αντίστοιχα για τους υπολογιστές.

Το **OntoNL Framework** [1] για τα συστήματα επεξεργασίας φυσικής γλώσσας έχει ως κίνητρο τις εξής παρατηρήσεις:

1. Υπάρχει η ανάγκη να οργανωθούν οι διάφορες μέθοδοι που χρησιμοποιούνται για τη συντακτική και σημασιολογική ανάλυση σε ένα γενικευμένο πλαίσιο παραγωγής διεπαφών φυσικής γλώσσας με καλά αποτελέσματα χωρίς την εξάρτηση από συγκεκριμένους γραμματικούς κανόνες και εξαρτώμενα λεξικά από περιοχές γνώσης Το πλαίσιο περιγράφει μια μεθοδολογία για ένα σύστημα πληροφοριών από την άποψη ενός συνόλου δομικών μονάδων και πως αυτές οι δομικές μονάδες εγκαθίστανται από κοινού.
2. Η γνώση του πλαισίου στο οποίο μια ασάφεια εμφανίζεται είναι κρίσιμη για την επίλυση του. Η ανάλυση της πρότασης γενικά και η κατανόηση των εννοιών ειδικότερα, απαιτούν μια σειρά από στάδια λεξικολογικής και εννοιολογικής αποσαφήνισης. Το σύστημα αντιστοιχίζει τις λέξεις μιας πρότασης στα μέρη του λόγου, τις ακολουθίες από μέρη του λόγου σε

χαμηλού επιπέδου συστατικά και τα χαμηλού επιπέδου συστατικά σε γλωσσικά μοντέλα.

3. Με την ανάπτυξη μιας γεννήτριας διεπαφών φυσικής γλώσσας παρέχουμε στους υπεύθυνους για την ανάπτυξη συστημάτων τον πιο φυσικό τρόπο για να επικοινωνήσουν με τους χρήστες τους. Έτσι λοιπόν δημιουργήθηκε η ανάγκη τεκμηρίωσης της λειτουργικότητας και χρήσης του **OntoNL.** Αποτέλεσμα αυτής της ανάγκης ήταν η υλοποίηση και χρήση του OntoNL σε συγκεκριμένο περιβάλλον που θα αναλυθεί στα πλαίσια της παρούσας διπλωματικής εργασίας

1.2 Στόχοι διπλωματικής

Στη παρούσα διπλωματική εργασία θα παρουσιαστεί η υλοποίηση του πλαισίου OntoNL που αποτελεί γεννήτρια διεπαφών φυσικής γλώσσας για αλληλεπιδράσεις με βάσεις γνώσεων και η οποία θέτει τους εξής στόχους:

1. Το **OntoNL Framework** είναι σε θέση να εξετάσει ομοιόμορφα μια σειρά προβλημάτων στην ανάλυση της πρότασης, κάθε ένα από τα οποία θα απαιτούσε διαφορετικό υπολογιστικό μηχανισμό. Ειδικότερα μια ενιαία αρχιτεκτονική που :

   α) χειρίζεται τις συντακτικές και σημασιολογικές ασάφειες

   β) χρησιμοποιεί τα σημασιολογικά μέτρα συσχέτισης για τις έννοιες της οντολογίας έτσι ώστε να παρέχει τα καλύτερα ταξινομημένα αποτελέσματα

2. Το **OntoNL Framework** κάνει χρήση της γραμματικής και των κανόνων της OWL[18] χρησιμοποιώντας τις ανώτερες και περιοχής οντολογίες (upper και domain). Η σημασιολογική διαδικασία αναζήτησης που υλοποιείται εδώ έχει ως σκοπό να ικανοποιήσει τα διαφορετικά είδη και τα επίπεδα ασάφειας. Δοθέντος μιας OWL οντολογίας , τα βάρη ορίζονται βασισμένα σε ορισμένες ιδιότητες της οντολογίας, έτσι ώστε να μετρούν το επίπεδο συγγένειας μεταξύ των εννοιών. Κατ' αυτό τον τρόπο [1] προσδιορίζονται οι σχετικές - συσχετιζόμενες έννοιες στην οντολογία που καθοδηγούν τη σημασιολογική διαδικασία αναζήτησης.
3. Το **OntoNL Framework** είναι επαναχρησιμοποιήσιμο, ανεξάρτητο της περιοχής γνώσης, και δουλεύει με είσοδο μόνο μια OWL οντολογία που μπορεί να χρησιμοποιηθεί ως σχήμα αναφοράς (reference schema) για τη δημιουργία μιας βάσης γνώσης στην οποία θα απευθυνθεί η διεπαφή.

Για να αποδείξουμε αυτές τις αξιώσεις , χρησιμοποιούμε το OntoNL Framework για να δημιουργήσουμε μια διεπαφή φυσικής γλώσσας για ένα MPEG-7 Semantic Repository με πληροφορία που αφορά το τομέα του ποδοσφαίρου. Η διεπαφή φυσικής γλώσσας (Natural Language Interface) χρησιμοποιείται σε ένα σύστημα ερωταπαντήσεων (question – answering) το οποίο έχει ως είσοδο το αίτημα του χρήστη σε φυσικό λόγο και μετά από διάφορα στάδια επεξεργασίας επιστρέφει αποτελέσματα σε αυτόν.

1.3 Το πρόγραμμα DELOS II

Η εργασία αυτή πραγματοποιήθηκε μέσα στα πλαίσια του ερευνητικού προγράμματος **DELOS II**, στο οποίο συμμετέχει το Εργαστήριο Διανεμημένων Πληροφοριακών Συστημάτων και Εφαρμογών.

Το **DELOS II** είναι ένα ευρωπαϊκό πρόγραμμα (Network of Excellence) στις ψηφιακές βιβλιοθήκες που χρηματοδοτείται μερικώς από την Ευρωπαϊκή Επιτροπή στο πλαίσιο του προγράμματος τεχνολογιών κοινωνίας των πληροφοριών (IST). Οι κύριοι στόχοι **DELOS II** είναι η έρευνα, τα αποτελέσματα της οποίας αφορούν στο δημόσιο τομέα, και τη μεταφορά τεχνολογίας, μέσω των συμφωνιών συνεργασίας με τα ενδιαφερόμενα συμβαλλόμενα μέρη.

Το ότι όλοι οι πολίτες, οπουδήποτε, οποτεδήποτε, πρέπει να έχουν πρόσβαση στις συνδεμένες με το διαδίκτυο ψηφιακές συσκευές για να ψάξουν την όλη ανθρώπινη γνώση, ανεξάρτητα από τα εμπόδια του χρόνου, της θέση, του πολιτισμού ή της γλώσσας είναι ένα όραμα του **DELOS II** από την έναρξή του. Με αυτό το στόχο, το **DELOS II** διευθύνει ένα κοινό πρόγραμμα των δραστηριοτήτων που στοχεύουν στην ενσωμάτωση και το συντονισμό των τρεχουσών ερευνητικών προσπαθειών των σημαντικότερων ευρωπαϊκών ομάδων που εργάζονται στις ψηφιακές βιβλιοθήκη-σχετικές με τον περιοχές. Ο κύριοι στόχος και ο σκοπός του είναι να αναπτύξει την επόμενη γενεά των ψηφιακών τεχνολογιών βιβλιοθήκης, βασισμένη στις υγιή περιεκτικά θεωρίες και τα πλαίσια για τον κύκλο της ζωής των ψηφιακών πληροφοριών βιβλιοθήκης.

Το **DELOS II** λειτουργεί αυτήν την περίοδο στην ανάπτυξη ενός ψηφιακού προτύπου αναφοράς βιβλιοθήκης που σχεδιάζεται για να ικανοποιήσει τις ανάγκες των συστημάτων επόμενης γενιάς, και σε μια συνολικά ενσωματωμένη εφαρμογή πρωτοτύπων ενός ψηφιακού συστήματος διαχείρισης βιβλιοθήκης, αποκαλούμενη

DelosDLMS, το οποίο θα χρησιμεύσει ως μια συγκεκριμένη μερική εφαρμογή του προτύπου αναφοράς και θα καλύψει πολλά τμήματα λογισμικού που αναπτύσσονται από τους συνεργάτες του **DELOS II**. Είναι δύο σημαντικά βήματα στην κατεύθυνση του οράματος **DELOS II**.

Για να επιτευχθούν οι στόχοι του **DELOS II** ένας αριθμός από εργασίες ορίστηκαν και εκτελέστηκαν. Ανάμεσα σε αυτές, υπήρχε και η πρόταση που αφορά το **OntoNL[1]** και προέβλεπε μια γενική ψηφιακή γνώση διαχείρισης αρχιτεκτονικής βιβλιοθηκών. Η βασική υπόθεση που έγινε είναι ότι υπάρχουν οντολογίες εννοιών συγκεκριμένες σε περιοχές γνώσεις, οι οποίες χρησιμοποιούνται και για τη δημιουργία του περιεχομένου μεταδεδομένων της ψηφιακής αποθήκης, καθώς επίσης και για την καθοδήγηση της διατύπωσης, το συλλογισμό και την απάντηση των αιτημάτων των χρηστών. Μια γενική γλώσσα οντολογίας (όπως η OWL) θα χρησιμοποιηθεί για να περιγράψει οποιεσδήποτε οντολογίες περιοχών που χρησιμοποιούνται στις εφαρμογές και μια γλώσσα χειρισμού γνώσης (όπως OWL-QL) θα είναι ο επίσημος τρόπος επικοινωνίας με την βάση γνώσης. Το σύστημα διεπαφών φυσικής γλώσσας βοηθά στην εισαγωγή αιτημάτων από τους χρήστες στην επίσημη γλώσσα χειρισμού γνώσης, λαμβάνοντας υπόψη τις συγκεκριμένες οντολογίες περιοχών. Αυτό το βήμα περιλαμβάνει την εξαγωγή των βασικών εννοιών, των σχέσεων και των ιδιοτήτων από το αίτημα του χρήστη, που δημιουργεί μια γραφική παράσταση με τη βοήθεια της γνώσης στις οντολογίες και την αποθήκη.

Ο γενικός στόχος είναι να αυτοματοποιηθεί όσο το δυνατόν περισσότερο η κατασκευή των διεπαφών φυσικής γλώσσας στις βάσεις γνώσης. Σε αυτό το στόχο δεν διευκρινίζεται ποια θα πρέπει να είναι η δομή αποθήκευσης για τα μεταδεδομένα. Τα μεταδεδομένα θα μπορούσαν να αποθηκευτούν σε μια αποθήκη γνώσης (όπως μια αποθήκη RDF) ή θα μπορούσαν να αποθηκευτούν σε συγγενικά συστήματα υπό τον όρο ότι οι μηχανισμοί συμπεράσματος που υποστηρίζουν τη γλώσσα χειρισμού γνώσης θα έχουν χτιστεί πάνω σε αυτά. Το σύστημα φυσικής γλώσσας θα πρέπει επίσης να λάβει υπόψη εκτός από τις οντολογίες περιοχής (domain ontologies), τις οντολογίες λέξεων (όπως WordNet) και τη διεπαφή μεταξύ αυτών των δύο [11].

1.4 Δομή εργασίας

Μετά το τρέχον –πρώτο κεφάλαιο-, το οποίο αποτελεί και την εισαγωγή της διπλωματικής διατριβής, ακολουθούν εφτά ακόμα κεφάλαια.

Στο δεύτερο κεφάλαιο παρουσιάζεται η σχετική με τη διπλωματική διατριβή έρευνα, δηλαδή όλες οι γνώσεις και τεχνολογίες οι οποίες σχετίζονται με το **OntoNL** γενικότερα. Στο ίδιο κεφάλαιο παρουσιάζονται οι πιο δημοφιλείς εργασίες οι οποίες έχουν παρόμοιο αντικείμενο μελέτης με τη παρούσα εργασία.

Στο τρίτο κεφάλαιο περιγράφονται οι τεχνολογίες υλοποίησης που χρησιμοποιήθηκαν για τη δημιουργία της εφαρμογής **OntoNL.**

Στο τέταρτο κεφάλαιο παρουσιάζεται η αρχιτεκτονική του συστήματος. Περιγράφεται ο τρόπος με τον οποίο σχεδιάστηκε και γιατί, καθώς και οι μονάδες που το αποτελούν, οι οποίες αναλύονται με τέτοιο τρόπο έτσι ώστε να γίνει κατανοητή η χρησιμότητα της καθεμιάς και ο τρόπος με τον οποίο αλληλεπιδρούν μεταξύ τους.

Στο πέμπτο κεφάλαιο αναλύεται λεπτομερώς η υλοποίηση της συγκεκριμένης αρχιτεκτονικής και επίσης παραθέτονται παραδείγματα που κάνουν πιο κατανοητή τη συγκεκριμένη υλοποίηση.

Στο έκτο κεφάλαιο παρουσιάζεται ένα ολοκληρωμένο σενάριο εφαρμογής του **OntoNL** Framework για την ικανοποίηση ανάκτησης οπτικοακουστικού υλικού που αφορά το ποδόσφαιρο, η αρχιτεκτονική του, πως αυτό υλοποιείται καθώς και διάφορες ιδιαιτερότητες που αντιμετωπίσαμε

Στο έβδομο κεφάλαιο παρουσιάζονται τα αποτελέσματα δοκιμών χρήσης της εφαρμογής του **OntoNL** πλαισίου στο τομέα του ποδοσφαίρου που μας βοηθάνε να συμπεράνουμε τη σταθερότητα και την ευχρηστία του συστήματος.

Στο όγδοο και τελευταίο κεφάλαιο γίνεται ανακεφαλαίωση της διατριβής, αναφέρονται τα συμπεράσματα της εργασίας και επισημαίνονται μελλοντικές επεκτάσεις.

ΚΕΦΑΛΑΙΟ 2
ΕΠΙΣΚΟΠΗΣΗ ΣΧΕΤΙΚΗΣ ΕΡΕΥΝΑΣ

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναφερθούν γνώσεις σχετικές με το αντικείμενο μελέτης της παρούσας εργασίας, για τις οποίες θα γίνει μία σύντομη περιγραφή των βασικών χαρακτηριστικών τους. Επίσης θα περιγραφούν και υπάρχουσες εργασίες με σχετιζόμενο αντικείμενο μελέτης, τις οποίες ξεχωρίσαμε και θα αναφερθούν τα πλεονεκτήματα και τα μειονεκτήματα τους.

2.2 Φυσική γλώσσα

Στις παρακάτω παραγράφους θα δώσουμε το θεωρητικό υπόβαθρο για την ομαλή εισαγωγή μας σε έννοιες και λειτουργίες που αφορούν τη φυσική γλώσσα.

2.2.1 Προκαταρκτικά

Προκειμένου να καταστεί το υπόλοιπο αυτής της διατριβής συνεπές και κατανοητό, θα πρέπει να διευκρινιστούν μερικές βασικές έννοιες. Αυτή η εργασία εξετάζει τις πτυχές της γλωσσικής τεχνολογίας, και όχι τη λεκτική τεχνολογία [17]. Η λεκτική τεχνολογία αναφέρεται στην επεξεργασία γλωσσών στο επίπεδο της φωνολογίας και της φωνητικής και θεμελιώνεται στην Ακουστική, την Ηλεκτρομηχανική και την Πληροφορική.

Η γλωσσική τεχνολογία είναι η διαδικασία της ανάλυσης της εισαγωγής κειμένου από την άποψη της σύνταξης, της μορφολογίας, και της σημασιολογίας. Τα θεμέλια για αυτήν βρίσκονται στην υπολογιστική γλωσσολογία, την ψυχολογία και την πληροφορική. Τα συστήματα που περιγράφονται εδώ είναι συστήματα διαλόγου που περιορίζονται στην επεξεργασία της γραπτής - ή δακτυλογραφημένης - αλληλεπίδρασης .

Η έννοια της φυσικής γλώσσας (natural language) αφορά στις ανθρώπινες γλώσσες που μπορούν να γραφτούν ή/ και να μιληθούν (π.χ. Αγγλικά, Ελληνικά).

2.2.2 Συντακτική Ασάφεια

Υπάρχουν πολλές περιπτώσεις φυσικού λόγου, στις οποίες παρουσιάζονται ασάφειες στα μέρη του λόγου που αποτελούν τη πρόταση, δηλαδή περιπτώσεις που ίδιες λέξεις αποτελούν διαφορετικό μέρους του λόγου ανάλογα με τη πρόταση στην οποία βρίσκονται. Ας θεωρήσουμε τη λέξη "**chairs**" στις δύο παρακάτω περιπτώσεις.

1. Professor James Cameron **chairs** the workshop held by the Technical University of London.
2. IKEA **chairs** were used for the workshop held by the Technical University of London.

Παρατηρούμε λοιπόν ότι η λέξη **chairs** είναι ρήμα στο πρώτο παράδειγμα ενώ ουσιαστικό στο δεύτερο. Αντιλαμβανόμαστε λοιπόν πόσο σημαντικό είναι να διακρίνουμε σωστά τα μέρη του λόγου, έτσι ώστε να αποσαφηνίσουμε αποδοτικά το νόημα της κάθε πρότασης. Η διάκριση αυτή είναι απαραίτητη εάν επιθυμούμε να αναλύσουμε τη πρόταση ,αφού ίδιες λέξεις αν ερμηνευτούν με διαφορετικό τρόπο, δίνουν τελείως διαφορετικό νόημα στη πρόταση.

2.2.3 Σημασιολογική Ασάφεια

Ακόμα και αν το μέρος του λόγου μιας λέξης είναι γνωστό, η έννοια της λέξης μέσα σε μία πρόταση επίσης απαιτεί συχνά αποσαφήνιση. Η λέξη "**player**" για παράδειγμα, είναι ουσιαστικό και στις δύο προτάσεις που ακολουθούν.

1. He was the best **player** of Barcelona ever.
2. He was a major **player** in setting up the corporation.

Παρόλα αυτά, αν και η λέξη **player** είναι ουσιαστικό και στις δύο παραπάνω προτάσεις, στη πρώτη πρόταση αναφέρεται σε ένα πρόσωπο που συμμετέχει ή είναι ειδικευμένο σε κάποιο παιχνίδι, ενώ στη δεύτερη πρόταση χρησιμοποιείται μεταφορικά για να επισημάνει έναν σημαντικό συμμετέχοντα .

Ακόμα και όταν φαίνεται ότι μια λέξη είναι απόλυτα σαφής, μπορεί να χρησιμοποιηθεί σε πλαίσια που παρέχουν μια νέα έννοια. Γενικά, κάποιος ίσως να έλεγε ότι η λέξη **Milan** είναι σαφής, παραδείγματος χάριν. Αναφέρεται σε μια πόλη στη βόρεια Ιταλία. Ας εξετάσουμε όμως τη παρακάτω περίπτωση.

- **Milan** will be looking to continue their positive run of results at Chievo when they take the pitch at the Bentegodi stadium on Saturday.

Σε αυτή τη περίπτωση λοιπόν αντιλαμβανόμαστε ότι η λέξη αναφέρεται στη ποδοσφαιρική ομάδα **Milan** η οποία συμμετέχει σε κάποιο ποδοσφαιρικό πρωτάθλημα.

Επομένως συμπεραίνουμε πως ακόμη και αν γνωρίζουμε το μέρος του λόγου μιας λέξης δε μπορούμε να είμαστε σίγουροι για την έννοια της, γεγονός που απαιτεί επιπλέον αποσαφήνιση.

2.2.4 Ενώσεις Ουσιαστικού (Noun Compound Analysis)

Οι ενώσεις ουσιαστικού (noun compounds) εμφανίζονται συχνά στην επεξεργασία φυσικής γλώσσας (Natural Language Processing, NLP). Σύμφωνα με αυτές δύο ή περισσότερα ουσιαστικά σε ακολουθία λειτουργούν μαζί εννοιολογικά ως ουσιαστικό. Παρόλα αυτά η επεξεργασία τους δεν είναι και τόσο απλή καθώς παρουσιάζονται περιπτώσεις στις οποίες απαιτείται διαφορετική αντιμετώπιση ανάλογα με τη σειρά που ενδεχομένως να εμφανίζονται τα ουσιαστικά μέσα στη πρόταση.

Ας εξετάσουμε λοιπόν τις παρακάτω δύο περιπτώσεις που εμφανίζονται ζευγάρια που αποτελούν noun compounds:

1. [player [shirt number]]
2. [soccer team] shirt]

Αν και στις δύο περιπτώσεις εμφανίζονται τα ίδια μέρη λόγου παρατηρούμε ότι στη πρώτη περίπτωση έχουμε right – branching αφού το number αναφέρεται στο shirt και το shirt number στο player αντίστοιχα ενώ στη δεύτερη left – branching (το soccer αναφέρεται στο team και το soccer team στο shirt). Συμπεραίνουμε λοιπόν ότι αν και συχνό φαινόμενο, τα noun compounds απαιτούν ιδιαίτερη προσοχή ώστε να ερμηνευτούν σωστά.

2.2.5 Οι διεπαφές φυσικής γλώσσας σε βάσεις δεδομένων (Ιστορική Αναδρομή)

Το πρωτότυπο NLIDBs (Natural Languages Interfaces to Databases) είχε εμφανιστεί ήδη προς το τέλος της δεκαετίας του '60 και στις αρχές της δεκαετίας του

'70. Το ποιο γνωστό NLIDB εκείνης της περιόδου είναι το Lunar [29] , μια διεπαφή φυσικής γλώσσας σε μια βάση δεδομένων που περιέχει τις χημικές αναλύσεις για τους βράχους του φεγγαριού (moon rocks). Το Lunar και άλλες πρόσφατες διεπαφές φυσικής γλώσσας χτίστηκαν έχοντας μια ιδιαίτερη βάση δεδομένων στο μυαλό, και δεν θα μπορούσαν έτσι να τροποποιηθούν εύκολα για να χρησιμοποιηθούν σε διαφορετικές βάσεις δεδομένων (αν και οι εσωτερικές μέθοδοι αντιπροσώπευσης που χρησιμοποιήθηκαν στο Lunar υποστηρίχτηκαν για να διευκολύνουν την ανεξαρτησία μεταξύ της βάσης δεδομένων και άλλων ενοτήτων [30], ο τρόπος που αυτές χρησιμοποιήθηκαν ήταν κάπως συγκεκριμένος για τις ανάγκες εκείνου του προγράμματος).

Μέχρι τα τέλη της δεκαετίας '70 αρκετά ακόμα NLIDBs είχαν εμφανιστεί. Το Rendezvous [31] παρείχε στο χρήστη διάλογους (dialogues) για να τον βοηθήσουν να διατυπώσει τις ερωτήσεις του. Το Ladder [32] μπορούσε να χρησιμοποιηθεί από τις μεγάλες βάσεις δεδομένων, και μπορούσε να διαμορφωθεί ώστε να διασυνδέεται σε διαφορετικά συστήματα διαχείρισης βάσεων δεδομένων (DBMSS). Το Ladder χρησιμοποίησε τις σημασιολογικές γραμματικές, μια τεχνική που κάνει διαφύλλωση (interleaving) στη συντακτική και σημασιολογική επεξεργασία. Αν και οι σημασιολογικές γραμματικές βοήθησαν να εφαρμοστούν συστήματα με εντυπωσιακά χαρακτηριστικά, τα προκύπτοντα συστήματα αποδείχθηκαν δύσκολα ώστε να εφαρμοστούν στις διαφορετικές περιοχές εφαρμογής. Πράγματι, μια διαφορετική γραμματική έπρεπε να αναπτυχθεί όποτε ο Ladder διαμορφωνόταν για μια νέα εφαρμογή. Δεδομένου ότι οι ερευνητές άρχισαν να εστιάζουν σε φορητά (portable) NLIDBs, οι σημασιολογικές γραμματικές εγκαταλείφθηκαν βαθμιαία. Το Planes [33] και το Philiqa1 [34] ήταν μερικά από τα άλλα NLIDBs που εμφανίστηκαν προς το τέλος της δεκαετίας του '70.

Το Chat-80 [35] είναι ένα από τα πιο γνωστά NLIDBs στις πρώτες δεκαετίας του '80. Το Chat-80 υλοποιήθηκε εξ ολοκλήρου σε Prolog. Μετασχημάτιζε τις αγγλικές ερωτήσεις σε εκφράσεις Prolog (Prolog expressions), οι οποίες αξιολογούνταν στη μια βάση δεδομένων με πληροφορία σε Prolog. Ο κώδικας του Chat-80 κυκλοφόρησε ευρέως, και αποτέλεσε τη βάση για αρκετά άλλο πειραματικά NLIDBs.

Στα μέσα της δεκαετίας του '80 τα NLIDBs ήταν ένας πολύ δημοφιλής τομέας της έρευνας, και πολυάριθμα συστήματα πρωτοτύπων εφαρμόζονταν. Ένα μεγάλο μέρος της έρευνας εκείνης της εποχής αφιερώθηκε στα ζητήματα φορητότητας.

Παραδείγματος χάριν, το Team [36] σχεδιάστηκε για να είναι εύκολα διαμορφώσιμο από τους administrators βάσεων δεδομένων χωρίς τη γνώση των NLIDBs.

Το Ask [37] επέτρεπε στους τελικούς χρήστες να διδάξουν το σύστημα νέες λέξεις και έννοιες σε οποιοδήποτε σημείο κατά τη διάρκεια της αλληλεπίδρασης. Το Ask ήταν πραγματικά ένα πλήρες σύστημα διαχείρισης πληροφοριών, που παρείχε την ενσωματωμένη βάση δεδομένων του, και τη δυνατότητα να αλληλεπιδράσει με πολλαπλές εξωτερικές βάσεις δεδομένων, προγράμματα ηλεκτρονικού ταχυδρομείου, και άλλες εφαρμογές υπολογιστών. Όλες οι εφαρμογές που συνδέονταν για να ρωτήσουν ήταν προσιτές στον τελικό χρήστη μέσω των αιτημάτων φυσικής γλώσσας. Ο χρήστης δήλωνε τα αιτήματά του στα αγγλικά, και το Ask παρήγαγε αιτήματα στα κατάλληλα συστήματα.

Το Janus [38] είχε δυνατότητες ώστε να διασυνδέεται στα πολλαπλά βασικά συστήματα (βάσεις δεδομένων, experts συστήματα, συσκευές γραφικής παράστασης, κ.λπ). Όλα τα συστήματα θα μπορούσαν να συμμετέχουν στην αξιολόγηση ενός αιτήματος φυσικής γλώσσας, χωρίς ο χρήστης να γίνεται ενήμερος για την ετερογένεια του γενικού συστήματος. Το Janus ήταν επίσης ένα από τα λίγα συστήματα που υποστήριζαν χρονικές (temporal) ερωτήσεις.

Αν και μερικά από τα πολυάριθμα NLIDBs που αναπτύχθηκαν στα μέσα της δεκαετίας του '80 κατέδειξαν εντυπωσιακά χαρακτηριστικά σε ορισμένους τομείς εφαρμογής, τα NLIDBs δεν κέρδισαν την αναμενόμενη γρήγορη και ευρεία εμπορική αποδοχή. Παραδείγματος χάριν, το 1985 το Ovum Ltd. [39] προέβλεπε ότι "μέχρι το 1987 μια διεπαφή φυσικής γλώσσας πρέπει να είναι μια τυποποιημένη επιλογή για τους χρήστες των DBMS και των λογισμικού τύπου κέντρου πληροφόρησης και θα υπάρξει μια λογική επιλογή των εναλλακτικών λύσεων." Από τότε, διάφορα εμπορικά διαθέσιμα NLIDBs έχουν εμφανιστεί, και μερικά από αυτά θεωρούνται ότι είναι εμπορικά επιτυχή. Εντούτοις, τα NLIDBs αντιμετωπίζονται ακόμα ως έρευνα ή ως «εξωτικά» συστήματα, παρά μια τυποποιημένη επιλογή για τις βάσεις δεδομένων, και η χρήση τους δεν είναι βεβαίως διαδεδομένη. Η ανάπτυξη των επιτυχών εναλλακτικών λύσεων NLIDBs, όπως οι γραφικές και οι βασισμένες σε φόρμες (form based) διεπαφές, και τα εγγενή προβλήματα των NLIDBs (και τα δύο αναφέρονται στη συγκεκριμένη ενότητα) είναι πιθανώς οι κύριοι λόγοι για την έλλειψη αποδοχής NLIDBs.

Τα τελευταία χρόνια έχει υπάρξει μια σημαντική μείωση στον αριθμό εγγράφων (papers) για τα NLIDBs που δημοσιεύονται ετησίως. Παρόλα αυτά, τα NLIDBs

συνεχίζουν να εξελίσσονται, υιοθετώντας τις προόδους στο γενικό τομέα επεξεργασίας φυσικής γλώσσας, που ερευνούν τις αρχιτεκτονικές που μετασχηματίζουν τα NLIDBs σε λογικούς πράκτορες (reasoning agents), και ενοποιούν τη γλώσσα και τα γραφικά για να εκμεταλλευτούν τα πλεονεκτήματα και των δύο μορφών. Επίσης, έχουν εμφανιστεί τα γενικής χρήσης γλωσσικά front-ends. Αυτά είναι γενικής χρήσης συστήματα που χαρτογραφούν τη φυσική γλώσσα που εισάγεται σε εκφράσεις μιας λογικής γλώσσας (π.χ. το Cle system [40]). Αυτά τα γενικά front-ends μπορούν να μετατραπούν σε NLIDBs, με την ένωση των πρόσθετων ενοτήτων που αξιολογούν τις λογικές εκφράσεις σε μια βάση δεδομένων.

Η βάση δεδομένων είναι δομημένη σύμφωνα με κάποιο πρότυπο δεδομένων, και το NLIDB έχει ως σκοπό να λειτουργήσει με αυτό το πρότυπο δεδομένων. Τα συστήματα βάσεων δεδομένων έχουν εξελιχθεί επίσης πολύ κατά τη διάρκεια των τελευταίων δεκαετιών. Πολλά από τα βασικά συστήματα βάσεων δεδομένων των πρώτων NLIDBs δεν θα άξιζαν να ονομάζονται συστήματα βάσεων δεδομένων με τα σημερινά πρότυπα.

Τις πρώτες ημέρες των συστημάτων βάσεων δεδομένων, δεν υπήρχε καμία έννοια των απλών τελικών χρηστών που να έχει πρόσβαση στα στοιχεία άμεσα, αυτό γινόταν από έναν ειδικό προγραμματιστή που έγραφε συγκεκριμένα προγράμματα υπολογιστών.

Ο λόγος για αυτό ήταν η φύση πλοήγησης (navigational nature) του μοντέλου δεδομένων που χρησιμοποιήθηκε από αυτά τα πρώτα συστήματα βάσεων δεδομένων. Όχι μόνο έπρεπε οι χρήστες να ξέρουν για τη δομή των στοιχείων στην εφαρμογή αλλά έπρεπε επίσης να ξέρουν πολλά τεχνάσματα προγραμματισμού για να παίρνουν τα δεδομένα. Η ανάπτυξη του σχεσιακού μοντέλου δεδομένων στη δεκαετία του '70 [41] άσκησε σημαντική επίδραση στα συστήματα βάσεων δεδομένων. Στο σχεσιακό πρότυπο, η μόνη δομή αποθήκευσης είναι ο πίνακας, και αυτό ήταν κάτι που ακόμη και οι απλοί χρήστες θα μπορούσαν να καταλάβουν. Οι σχετικά απλές γλώσσες διατύπωσης ερωτήσεων, όπως η SQL [42], αναπτύχθηκαν για αυτήν την κατηγορία χρηστών.

Αυτήν την περίοδο, υπάρχουν δύο σημαντικές εξελίξεις στην τεχνολογία βάσεων δεδομένων που θα ασκήσουν επίδραση στα NLIDBs. Ο πρώτος είναι η αυξανόμενη σημαντικότητα των αντικειμενοστραφών συστημάτων βάσεων δεδομένων (object-oriented database systems), και ο δεύτερος είναι η τάση από τις σχεσιακές βάσεις δεδομένων προς τις πιο σύνθετες δομές αποθήκευσης έτσι ώστε να διευκολυνθεί η

προηγμένη μοντελοποίηση δεδομένων. Και οι δύο απεικονίζουν μια τάση να επικεντρωθούν στους νέους σύνθετους τομείς εφαρμογής βάσεων δεδομένων, όπως η διαχείριση δικτύων και ο computer-aided σχεδιασμός . Η άμεση πρόσβαση στη βάση δεδομένων πραγματοποιείται συχνά μέσω ενός επιπέδου της εφαρμογής του λογισμικού.

2.3 Γλώσσα περιγραφής οντολογιών ιστού –Ontology Web Language

2.3.1 Γενικά

Η γλώσσα περιγραφής οντολογιών ιστού (Ontology Web Language / στο εξής OWL[18]) έχει υιοθετηθεί από το W3C (World Wide Web Consortium) ως η πρότυπη γλώσσα περιγραφής οντολογιών. Η OWL προορίζεται να χρησιμοποιηθεί στις περιπτώσεις κατά τις οποίες η πληροφορία που περιέχεται σε έγγραφα χρειάζεται να επεξεργαστεί από εφαρμογές, σε αντίθεση με τις περιπτώσεις κατά τις οποίες το περιεχόμενο χρειάζεται μόνο να παρουσιαστεί στους ανθρώπους. Η OWL χρησιμοποιείται για την περιγραφή των εννοιών ενός πεδίου γνώσης καθώς και των σχέσεων μεταξύ των εννοιών αυτών. Παρακάτω θα αναλύσουμε κάποιες βασικές έννοιες που πραγματεύεται η OWL και οι οποίες έχουν επεξεργαστεί – χρησιμοποιηθεί στη παρούσα διπλωματική εργασία.

2.3.2 Οντολογίες (Ontologies)

Μία οντολογία προσδιορίζει τα διάφορα στοιχεία τα οποία μπορούν να χρησιμοποιηθούν στην περιγραφή και την αναπαράσταση ενός πεδίου γνώσης. Περιλαμβάνει προσδιορισμούς βασικών εννοιών ενός πεδίου γνώσης και των μεταξύ τους σχέσεων, χρησιμοποιώντας στοιχεία όπως είναι οι κλάσεις (classes), τα στιγμιότυπα (individuals), οι ιδιότητες (properties) κτλ. Γενικεύοντας, μία οντολογία περιέχει ορισμούς και ιεραρχίες εννοιών (π.χ. κλάσεις). Παραδείγματα τέτοιων ορισμών είναι ο ορισμός της έννοιας “person”, ο ορισμός ιδιοτήτων (properties) εννοιών (π.χ. το name ή το age του person), ο ορισμός σχέσεων μεταξύ εννοιών (π.χ. ένα person *“has_friend”* κάποιο άλλο person), και ο ορισμός στιγμιότυπων (things ή individuals) εννοιών (π.χ. ο *“John”* ο οποίος *has_friend* τον *“Nick”*). Αυτές είναι οι βασικές έννοιες της περιγραφής μίας οντολογίας. Βασισμένες σε αυτές τις έννοιες

μοντελοποίησης μπορούν να οριστούν άλλες δομές όπως είναι οι περιορισμοί (restrictions), οι λίστες (lists) κτλ.

2.3.3 Κλάσεις

Μία θεμελιώδης έννοια στην OWL είναι η κλάση (class). Η κλάση αναπαριστά την ομαδοποίηση παρόμοιων στοιχείων (που ονομάζονται στιγμιότυπα (things ή individuals)) μέσα σε μία οντολογία. Για τον λόγο αυτό κάθε κλάση συνδέεται με ένα σύνολο από στιγμιότυπα, που ονομάζεται επέκταση της κλάσης (class extension). Για παράδειγμα σε μία οντολογία μπορεί να οριστεί η κλάση “Color_Of_wine”, η οποία να αναπαριστά την ομαδοποίηση των στιγμιότυπων “Red”, “White” και “Rose”. Είναι εύκολο να καταλάβει κανείς πως στο προηγούμενο παράδειγμα πρώτα δημιουργείται η κλάση και στην συνέχεια τα στιγμιότυπά της. Η σημασιολογία μίας κλάσης έχει σχέση αλλά δεν ταυτίζεται με την επέκτασή της (class extension). Αυτό σημαίνει πως δύο κλάσεις μπορεί ενώ έχουν την ίδια επέκταση (class extension) και να είναι διαφορετικές η μία από την άλλη σημασιολογικά.

2.3.4 Ιδιότητες μεταξύ κλάσεων (object properties)

Οι ιδιότητες μεταξύ κλάσεων είναι οι ιδιότητες οι οποίες έχουν σαν αντικείμενο (range) μία άλλη κλάση. Αυτό το είδος ιδιοτήτων συσχετίζει στιγμιότυπα μίας κλάσης με τα στιγμιότυπα μιας άλλης κλάσης. Είναι δυνατό να οριστεί πως μία ιδιότητα αυτού του είδους (object property) είναι αντίστροφη κάποιας άλλης ιδιότητας του ίδιου είδους, ένα inverse property δηλαδή. Για παράδειγμα, η ιδιότητα “made_of_grape” της κλάσης “Wine” σε μία οντολογία οινολογίας, είναι αντίστροφη της ιδιότητας “convert_to_wine” της κλάσης “Grape”. Η σχέση «inverseOf» μεταξύ των ιδιοτήτων είναι συμμετρική. Αυτό σημαίνει πως αν η ιδιότητα Α είναι αντίστροφη της ιδιότητας Β, τότε η ιδιότητα Β είναι αντίστροφη της ιδιότητας Α.

2.3.5 Ιδιότητες μεταξύ κλάσεων και συγκεκριμένων τιμών (Datatype Property)

Οι ιδιότητες μεταξύ κλάσεων και συγκεκριμένων τιμών είναι οι ιδιότητες οι οποίες παίρνουν συγκεκριμένες τιμές σαν αντικείμενο (range). Για τον σκοπό αυτό ορίζεται η έννοια του Εύρους Τιμών (DataRange).Το Εύρους Τιμών είναι μία αφηρημένη δομή προσδιορισμού του γεγονότος ότι το εύρος τιμών μίας ιδιότητας αυτού του είδους (datatype property) μπορεί να είναι είτε ένας συγκεκριμένος τύπος δεδομένων (π.χ. ακέραιος), είτε ένα σύνολο από τιμές. Για παράδειγμα, ο χρήστης μπορεί να

ορίσει πως η ιδιότητα (datatype property) "has_Year" της κλάσης "Year_Of_Vintage" (σε μία οντολογία οινολογίας) είναι ένας θετικός ακέραιος αριθμός (π.χ. 1998), ή πως η ίδια ιδιότητα μπορεί να πάρει τιμή "1998" ή "1999" ή "2000".

2.3.6 Στιγμιότυπα (Things ή Individuals)

Οι κλάσεις παρέχουν έναν αφαιρετικό μηχανισμό ομαδοποίησης στοιχείων με παρόμοια χαρακτηριστικά. Κάθε κλάση συνδέεται με ένα σύνολο από στιγμιότυπα, το οποίο ονομάζεται επέκταση της κλάσης (class extension). Το στοιχείο «Thing» είναι μία αφηρημένη κλάση που χρησιμοποιείται για την αναπαράσταση όλων των στιγμιότυπων (είτε πρόκειται για στιγμιότυπα κλάσεων, είτε για στιγμιότυπα ιδιοτήτων) σε μία οντολογία. Επιπλέον, ένα στιγμιότυπο μπορεί να είναι διαφορετικό από κάποιο άλλο στιγμιότυπο. Για παράδειγμα το στιγμιότυπο «Sweet» της κλάσης «Sugar_Of_Wine» σε μία οντολογία οινολογίας είναι διαφορετικό από το στιγμιότυπο «Dry» της ίδιας κλάσης. Ακόμα ένα στιγμιότυπο μπορεί να είναι ίδιο με κάποιο άλλο στιγμιότυπο.

2.4 Το MPEG-7

2.4.1 Γενικά

Το MPEG-7[15] είναι ένα πρότυπο μεταδεδομένων, δημιουργημένο από το Moving Picture Experts Group, για να καλύψει τις αυξανόμενες ανάγκες για μια αυστηρά ορισμένη περιγραφή οπτικοακουστικού περιεχομένου σε περιβάλλον πολυμέσων, με όλα τα πλεονεκτήματα που προσφέρει κάτι τέτοιο στην ταχύτερη ανάπτυξη συστημάτων ανάκτησης και πλοήγησης στο υλικό καθώς και τη δυνατότητα επικοινωνίας μεταξύ διαφορετικών συστημάτων. Δεν έρχεται να αντικαταστήσει προηγούμενα πρότυπα για οπτικοακουστικό υλικό, όπως το MPEG-1, MPEG-2 και MPEG-4, αλλά να προσφέρει μία υψηλότερου επιπέδου περιγραφή με δεικτοδότηση προς αυτά. Για παράδειγμα, ίσως η περιγραφή ενός σχήματος σε MPEG-4 να είναι χρήσιμη και στο MPEG-7, όπως και ένα διάνυσμα κίνησης από τα MPEG-1 ή και MPEG-2.

Με τον όρο 'περιγραφή οπτικοακουστικού περιεχομένου σε περιβάλλον πολυμέσων' αναφερόμαστε στην περιγραφή εικόνας, ήχου, λόγου, γραφικών, τρισδιάστατων μοντέλων, video, αλλά και την περιγραφή για το πως κάποια από τα

παραπάνω συσχετίζονται μεταξύ τους για να παράγουν το τελικό αποτέλεσμα. Η πληροφορία που παρέχεται από το MPEG-7 ανάγεται στους τομείς:

- Της δημιουργίας των διαδικασιών παραγωγής του υλικού (π.χ. σκηνοθέτης, τίτλος, ταινία μικρού μήκους)
- Του συσχετιζόμενου με το περιεχόμενο υλικό (copyright, ιστορία χρήσης, πρόγραμμα μετάδοσης)
- Των χαρακτηριστικών αποθήκευσης του περιεχομένου (format, κωδικοποίηση)
- Της δομής του περιεχομένου και των επιμέρους τμημάτων του (σκηνές, τμηματοποίηση του video)
- Των χαμηλού επιπέδου χαρακτηριστικών του περιεχομένου (χρώμα, υφή της εικόνας)
- Της σημασιολογικής πληροφορίας από την 'πραγματικότητα' που απεικονίζεται στο περιεχόμενο (αντικείμενα, πρόσωπα, σχέσεις μεταξύ τους)
- Της πλοήγησης στο περιεχόμενο
- Των συλλογών αντικειμένων
- Της αλληλεπίδρασης του χρήστη με το περιεχόμενο.

Όλες αυτές οι περιγραφές δομούνται σε XML [22]. έγγραφα που συνοδεύουν και δεικτοδοτούν το υλικό και που, όπως προαναφέρθηκε, υπακούουν σε ένα σχήμα δομημένο με βάση το XML Schema.

2.4.2 Η MP7QL γλώσσα ερωτήσεων

2.4.2.1 Γενικά

Η MP7QL[13]. είναι μία γλώσσα ερωτήσεων η οποία σχεδιάστηκε για να απευθύνει ερωτήσεις (queries) σε MPEG-7 περιγραφές . Γενικότερα, ικανοποιεί τις MP7QF[12] απαιτήσεις, μοντέλο δεδομένων της αποτελεί το MPEG-7 και είναι εκφρασμένη σε XML schema και OWL σύνταξη.

Η MP7QL επιτρέπει ανάκτηση και φιλτράρισμα MPEG-7 περιγραφών με ομοιόμορφο και διαφανή τρόπο, μπορεί να δημιουργήσει ερώτηση για κάθε στοιχείο

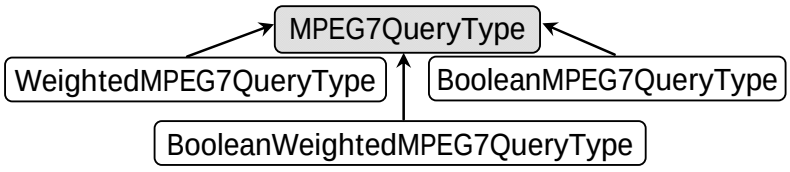
μιας MPEG-7 περιγραφής αντικειμένων πολυμέσων και υποστηρίζει το σαφή προσδιορισμό boolean τελεστών και βαθμών προτίμησης.

Τέλος, η μορφή της εξόδου (query output format) της MP7QL είναι το MPEG-7 και οι MP7QL ερωτήσεις χρησιμοποιούν τις προτιμήσεις του χρήστη και το ιστορικό χρήσης ως γενικευμένο πλαίσιο (context) υποβολής των ερωτήσεων.

2.4.2.2 Το input query format της MP7QL γλώσσας ερωτήσεων

Παρουσιάζεται σε αυτή την ενότητα το input format της MP7QL γλώσσας ερωτήσεων που επιτρέπει τη διατύπωση ερωτήσεων σε MPEG-7 περιγραφές (όλες οι λεπτομέρειες της MP7QL βρίσκονται – αναλύονται εκτενώς στο [13]). Το input format της MP7QL επιτρέπει τη διατύπωση ερωτήσεων σε κάθε πτυχή μιας περιγραφής αντικειμένου MPEG-7 οπτικοακουστικού υλικού. Οι ερωτήσεις MP7QL μπορούν να χρησιμοποιήσουν τις προτιμήσεις χρηστών (user preferences) και το ιστορικό χρήσης (usage history) ως πλαίσιο, επιτρέποντας κατά συνέπεια την εξατομικευμένη ανάκτηση οπτικοακουστικού (multimedia) περιεχομένου.

Το θεμελιώδες στοιχείο της MP7QL είναι το MP7QL query, το οποίο αντιπροσωπεύεται από τον αφηρημένο τύπο MPEG7QueryType. Η γλώσσα διατύπωσης ερωτήσεων MP7QL επιτρέπει τη ρητή προδιαγραφή των boolean operators και των τιμών προτίμησης (preference values) για τα στοιχεία ερώτησης MP7QL. Τρεις υποκατηγορίες MPEG7QueryType έχουν καθοριστεί για την αντιπροσώπευση όλων των πιθανών τύπων ερωτήσεων, όπως φαίνεται στο σχήμα 1: ***(α)*** Το WeightedMPEG7QueryType, το οποίο αντιπροσωπεύει τις ερωτήσεις με τις ρητές τιμές προτίμησης. Οι τιμές προτίμησης είναι ακέραιοι αριθμοί στη σειρά [- 100 ..100], με την προκαθορισμένη αξία 10. Ο τύπος, η σειρά και η προκαθορισμένη αξία των τιμών προτίμησης είναι συμβατοί με τις τιμές προτίμησης που χρησιμοποιούνται στα MPEG-7 FASPs. ***(β)*** Το BooleanMPEG7QueryType, το οποίο αντιπροσωπεύει τις ερωτήσεις με τους ρητούς boolean operators (AND/OR/NOT) και ***(γ)*** Το BooleanWeightedMPEG7QueryType, το οποίο αντιπροσωπεύει τις ερωτήσεις με τις ρητές τιμές προτίμησης (preference values) και τους boolean operators.



Εικόνα 1: Το MP7QL Query Type Hierarchy

Ένα MP7QL query έχει μία σύνταξη SELECT-FROM-WHERE και τυπικά περιγράφεται με τη σύνταξη κανονικών εκφράσεων (regular expressions) του (1):

Q=[Select][From][Where][OrderBy][GroupBy] (1)

Το προαιρετικό SELECT στοιχείο μιας ερώτησης MP7QL επιτρέπει την προδιαγραφή των and/or των MPEG-7 περιγραφών που θα επιστραφούν στα αποτελέσματα ερώτησης Εάν το SELECT στοιχείο δεν είναι παρόν σε μια ερώτηση MP7QL, τα αποτελέσματα ερώτησης θα διαμορφωθούν με τον τρόπο προεπιλογής (default way). Το SELECT στοιχείο μιας ερώτησης MP7QL περιγράφεται τυπικά χρησιμοποιώντας την κανονική σύνταξη έκφρασης (2)

Select=Item [format][transformationRules] [maxItems][numOfPageItems] [page][timeLimit]* (2)

Τα *Item* στοιχεία αναπαριστούν, υπό μορφή εκφράσεων XPath, τα στοιχεία και/ή τις ιδιότητες των MPEG -7 περιγραφών που πρέπει να επιστραφούν για κάθε ένα από τα αποτελέσματα ερώτησης. Το προαιρετικό στοιχείο *format* αντιπροσωπεύει το URI του αρχείου, όπου η δομή του σχήματος επίδειξης παραγωγής (δηλαδή το σχήμα με το οποίο τα αποτελέσματα ερώτησης θα επιδειχθούν στην τερματική συσκευή του χρήστη) διευκρινίζεται και έχει ως προκαθορισμένη αξία το URI του σχήματος παραγωγής ερώτησης προεπιλογής. Το προαιρετικό στοιχείο *transformationRules* αντιπροσωπεύει το URI του XSL stylesheet [52] που πρέπει να εφαρμοστεί στο standart MP7QL output προκειμένου να παρουσιαστεί σύμφωνα με ένα διαφορετικό format. Το προαιρετικό στοιχείο *maxItems* αντιπροσωπεύει το μέγιστο αριθμό των αποτελεσμάτων ερώτησης που θα επιστραφούν στο χρήστη και έχει προκαθορισμένη αξία τη τιμή ‘απεριόριστο’. Το προαιρετικό στοιχείο *numOfPageItems* αντιπροσωπεύει τον αριθμό των αποτελεσμάτων ερώτησης που θα επιδειχθούν σε κάθε σελίδα αποτελέσματος και έχει τον αριθμό 10 ως προκαθορισμένη αξία. Το προαιρετικό στοιχείο *timeLimit* αντιπροσωπεύει το χρονικό όριο στα δευτερόλεπτα μέχρι το οποίο η ερώτηση πρέπει να απαντηθεί και έχει τον αριθμό 300 ως προκαθορισμένη αξία. Το προαιρετικό στοιχείο *page* διευκρινίζει ποια σελίδα αποτελέσματος πρέπει να επιστραφεί στο χρήστη και έχει τον αριθμό 1 ως προκαθορισμένη αξία.

Το προαιρετικό στοιχείο From μιας ερώτησης MP7QL επιτρέπει την προδιαγραφή του τύπου των MPEG -7 περιγραφών στις οποίες η ερώτηση θα τεθεί και περιγράφεται τυπικά χρησιμοποιώντας την κανονική σύνταξη έκφρασης (3).

$$From = FromItem* \quad (3)$$

Τα στοιχεία *FromItem* μπορούν να πάρουν προκαθορισμένες τιμές που διευκρινίζουν τον τύπο των MPEG -7 περιγραφών στις οποίες η ερώτηση θα τεθεί. Οι επιτρεπόμενες τιμές του στοιχείου *FromItem* μπορεί να είναι: **(α)** Τα ονόματα των MPEG -7 τύπων περιγραφής πολυμέσων, που περιλαμβάνουν "MultimediaCollectionType", "ImageType", "VideoType", "AudioType", "AudioVisualType", "MultimediaType", "SignalType", "AnalyticEditedVideoType", "InkContentType". Αυτές οι τιμές δηλώνουν ότι η ερώτηση πρέπει να τεθεί στις περιγραφές πολυμέσων του τύπου που διευκρινίζεται. Παραδείγματος χάριν, ο τύπος "ImageType" πρέπει να χρησιμοποιηθεί στην ερώτηση “give me the images where Marques is shown”. **(β)** “AllMultimediaDescriptions”, το οποίο είναι η προκαθορισμένη τιμή και δηλώνει ότι η ερώτηση πρέπει να τεθεί σε όλους τους προηγούμενους τύπους περιγραφής πολυμέσων προκειμένου να βρεθεί το περιεχόμενο πολυμέσων που ικανοποιεί τα ειδικά κριτήρια. **(γ)** “SemanticEntity-Definition”, το οποίο δηλώνει ότι η ερώτηση πρέπει να τεθεί στις σημασιολογικές οντότητες που ικανοποιούν τα ειδικά κριτήρια (για παράδειγμα, “give me the players of the soccer team Barcelona”) και **(δ)** “Ontology”, που δηλώνει ότι η ερώτηση πρέπει να τεθεί στις οντολογίες περιοχών που εκφράζονται χρησιμοποιώντας τη σύνταξη MPEG-7 (παραδείγματος χάριν, , “give me the subclasses of *SoccerPlayer*”). Το προαιρετικό στοιχείο *OrderBy* μιας ερώτησης MP7QL επιτρέπει την προδιαγραφή των κριτηρίων για τη ταξινόμηση των αποτελεσμάτων και περιγράφεται τυπικά χρησιμοποιώντας την κανονική σύνταξη έκφρασης (4).

$$OrderBy=Criterion* \quad (4)$$

Τα Criterion στοιχεία του OrderBy αντιπροσωπεύουν τη διάταξη των κριτηρίων. Ένα Criterion στοιχείο περιγράφεται τυπικά χρησιμοποιώντας την κανονική σύνταξη έκφρασης (5).

$$Criterion=Item\ [priority][order] \quad (5)$$

Το στοιχείο Item του Criterion αντιπροσωπεύει, ως έκφραση XPath, ένα στοιχείο ή μια ιδιότητα των MPEG-7 περιγραφών, στις οποίες η ταξινόμηση θα βασιστεί. Το προαιρετικό στοιχείο *priority* αντιπροσωπεύει την προτεραιότητα των στοιχείων /ιδιοτήτων στη διάταξη και έχει 0 ως προκαθορισμένη αξία. Το προαιρετικό στοιχείο

order αντιπροσωπεύει τον τύπο (ανοδικό ή καθοδικό) της ταξινόμησης βασισμένης στο τρέχων στοιχείο/ ιδιότητα και έχει τη τιμή "ανοδικό" ως προκαθορισμένη αξία.

Το προαιρετικό στοιχείο *GroupBy* μιας ερώτησης MP7QL διευκρινίζει, στη μορφή μιας έκφρασης XPath, τις ιδιότητες ή το στοιχείο που θα χρησιμοποιηθούν για την ομαδοποίηση των αποτελεσμάτων ερώτησης.

Το προαιρετικό WHERE στοιχείο μιας ερώτησης MP7QL επιτρέπει την έκφραση των όρων ερώτησης που τίθενται από το χρήστη. Η δομή του στοιχείου WHERE είναι διαφορετικό για τους διαφορετικούς τύπους των ερωτήσεων MP7QL. Ειδικότερα:

1. Το WHERE στοιχείο μιας ερώτησης MP7QL με τις ρητές τιμές προτίμησης περιγράφεται τυπικά χρησιμοποιώντας την κανονική σύνταξη έκφρασης (6).

$$WWhere=(WQS\ pv)* \quad (6)$$

το pv είναι μια ρητή αξία προτίμησης και το WQS είναι μια προδιαγραφή ερώτησης με τις ρητές τιμές προτίμησης (που περιγράφονται τυπικά (9)). Η προδιαγραφή ερώτησης αντιπροσωπεύει τους όρους ερώτησης που τίθενται από το χρήστη.

2. Το WHERE στοιχείο μιας ερώτησης MP7QL με τους ρητούς boolean operators (BWhere) περιγράφεται τυπικά χρησιμοποιώντας την κανονική σύνταξη έκφρασης (7).

$$BWhere=BQS[NOT]\ ((AND|OR)\ BQS\ [NOT])* \quad (7)$$

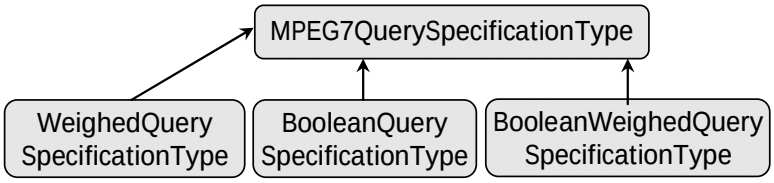
Το *BQS* είναι μια προδιαγραφή ερώτησης με boolean operators

3. Το WHERE στοιχείο μιας ερώτησης MP7QL με τις ρητές τιμές προτίμησης και τους boolean operators (BWWhere) περιγράφεται τυπικά χρησιμοποιώντας την κανονική σύνταξη έκφρασης (8).

$$BWWhere=BWQS\ pv\ ((AND|OR)\ BWQS\ pv)* \quad (8)$$

Το BWQS είναι μια προδιαγραφή ερώτησης με τις ρητά διευκρινισμένες τιμές προτίμησης και τους boolean operators.

Οι προδιαγραφές ερώτησης MP7QL αντιπροσωπεύονται από τον αφηρημένο τύπο MPEG7QuerySpecificationType, ο οποίος είναι εξειδικευμένος σύμφωνα με την παρουσία/απουσία ρητών boolean operators or/and τιμών προτίμησης όπως φαίνεται στο σχήμα 2.



Εικόνα 2: Το MP7QL Query Specification Type Hierarchy

Το WeightedQuerySpecificationType αντιπροσωπεύει τις προδιαγραφές ερώτησης με τις ρητές τιμές προτίμησης, το BooleanQuery-SpecificationType αντιπροσωπεύει τις προδιαγραφές ερώτησης με τους boolean operators και ο BooleanWeightedQuerySpecification-Type αντιπροσωπεύει τις προδιαγραφές ερώτησης με τους boolean operators και τις τιμές προτίμησης.

Οι προδιαγραφές ερώτησης (query specification) έχουν σχεδιαστεί με σκοπό να επιτρέψουν την έκφραση συνθηκών σε κάθε πτυχή ενός αντικειμένου πολυμέσων που έχει περιγραφθεί χρησιμοποιώντας το MPEG-7, έτσι ώστε η MP7QL να μπορεί να χρησιμοποιηθεί για διατύπωση ερωτήσεων σε οποιασδήποτε περιγραφής αντικειμένου MPEG -7 πολυμέσων. Κατά συνέπεια, κάθε στοιχείο μιας περιγραφής αντικειμένου MPEG -7 πολυμέσων έχει ένα αντίστοιχο στοιχείο προδιαγραφών ερώτησης στις προδιαγραφές ερώτησης MP7QL. Το αντίστοιχο στοιχείο προδιαγραφών ερώτησης χρησιμοποιείται για να διευκρινίσει τους όρους που πρέπει να κρατήσουν στις τιμές των MPEG-7 στοιχείων των τμημάτων που ανακτώνται. Προκειμένου να ικανοποιηθεί η απαίτηση των MPEG-7 Query Format Requirements [53] που δηλώνουν ότι η ύπαρξη MPEG -7 FASPs πρέπει να είναι έγκυρες προδιαγραφές ερώτησης, γίνεται χρήση του ίδιου σχεδίου ονομασίας και δακτυλογράφησης με τα MPEG -7 FASPs για όλα τα στοιχεία που υπάρχουν στα MPEG -7 FASPs . Το υπόλοιπο των στοιχείων προδιαγραφών ερώτησης MP7QL ακολουθεί το σχέδιο ονομασίας και δακτυλογράφησης των MPEG -7 περιγραφών αντικειμένου πολυμέσων. Οι προδιαγραφές ερώτησης MP7QL περιλαμβάνουν τα ακόλουθα (προαιρετικά) στοιχεία:

- Το στοιχείο MediaProfile (MP), όπου οι χρήστες διευκρινίζουν τους όρους στα χαρακτηριστικά γνωρίσματα μέσων (δηλ. σχήμα μέσων, ποιότητα κ.λπ..) από τα αποτελέσματα ερώτησης.
- Το στοιχείο CreationPreferences (CrP), όπου οι χρήστες διευκρινίζουν τους όρους στις λεπτομέρειες δημιουργιών των ζητούμενων στοιχείων (δηλ. τίτλος, δημιουργοί, σχετικοί υλικό κ.λπ.).

- Το στοιχείο ClassificationPreferences (ClP), όπου οι χρήστες διευκρινίζουν τους όρους στην ταξινόμηση των ζητούμενων στοιχείων (δηλ. γλώσσα, ύφος, κ.λπ.)
- Το στοιχείο SourcePreferences (SoP), όπου οι χρήστες διευκρινίζουν τους όρους στη διάδοση των ζητούμενων στοιχείων (δηλ. πηγή διάδοσης, σχήμα κ.λπ.).
- Το στοιχείο MediaIdentification (MI), όπου οι χρήστες διευκρινίζουν τους όρους για τον προσδιορισμό των αποτελεσμάτων ερώτησης.
- Το στοιχείο MediaLocator (ML), όπου οι χρήστες διευκρινίζουν τους όρους στα πραγματικά πολυμέσα.
- Το σημασιολογικό στοιχείο (SeP), όπου οι χρήστες διευκρινίζουν τους όρους στη σημασιολογία του περιεχομένου των ζητούμενων στοιχείων. Οι όροι στο σημασιολογικό περιεχόμενο είναι πολύ σημαντικοί στα event-based περιβάλλοντα (όπως στα sports). Επιπλέον, το σημασιολογικό στοιχείο χρησιμοποιείται για ερωτήσεις γύρω από επαναχρησιμοποιήσιμες σημασιολογικές οντότητες και για οντολογίες περιοχών που εκφράζονται χρησιμοποιώντας τη σύνταξη MPEG -7.
- Το στοιχείο StructuralUnit (SU), όπου οι χρήστες διευκρινίζουν τους όρους στη δομή των ζητούμενων στοιχείων
- Το στοιχείο PreferenceCondition (PC), όπου οι χρήστες δηλώνουν ποιους όρους πρέπει να κρατήσουν, από την άποψη του τόπου και χρόνου, για την προδιαγραφή ερώτησης που λαμβάνεται υπόψη.
- Το στοιχείο UsageInformation (UI), όπου οι χρήστες διευκρινίζουν τους όρους στη χρήση των ζητούμενων στοιχείων (δηλ. δικαιώματα, διαθεσιμότητα κ.λπ.).
- Το στοιχείο MatchingHint (MH), όπου οι χρήστες διευκρινίζουν τους όρους στο ταίριασμα των χαμηλού επιπέδου χαρακτηριστικών γνωρισμάτων με τα χαρακτηριστικά γνωρίσματα στοιχείων πολυμέσων.
- Το στοιχείο PointOfView (PoV), όπου οι χρήστες διευκρινίζουν τους όρους στη σημαντικότητα του περιεχομένου πολυμέσων από συγκεκριμένες απόψεις.
- Το στοιχείο RelatedMaterial (RM), όπου οι χρήστες διευκρινίζουν τους όρους στο σχετικό υλικό των ζητούμενων στοιχείων.

- Το στοιχείο Relation (R), όπου οι χρήστες διευκρινίζουν τις σχέσεις των ζητούμενων στοιχείων με άλλα μέσα ή στοιχεία μεταδεδομένων. Μια σχέση μπορεί να είναι κατευθυνόμενη ή όχι (directed, undirected) και χαρακτηρίζει έναν τύπο σχέσης, το στόχο (traget) και την πηγή (source) της σχέσης (relationship). Οι τυποποιημένοι MPEG-7 τύποι σχέσης είναι περισσότεροι από 100 και είναι ταξινομημένοι σε: *(α)* Βασικοί τύποι σχέσης (equals, refines κ.λπ.), οι οποίοι διευκρινίζονται στο BaseRelation *CS* *(β)* Τύποι σχέσης κόμβων γραφικών παραστάσεων (Graph node relationship types), οι οποίοι διευκρινίζονται στο GraphRelation *CS* (identity, equivalent κ.λπ.) *(γ)* Σημασιολογικοί τύποι σχέσης (Semantic relationship types), οι οποίοι διευκρινίζονται στο SemanticRelation *CS* (agent, causer κ.λπ.) *(δ)* Χωρικοί τύποι σχέσης (Spatial relationship types), οι οποίοι διευκρινίζονται στο SpatialRelation *CS* (over, north κ.λπ.) και *(ε)* χρονικοί τύποι σχέσης (Temporal relationship types), οι οποίοι διευκρινίζονται στο TemporalRelation *CS* (precedes, overlaps κ.λπ.)
- Το στοιχείο TextAnnotation (TA), όπου οι χρήστες διευκρινίζουν τους όρους στους κειμενικούς σχολιασμούς των ζητούμενων στοιχείων
- Το στοιχείο VisualDescriptor (VD), όπου οι αναφορές διευκρινίζονται στα χαμηλού επιπέδου οπτικά χαρακτηριστικά γνωρίσματα που κτίζονται σύμφωνα με τους περιγραφείς των MPEG-7 Visual [54] και που πρέπει να αντιστοιχηθούν με τα αντίστοιχα χαμηλού επιπέδου οπτικά χαρακτηριστικά γνωρίσματα των ζητούμενων στοιχείων μέσων.
- Το στοιχείο VisualDescriptionScheme (VDS), όπου οι αναφορές διευκρινίζονται στα χαμηλού επιπέδου οπτικά χαρακτηριστικά γνωρίσματα που κτίζονται σύμφωνα με τα σχέδια περιγραφής των MPEG-7 Visual και που πρέπει να αντιστοιχηθούν με τα αντίστοιχα χαμηλού επιπέδου οπτικά χαρακτηριστικά γνωρίσματα των ζητούμενων στοιχείων μέσων.
- Το στοιχείο AudioDescriptor (AD), όπου οι αναφορές διευκρινίζονται στα χαμηλού επιπέδου ακουστικά χαρακτηριστικά γνωρίσματα που κτίζονται σύμφωνα με τους περιγραφής του ήχου MPEG-7 [55] και που πρέπει να αντιστοιχηθούν με τα αντίστοιχα χαμηλού επιπέδου οπτικά χαρακτηριστικά γνωρίσματα των ζητούμενων στοιχείων μέσων.

- Το στοιχείο AudioDescriptionScheme (ADS), όπου οι αναφορές διευκρινίζονται στα χαμηλού επιπέδου ακουστικά χαρακτηριστικά γνωρίσματα που κτίζονται σύμφωνα με τα σχέδια περιγραφής του ήχου MPEG-7 και που πρέπει να αντιστοιχηθούν με τα αντίστοιχα χαμηλού επιπέδου οπτικά χαρακτηριστικά γνωρίσματα των ζητούμενων στοιχείων μέσων.
- Το στοιχείο VisualTimeSeriesDescriptor (VTSD), όπου οι αναφορές διευκρινίζονται στους περιγραφείς που αντιπροσωπεύουν τη χρονική διαταγή των οπτικών χαρακτηριστικών γνωρισμάτων της κίνησης των περιοχών που κτίζονται σύμφωνα με τα σχέδια περιγραφής των MPEG-7 Visual και που πρέπει να αντιστοιχηθούν με τα αντίστοιχα χαμηλού επιπέδου οπτικά χαρακτηριστικά γνωρίσματα των ζητούμενων στοιχείων μέσων.
- Το στοιχείο DescriptorRef (DR), όπου οι χρήστες διευκρινίζουν μια υπάρχουσα περιγραφή MPEG-7 που πρέπει να καθοδηγήσει τις ερωτήσεις query-by-example για τις ικανοποιημένες περιγραφές πολυμέσων.
- Το στοιχείο SemanticEntityRef (SER), όπου οι χρήστες διευκρινίζουν μια υπάρχουσα περιγραφή MPEG-7 που πρέπει να καθοδηγήσει τις ερωτήσεις query-by-example για τις επαναχρησιμοποιήσιμες σημασιολογικές οντότητες και τις οντολογίες περιοχών που εκφράζονται χρησιμοποιώντας τη σύνταξη MPEG-7.

Οι μεταβλητές παρέχονται στην MP7QL, προκειμένου να υποστηρίξει ενώσεις (joins) στους όρους για τα χαρακτηριστικά γνωρίσματα των MPEG-7 περιγραφών. Από συντακτική άποψη, μια μεταβλητή είναι ένα προσδιοριστικό (identifier) που αρχίζει με το χαρακτήρα '$'.

Το MP7QL input query format έχει εκφραστεί χρησιμοποιώντας και τη σύνταξη XML Schema και της OWL διαθέσιμο στο http://www.music.tuc.gr/delos/resources/MP7QL_XS.zip και στο http://www.music.tuc.gr/delos/resources/MP7QL_OWL.zip).

2.5 Το Πλαίσιο Διαχείρισης Μεταδεδομένων για Οπτικοακουστικό Υλικό DS-MIRF

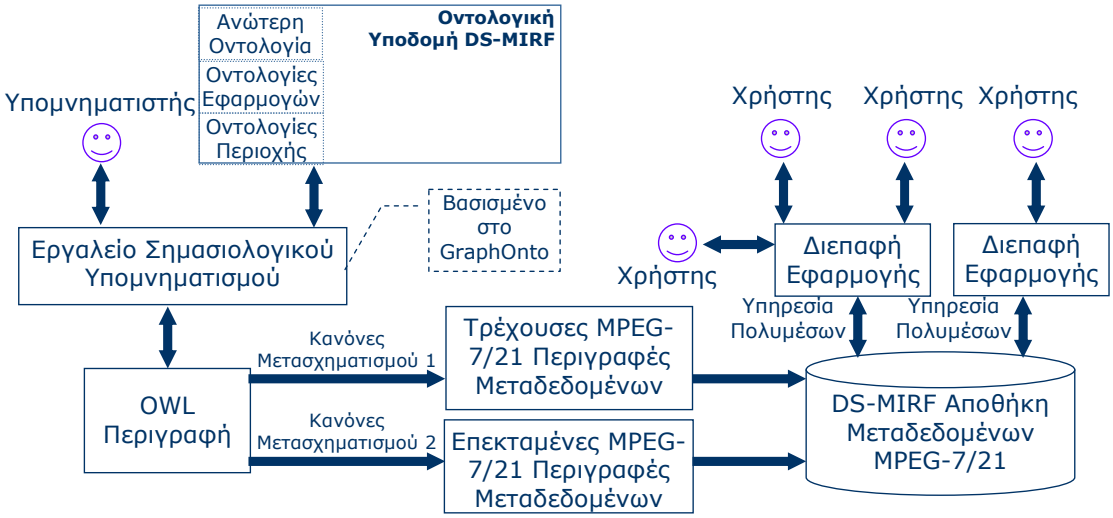
Στην ενότητα αυτή περιγράφεται το πλαίσιο διαχείρισης μεταδεδομένων για οπτικοακουστικό υλικό **DS-MIRF (Domain-Specific Multimedia Information and Filtering Framework).** Το πλαίσιο DS-MIRF παρέχει ένα σύνολο υπηρεσιών που υποστηρίζουν την ανάπτυξη εφαρμογών πολυμέσων, οι οποίες βασίζονται σε γνώση περιοχής και χρησιμοποιούν και επεκτείνουν τα πρότυπα MPEG-7 και MPEG-21.

Παρουσιάζεται η αρχιτεκτονική του πλαισίου DS-MIRF για την υποστήριξη σημασιολογικής διαλειτουργικότητας μεταξύ προτύπων που βασίζονται σε XML Schema και OWL. Το πλαίσιο DS-MIRF παρέχει ένα σύνολο υπηρεσιών που υποστηρίζουν την ανάπτυξη εφαρμογών πολυμέσων, οι οποίες βασίζονται σε γνώση περιοχής και χρησιμοποιούν και επεκτείνουν τα πρότυπα MPEG-7 και MPEG-21. Οι χρήστες του πλαισίου DS-MIRF διακρίνονται σε δυο κατηγορίες:

1. Τους **υπομνηματιστές (annotators)**, οι οποίοι είναι υπεύθυνοι για τη σημασιολογική περιγραφή του περιεχομένου του διαθέσιμου οπτικοακουστικού υλικού, χρησιμοποιώντας γνώση περιοχής (π.χ. είναι υπεύθυνοι να περιγράψουν σημασιολογικά ένα τμήμα video που περιέχει ένα γκολ της ΑΕΚ σε βάρος του Ολυμπιακού). Έτσι, οι υπομνηματιστές είναι στην ουσία υπεύθυνοι για τον ορισμό σημασιολογικών μεταδεδομένων (semantic metadata) που περιγράφουν οπτικοακουστικό υλικό.

2. Τους **τελικούς χρήστες (end users)**, οι οποίοι χρησιμοποιούν της εφαρμογές αναζήτησης, ανάκτησης και φιλτραρίσματος περιεχομένου πολυμέσων που υποστηρίζονται από της υπηρεσίες που παρέχει το πλαίσιο DS-MIRF..

Η αρχιτεκτονική του πλαισίου DS-MIRF και η ροή πληροφορίας μεταξύ των συστατικών του απεικονίζονται στην παρακάτω εικόνα.



Εικόνα 3: Η αρχιτεκτονική του πλαισίου DS MIRF

Όπως φαίνεται οι υπομνηματιστές χρησιμοποιούν κάποιο **εργαλείο σημασιολογικού υπομνηματισμού (semantic annotation tool)**, που βασίζεται στο συστατικό λογισμικού (software component) **GraphOnto** [14].Το GraphOnto επιτρέπει τη διαχείριση οντολογιών και μεταδεδομένων που ορίζονται με βάση οντολογίες και επιτρέπει την εκμετάλλευση και επέκταση της **οντολογικής υποδομής (ontological infrastructure)** του πλαισίου DS-MIRF.

Η οντολογική υποδομή του πλαισίου DS-MIRF απαρτίζεται από: **(α)** Μια **Ανώτερη Οντολογία (Upper Ontology)** που αποτυπώνει πλήρως τη σημασιολογία του MPEG-7 MDS και της MPEG-21 DIA Architecture; **(β)** Ένα σύνολο από **Οντολογίες Εφαρμογών (Application Ontologies)**, που επεκτείνουν τη σημασιολογία της Ανώτερης Οντολογίας με πληροφορία που σχετίζεται με εφαρμογές; Και **(γ)** Ένα σύνολο από **Οντολογίες Περιοχής (Domain Ontologies)**, που επεκτείνουν τη σημασιολογία της Ανώτερης Οντολογίας και των Οντολογιών Εφαρμογών με γνώση περιοχής.

Οι οντολογίες που απαρτίζουν την οντολογική υποδομή του πλαισίου DS-MIRF είναι εκφρασμένες με τη χρήση των δομών της OWL, που είναι η επικρατούσα γλώσσα ορισμού οντολογιών. Έτσι, το αποτέλεσμα του σημασιολογικού υπομνηματισμού είναι μια OWL περιγραφή του περιεχομένου του οπτικοακουστικού υλικού. Η χρήση των οντολογιών επιτρέπει τον αυστηρό έλεγχο της σημασιολογικής ορθότητας των παραγόμενων σημασιολογικών περιγραφών με τη χρήση γνώσης περιοχής (π.χ. δεν επιτρέπεται μια ομάδα να έχει ταυτόχρονα ενεργούς, κατά τη διάρκεια της αγώνα ποδοσφαίρου, περισσότερους από 11 παίκτες). Η

λειτουργικότητα αυτή παρέχεται από το GraphOnto, το οποίο υποστηρίζει, κατά τη διάρκεια του υπομνηματισμού, την επικύρωση της ορθότητας (validation) τόσο των οντολογιών όσο και των μεταδεδομένων που ορίζονται με βάση αυτές.

Στη συνέχεια, οι OWL/RDF περιγραφές μετατρέπονται, χρησιμοποιώντας το κατάλληλο σύνολο **κανόνων μετασχηματισμού (transformation rules)** που υλοποιούνται στο GraphOnto, σε MPEG-7/21 περιγραφές. Οι περιγραφές αυτές υπακούν, κατά περίπτωση, στην τρέχουσα μορφή των προτύπων ή σε προτεινόμενες επεκτάσεις της που απεικονίζονται σε κάποιες από της οντολογίες εφαρμογών, καλύπτουν της απαιτήσεις συγκεκριμένων εφαρμογών και βασίζονται σε άλλα πρότυπα ή σε ευρέως αποδεκτά μοντέλα.

Οι MPEG-7/21 περιγραφές αποθηκεύονται στη **DS-MIRF Αποθήκη Μεταδεδομένων MPEG-7/21 (DS-MIRF MPEG-7/21 Metadata Repository)**, η οποία παρέχει ένα σύνολο εξατομικευμένων, βασισμένων σε γνώση περιοχής υπηρεσιών για αναζήτηση, ανάκτηση, διανομή, παρουσίαση και φιλτράρισμα περιεχομένου πολυμέσων. Οι υπηρεσίες ανάκτησης βασίζονται στη γλώσσα MP7QL, που αναπτύχθηκε στα πλαίσια της παρούσας διατριβής και επιτρέπει ερωτήσεις σε όλες τις απόψεις των MPEG-7 περιγραφών, ενώ η εξατομίκευση των υπηρεσιών και οι υπηρεσίες φίλτρων βασίζονται στο μοντέλο προτιμήσεων χρηστών που αναπτύχθηκε στα πλαίσια της διατριβής. Οι υπηρεσίες που παρέχονται από τη DS-MIRF Αποθήκη Μεταδεδομένων MPEG-7/21 χρησιμοποιούνται στη συνέχεια από εφαρμογές που επικοινωνούν με της χρήστες μέσω **διεπαφών εφαρμογής (application interfaces)**, όπως για παράδειγμα η βασισμένη σε οντολογίες γεννήτρια διεπαφών φυσικής γλώσσας για αποθήκες οπτικοακουστικού υλικού OntoNL

Η προσέγγιση της παρούσας διατριβής για την υποστήριξη σημασιολογικής διαλειτουργικότητας μεταξύ OWL και MPEG-7/21 είναι χρήσιμη για δυο διαφορετικές κατηγορίες πιθανών χρηστών:

- Πιθανούς χρήστες που γνωρίζουν την ύπαρξη του πλαισίου DS-MIRF και του λογισμικού που αυτό παρέχει και θα ήθελαν να ενσωματώσουν οντολογίες που αναπτύσσουν στο πλαίσιο (ως οντολογίες περιοχής ή οντολογίες εφαρμογών), ώστε να υποστηριχθεί η διαλειτουργικότητά της με τα MPEG-7/21.

- Πιθανούς χρήστες που διαθέτουν οντολογίες ανεξάρτητες από το πλαίσιο DS-MIRF, οι οποίες θα μπορούσαν να ενσωματωθούν στο πλαίσιο DS-MIRF ως οντολογίες περιοχής ή οντολογίες εφαρμογών.

2.6 Το WordNet

Το WordNet [23]. είναι ένα σημασιολογικό λεξικό για την αγγλική γλώσσα. Ομαδοποιεί τις αγγλικές λέξεις σε σύνολα συνωνύμων αποκαλούμενων synsets, παρέχει σύντομους, γενικούς ορισμούς, και διάφορες σημασιολογικές σχέσεις μεταξύ αυτών των συνόλων συνωνύμων. Ο σκοπός είναι διπλός: να παράγει έναν συνδυασμό λεξικού και θησαυρού που είναι διαισθητικά πιο χρησιμοποιήσιμος, και για να υποστηρίξει τις εφαρμογές αυτόματης ανάλυσης κειμένων και τεχνητής νοημοσύνης.

Η βάση δεδομένων που χρησιμοποιεί η συγκεκριμένη εφαρμογή καθώς και τα εργαλεία λογισμικού που περιέχονται είναι ελεύθερα προς χρήση. Από το 2005, η βάση δεδομένων περιέχει περίπου 150.000 λέξεις που οργανώνονται σε πάνω από 115.000 synsets.

Το WordNet κάνει διάκριση μεταξύ των ουσιαστικών, των ρημάτων, των επιθέτων και των επιρρημάτων επειδή ακολουθούν τους διαφορετικούς γραμματικούς κανόνες. Κάθε synset (σύνολο εννοιών και συνωνύμων) περιέχει μια ομάδα συνώνυμων λέξεων ή παραθέσεων (μια *παράθεση* είναι μια ακολουθία λέξεων που πηγαίνουν μαζί να διαμορφώσουν μια συγκεκριμένη έννοια). Διαφορετικές έννοιες μίας λέξης είναι σε διαφορετικό synset. Ένα κλασικό παράδειγμα synset που ακολουθείται από επεξήγηση είναι το εξής:

Sense 1
player, participant -- (a person who participates in or is skilled at some game)
=> contestant -- (a person who participates in competitions)

Sense 2
musician, instrumentalist, player -- (someone who plays a musical instrument (as a profession))
=> performer, performing artist -- (an entertainer who performs a dramatic or musical work for an audience)

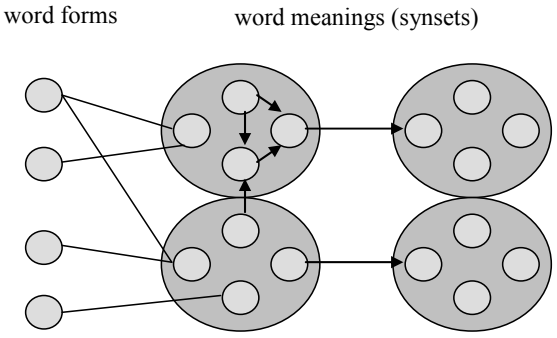
Sense 3
actor, histrion, player, thespian, role player -- (a theatrical performer)
=> performer, performing artist -- (an entertainer who performs a dramatic or musical work for an audience)

Sense 4
player -- (an important participant (as in a business deal); "he was a major player in setting up the corporation")
=> participant -- (someone who takes part in an activity)

Τα περισσότερα synsets συνδέονται με άλλα synsets μέσω διάφορων σημασιολογικών σχέσεων. Αυτές οι σχέσεις ποικίλλουν βασισμένοι στον τύπο λέξης και παρουσιάζονται στο παρακάτω πίνακα:

- Nouns
  - *hypernyms*: *Y* is a hypernym of *X* if every *X* is a (kind of) *Y*
  - *hyponyms*: *Y* is a hyponym of *X* if every *Y* is a (kind of) *X*
  - *coordinate terms*: *Y* is a coordinate term of *X* if *X* and *Y* share a hypernym
  - *holonym*: *Y* is a holonym of *X* if *X* is a part of *Y*
  - *meronym*: *Y* is a meronym of *X* if *Y* is a part of *X*
- Verbs
  - *hypernym*: the verb *Y* is a hypernym of the verb *X* if the activity *X* is a (kind of) *Y* (*travel* to *movement*)
  - *troponym*: the verb *Y* is a troponym of the verb *X* if the activity *Y* is doing *X* in some manner (*lisp* to *talk*)
  - *entailment*: the verb *Y* is entailed by *X* if by doing *X* you must be doing *Y* (*snoring* by *sleeping*)
  - *coordinate terms*: those verbs sharing a common hypernym
- Adjectives
  - *related nouns*
  - *participle of verb*
- Adverbs

Συνοψίζοντας, σε γενικές γραμμές στο WordNet, οι λέξεις οργανώνονται σε ταξινομίες όπου κάθε κόμβος είναι ένα σύνολο συνωνύμων (ένα synset) που αντιπροσωπεύουν μια ολοκληρωμένη έννοια. Η λογική δομή WordNet παρουσιάζεται στο επόμενο σχήμα:



Εικόνα 4: Η λογική δομή του Wordnet

Υπάρχουν 4 διαφορετικές ταξινομίες βασισμένες στα ευδιάκριτα μέρη του λόγου (ουσιαστικά, ρήματα, επίθετα, επιρρήματα) και πολλές σχέσεις που καθορίζονται μέσα σε κάθε μια. Για την εύρεση συνωνύμου και έννοιας χρησιμοποιούμε μόνο την ταξινομία ουσιαστικού με τις hyponym/hypernym σχέσεις, οι οποίες αφορούν τις συγκεκριμένες έννοιες

2.7 Σχετικές εργασίες

Οντολογίες και διεπαφές φυσικής γλώσσας

Οι οντολογίες φαίνεται να αποδεικνύονται η σωστή απάντηση για τη δόμηση και μοντελοποίηση της γνώσης. Η βασισμένη σε οντολογία σημασιολογικής πληροφορίας σήμανση σε μια βάση γνώσης, ανοίγει το δρόμο στις νέες, περίπλοκες μορφές ερωταπαντήσεων, έναντι των οποίων όχι μόνο μπορεί ενδεχομένως να παρέχει αυξανόμενο precision και recall σε σχέση με τις σημερινές μηχανές αναζήτησης, αλλά είναι επίσης σε θέση να παρέχει πρόσθετες λειτουργίες, όπως i) προσφέροντας επιπλέον πληροφορίες για μια απάντηση, ii) παρέχοντας μέτρα αξιοπιστίας και σταθερότητας iii) εξηγώντας πώς προέκυψε η απάντηση.

Ένα σύστημα αλληλεπίδρασης φυσικής γλώσσας που συγχωνεύει την επεξεργασία φυσικής γλώσσας (Natural Language Processing, NLP), τη λογική, τις οντολογίες και τις τεχνικές ανάκτησης πληροφοριών για να δοθούν οι απαντήσεις στις ερωτήσεις σε μια συγκεκριμένη περιοχή σε πραγματικό χρόνο είναι το AQUA [44]. Το AQUA μεταφράζει τις αγγλικές ερωτήσεις σε λογικές ερωτήσεις που χρησιμοποιούνται έπειτα για να παραγάγουν αποτελέσματα. Το AQUA συνδέεται με την οντολογία αναφοράς AKT για την ακαδημαϊκή περιοχή. Αυτή η οντολογία (που γράφεται σε OCML) αυτήν την περίοδο περιέχει τους ανθρώπους, τις οργανώσεις, τους ερευνητικούς τομείς, τα προγράμματα, τις δημοσιεύσεις, τις τεχνολογίες και τις εκδηλώσεις, και τις εργασίες ως pattern -matching, το οποίο σημαίνει ότι προσπαθεί να βρει την ακριβή αντιστοιχία με τα ονόματα στην οντολογία. Το μειονέκτημα είναι ότι οι δοκιμές που είναι πραγματοποιημένες αφορούν μόνο μια συγκεκριμένη οντολογία με μια συγκεκριμένη γραμματική και το σύστημα δεν λαμβάνει υπόψη τη σημασιολογία της οντολογίας. Οι δοκιμές αξιολόγησης ήταν αρκετά προκαταρκτικές και περιορισμένες.

Μια εργασία για ερωταπαντήσεις πάνω στη British Telecom Digital Library περιγράφεται από τον Cimiano και λοιπούς [45]. Είναι μια προσέγγιση στις

ερωταπαντήσεις πέρα από τους πόρους γνώσης ,που χρησιμοποιεί τα διαφορετικά τμήματα οντολογίας μέσα σε ένα σενάριο εφαρμογής της ψηφιακής βιβλιοθήκης της ΒΤ. Η καινοτομία της προσέγγισης βρίσκεται στον συνδυασμό διαφορετικών σημασιολογικών τεχνολογιών που παρέχουν ένα σαφές όφελος για το σενάριο εφαρμογής που εξετάζεται. Το μειονέκτημα αφορά τη διεπαφή φυσικής γλώσσας όπου η μετάφραση των ερωτήσεων φυσικής γλώσσας σε δομημένες ερωτήσεις στηρίζεται σε ένα περιορισμένο και μερικώς αυτόματα παραγόμενο λεξικό για την βασική οντολογία. Το λεξικό διευκρινίζει τις πιθανές λεξικολογικές αντιπροσωπεύσεις των στοιχείων οντολογίας στην ερώτηση του χρήστη. Έτσι, η διαδικασία αποσαφήνισης περιορίζεται στη συγκεκριμένη εφαρμογή και δεν είναι αυτόματη.

2.8 Ανακεφαλαίωση

Συνοψίζοντας, στο συγκεκριμένο κεφάλαιο κάναμε μία αναφορά σε γνώσεις σχετικές με το αντικείμενο μελέτης της παρούσας εργασίας. Επίσης, παρουσιάσαμε εργασίες με παρόμοιο αντικείμενο μελέτης, τη συμβολή τους στη περιοχή καθώς και τα προβλήματα που αντιμετώπιζαν. Στο επόμενο κεφάλαιο θα αναφερθούμε στις διάφορες τεχνολογίες που μελετήθηκαν και χρησιμοποιήθηκαν κατά τη παρούσα εργασία.

ΚΕΦΑΛΑΙΟ 3

ΤΕΧΝΟΛΟΓΙΚΗ ΒΑΣΗ

Εισαγωγή

Σε αυτό το κεφάλαιο θα επισημάνουμε τις τεχνολογίες που χρησιμοποιήσαμε στη παρούσα διπλωματική εργασία. Θα αναφέρουμε τα βασικότερα χαρακτηριστικά τους καθώς και τις διάφορες ιδιαιτερότητες που έχουν.

3.1 Ο Stanford Log-linear Part-Of-Speech Tagger

Σύμφωνα με τη διαδικασία του Pos-Tagging, αναθέτουμε ετικέτες που αναφέρουν τα μέρη του λόγου σε λέξεις έτσι ώστε αυτές να απεικονίζουν τη συντακτική κατηγορία τους. Συχνά οι λέξεις μπορούν να ανήκουν σε διαφορετικές συντακτικές κατηγορίες ανάλογα με τα συμφραζόμενα στα οποία αυτές ανήκουν. Ουσιαστικά το Pos-Tagging είναι μια πρώτη προσπάθεια να αποσαφηνιστούν οι έννοιες των λέξεων τις οποίες ο χρήστης έχει χρησιμοποιήσει στο αίτημα του.

Εφαρμόζοντας διάφορες μεθόδους, ο Stanford Tagger (http: // nlp.stanford.edu/ software/tagger.shtml) δίνει ακρίβεια 96.86% στο Penn Treebank (http://www.cis.upenn.edu/~treebank/), μία μείωση λάθους 4.4% στο καλύτερο προηγούμενο αποτέλεσμα και 86.91% σε λέξεις που δεν έχουν ξαναεμφανιστεί.

Σε γενικές γραμμές, ο Part-Of-Speech Tagger (POS Tagger) είναι μία εφαρμογή η οποία «διαβάζει» μία πρόταση και αναθέτει τα μέρη του λόγου όπως ουσιαστικό, ρήμα, επίθετο κ.τ.λ στις λέξεις που την αποτελούνε. Επίσης αναπαριστά σε δενδροειδή μορφή τις αλληλουχίες των λέξεων μέσα στη πρόταση.

3.2 Το JWordnet

Το JWordnet [24] είναι μια καθαρά αυτόνομη αντικειμενοστραφής διεπαφή της γλώσσας προγραμματισμού JAVA [16] στη βάση δεδομένων WordNet που περιέχει το θησαυρό των λεξικολογικών σχέσεων. Προορίζεται για τους προγραμματιστές οι οποίοι επιθυμούν να γράψουν JAVA εφαρμογές οι οποίες κάνουν χρήστη τοπικών αντιγράφων των αρχείων του Wordnet και εκμεταλλεύονται συγχρόνως όλη τη λειτουργικότητα που τους παρέχεται. Ένα απλό παράδειγμα χρήσης του είναι το εξής:

```
    import edu.brandeis.cs.steele.wn.*;

    /* This prints the senses of the noun 'dog' to the console. */
    public class Main {
      static void main(String[] args) {
      DictionaryDatabase dictionary = new FileBackedDictionary();
      IndexWord word = dictionary.lookupIndexWord(POS.NOUN, "dog");
        Synset[] senses = word.getSenses();
        int taggedCount = word.getTaggedSenseCount();
        System.out.print("The " + word.getPOS().getLabel() + " " +
    word.getLemma() + " has " + senses.length + " sense" +
    (senses.length == 1 ? "" : "s") + " ");
        System.out.print("(");
        if (taggedCount == 0) {
          System.out.print("no senses from tagged texts");
        } else {
          System.out.print("first " + taggedCount + " from tagged
    texts");
        }
        System.out.print(")\n\n");
        for (int i = 0; i < senses.length; ++i) {
          Synset sense = senses[i];
          System.out.println("" + (i + 1) + ". " +
    sense.getLongDescription());
        }
      }
}
```

3.3 Η γλώσσα διατύπωσης ερωτήσεων SPARQL Query Language for RDF.

3.3.1 Γενικά

Το RDF (Resource Description Framework) [19] είναι ένα γενικό πλαίσιο εργασίας το οποίο μπορεί να περιγράψει οποιοδήποτε πόρο του Internet, όπως ένα Web site και τα περιεχόμενα του. Ένας RDF γράφος (RDF Graph) είναι ένα σύνολο από τριάδες (triples) που η κάθε μία αποτελείται από ένα subject, ένα predicate και ένα object. Η SPARQL [21] λοιπόν, είναι μία γλώσσα ερωτήσεων (query language) η οποία απευθύνεται σε τέτοιους γράφους με σκοπό να ανακτήσει πληροφορία. Παρέχει διευκολύνσεις όπως:

- να εξάγει πληροφορίες υπό τη μορφή URIs, blank nodes και literals.
- να εξάγει RDF subgraphs
- να δημιουργεί νέους RDF graphs βασισμένους σε πληροφορία βασισμένη στα queried graphs.

Η γλώσσα SPARQL δεν επιλέχθηκε αυθαίρετα. Κάποιος μπορεί να ρωτήσει γιατί όχι SQL ή XML. Η απάντηση σε αυτό το ερώτημα περιγράφεται αναλυτικά στη δημοσίευση *"MELTON, J. 2006 SQL, XQuery, and SPARQL: What's Wrong With*

This Picture? XTech 2006: "Building Web 2.0" — 16-19 May 2006, Amsterdam, The Netherlands" [43] και η οποία αναφέρει τα εξής:

Οι γλώσσες διατύπωσης ερωτήσεων έχουν σχεδιαστεί με σκοπό να εφαρμοστούν στα στοιχεία που αντιστοιχούν σε ένα ιδιαίτερο πρότυπο στοιχείων. Παραδείγματος χάριν, η SQL χρησιμοποιείται για να ανακτήσει, να δημιουργήσει, να τροποποιήσει, και να διαγράψει τα στοιχεία που αντιπροσωπεύονται στο σχεσιακό μοντέλο δεδομένων. Όμοια, η XQuery [46] χρησιμοποιείται για να εντοπίσει και να ανακτήσει τα στοιχεία που αντιπροσωπεύονται από το μοντέλο δεδομένων XPath [47], το XDM [48]. Είναι μερικές φορές δυνατό να χρησιμοποιηθεί μια γλώσσα σε στοιχεία που αντιπροσωπεύονται σε ένα πρότυπο δεδομένων διαφορετικό από αυτό για το οποίο η γλώσσα σχεδιάστηκε. Αυτό μπορεί να επιτευχθεί με την αντιστοίχηση (mapping) των στοιχείων από το εγγενές πρότυπο στοιχείων του στο πρότυπο γλωσσικών στοιχείων διατύπωσης ερωτήσεων, αλλά η ερώτηση έπειτα θα είναι με κάποιο κόστος.

Κάθε ένα από αυτά τα πρότυπα δεδομένων και αντίστοιχες γλώσσες διατύπωσης ερωτήσεων έχουν πλεονεκτήματα μειονεκτήματα. Η SQL και το σχεσιακό μοντέλο έχουν ως σκοπό να αντιπροσωπεύσουν τα ιδιαίτερα δομημένα στοιχεία όπως αυτά που χρησιμοποιούνται σε πολλές επιχειρησιακές διαδικασίες. Τέτοια στοιχείο περιλαμβάνουν συνήθως μια τιμή για κάθε στήλη κάθε πίνακα. Η SQL, αλλά όχι το σχεσιακό μοντέλο, υποστηρίζει μια ειδική τιμή αποκαλούμενη μηδενική αξία (null value) για να αντιπροσωπεύσει το στοιχείο που λείπει. Η δυνατότητα των μηδενικών τιμών περιπλέκει τον καθορισμό και τις ερωτήσεις που γράφονται στη γλώσσα SQL, η οποία καθιστά την SQL όχι και τόσο λειτουργική για τη χρήση όσον αφορά στις λιγότερο δομημένες πληροφορίες. Η σύνταξη της γλώσσας SQL εστιάζει στον προσδιορισμό των στοιχείων για τα οποία τα περισσότερα ή όλα τα συστατικά είναι διαθέσιμα και συνδυάζοντας τα δεδομένα βασισμένα στις τιμές εκείνων των συστατικών.

Στην ερώτηση γιατί το RDF δεν είναι, ή δεν πρέπει να είναι, αποθηκευμένο σε μια σχεσιακή βάση δεδομένων και έπειτα να γίνονται ερωτήσεις χρησιμοποιώντας SQL, μια καλή απάντηση είναι αυτή: Η σύνταξη SPARQL κάνει όλες τις ενώσεις λειτουργιών (join operation) να εννοούνται, ενώ η σύνταξη SQL τις καθιστά συνήθως ρητές. Μια συνέπεια αυτού του σχεδιασμού είναι ότι οι εκφράσεις SQL για να απαντήσουν στις χαρακτηριστικές ερωτήσεις που θα υποβληθούν ενάντια στις συλλογές RDF τείνουν να είναι πολύ μεγαλύτερες και κάπως δυσκολότερες να

δημιουργηθούν λόγω της ανάγκης να γραφτούν ρητώς τα join operations. Επειδή οι χαρακτηριστικές ερωτήσεις που υποβάλλονται σε RDF περιλαμβάνουν συνήθως πολλά join operations, η SPARQL παρέχει μια συμπαγέστερη τεχνική που είναι ευκολότερη να γίνει σωστά και με λιγότερο χρόνο εκσφαλμάτωσης (debugging time).

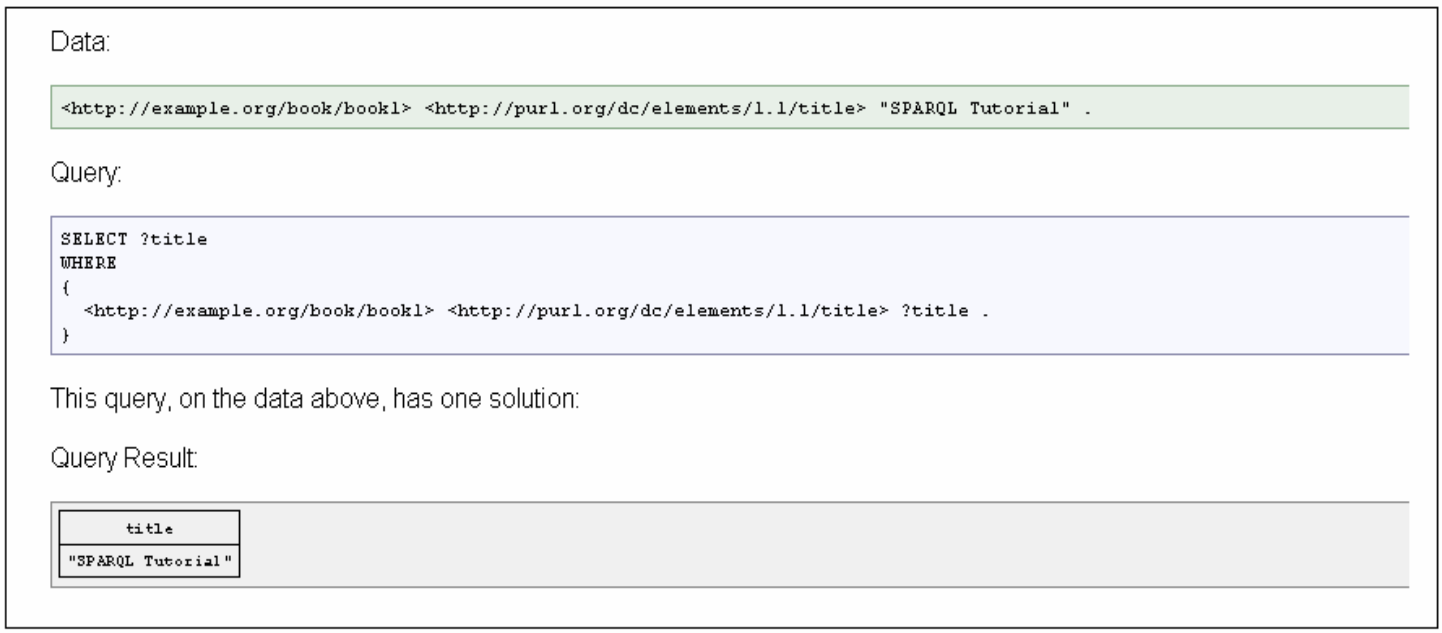
Στην ερώτηση γιατί η XQuery δεν είναι πιό κατάλληλη, δεδομένου ότι τα RDF έγγραφα αποθηκεύονται σε μορφή XML, η απάντηση είναι παρόμοια: ο αριθμός των join operations στα for clauses και ο αριθμός των join conditions που απαιτούνται στα where clauses καθιστούν πιθανώς τις εκφράσεις XQuery πιο δύσκολες να γραφθούν και να εκσφαλαματωθούν από τις αντίστοιχες ερωτήσεις σε SPARQL

Η SPARQL, που είναι και μια γλώσσα διατύπωσης ερωτήσεων και ένα πρωτόκολλο πρόσβασης δεδομένων (data access protocol), έχει τη δυνατότητα να γίνει ένα βασικό συστατικό στις σημασιολογικές εφαρμογές Ιστού: σαν ένα πρότυπο που υποστηρίζεται από ένα εύκαμπτο πρότυπο δεδομένων (data model), μπορεί να παρέχει έναν κοινό μηχανισμό ερώτησης για όλες τις σημασιολογικές εφαρμογές Ιστού. Επιλέγεται SPARQL ως γλώσσα διατύπωσης ερωτήσεων για να αντιπροσωπευθούν οι ερωτήσεις φυσικής γλώσσας μετά από τη συντακτική και σημασιολογική αποσαφήνιση δεδομένου ότι η SPARQL καθορίζεται από την άποψη του προτύπου δεδομένων W3C'S RDF και θα δουλέψει για οποιαδήποτε πηγή στοιχείων που μπορεί να αντιστοιχηθεί - χαρτογραφηθεί σε RDF

3.3.2 Γράφοντας ένα απλό query

Η γλώσσα διατύπωσης ερωτήσεων SPARQL είναι βασισμένη στο ταίριασμα πρότυπων γράφων (Graph patterns [26]). Τα Graph patterns περιέχουν πρότυπα τριάδων (triple patterns), και τα triple patterns με τη σειρά τους είναι σαν RDF triples με την πιθανότητα όμως να εμφανίζονται μεταβλητές στη θέση των RDF όρων στα subject, predicate και object.

Το παράδειγμα που ακολουθεί παρουσιάζει ένα SPARQL query το οποίο ζητά το τίτλο ενός βιβλίου από ένα δεδομένο data graph. Το query αποτελείται από δύο μέρη: Το SELECT clause, στο οποίο ορίζεται ποια(ες) μεταβλητή(ες) θα εμφανίζεται(ονται) στο query result, και το WHERE clause, στο οποίο παρουσιάζεται το graph pattern που επιθυμούμε να αντιστοιχηθεί στο data graph. Το graph pattern στο συγκεκριμένο παράδειγμα αποτελείται από μόνο ένα triple pattern με μόνο μία μεταβλητή (?title) στη θέση του object.


Data:

```
<http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> "SPARQL Tutorial" .
```

Query:

```
SELECT ?title
WHERE
{
  <http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> ?title .
}
```

This query, on the data above, has one solution:

Query Result:

title
"SPARQL Tutorial"

Εικόνα 5: Ένα απλό Sparql Query

Επομένως, στο παραπάνω παράδειγμα, στο WHERE κομμάτι του query στη θέση του subject υπάρχει το http://example.org/book/book1, στη θέση του predicate υπάρχει το http://purl.org/dc/elements/1.1/title και στη θέση του object υπάρχει η μεταβλητή title. Έτσι λοιπόν, στο query result θα είναι όλες οι τριάδες που αντιστοιχίζουν τα subject, predicate με τα αντίστοιχα που υπάρχουν στα data (συγκεκριμένα εδώ υπάρχει ένα αποτέλεσμα) και συγκεκριμένα θα εμφανίζεται ο τίτλος αφού αυτός ζητείται στο SELECT κομμάτι του query.

Επίσης, πολύ σημαντικό ρόλο στα sparql queries (τουλάχιστον σε αυτά που πραγματεύεται η παρούσα διπλωματική εργασία) έχει το keyword “UNION”. Το UNION ή αλλιώς το λογικό OR χρησιμοποιείται ανάμεσα σε triple patterns έτσι ώστε να υποδηλώσει ότι πρέπει να ικανοποιείται το λογικό Ή (OR) ανάμεσα σε αυτά τα patterns. Ακολουθεί ένα σύντομο παράδειγμα το οποίο καταδεικνύει τη λειτουργικότητα του UNION.

Data:

```
@prefix dc10:  <http://purl.org/dc/elements/1.0/> .
@prefix dc11:  <http://purl.org/dc/elements/1.1/> .

_:a  dc10:title     "SPARQL Query Language Tutorial" .
_:a  dc10:creator   "Alice" .

_:b  dc11:title     "SPARQL Protocol Tutorial" .
_:b  dc11:creator   "Bob" .

_:c  dc10:title     "SPARQL" .
_:c  dc11:title     "SPARQL (updated)" .
```

Query:

```
PREFIX dc10:  <http://purl.org/dc/elements/1.0/>
PREFIX dc11:  <http://purl.org/dc/elements/1.1/>

SELECT ?title
WHERE  { { ?book dc10:title  ?title } UNION { ?book dc11:title  ?title } }
```

Query result:

title
"SPARQL Protocol Tutorial"
"SPARQL"
"SPARQL (updated)"
"SPARQL Query Language Tutorial"

Εικόνα 6: Ένα Sparql query με UNION

Στο παραπάνω παράδειγμα παρουσιάζεται η λειτουργικότητα του UNION. Πιο συγκεκριμένα στο WHERE κομμάτι του query υπάρχουν 2 τριάδες (triples) που ενώνονται με το keyword UNION, η τριάδα *?book dc10:title ?title* και η τριάδα *?book dc11:title ?title*. Αφού ανάμεσα τους υπάρχει το UNION στο query result θα εμφανιστούν αποτελέσματα που ικανοποιούν είτε τη μία τριάδα είτε την άλλη (λογικό OR). Έτσι λοιπόν, αφού μόνο στο predicate υπάρχει όρος που πρέπει να αντιστοιχηθεί με τα data που υπάρχουν (*dc10:title, dc11:title*) στο query result θα εμφανιστούν οι τίτλοι όλων των τριάδων που ικανοποιούν αυτά τα predicate ("SPARQL Protocol Tutorial", "SPARQL", "SPARQL (updated)", "SPARQL Query Language Tutorial).

Τέλος αξίζει να σημειωθεί ότι στην υλοποίηση του **OntoNL Framework** χρησιμοποιούμε το Jena ARQ API [27] για να χτίσουμε αυτόματα SPARQL queries που υπακούουν στις αρχές δόμησης. Το Jena ARQ API δηλαδή, είναι ένα API που μας παρέχει έτοιμες JAVA συναρτήσεις έτσι ώστε να κάνουμε parse ένα query και να

το αναλύσουμε, να το μετατρέψουμε αλλά και να δημιουργήσουμε καινούρια queries από την αρχή.

3.4 GraphOnto – GraphOnto API

Το GraphOnto [14] είναι ένα γραφικό σύστημα που απλοποιεί τον ορισμό και την επεξεργασία τόσο οντολογιών που βασίζονται σε πρότυπα όσο και οντολογιών περιοχής. Η γλώσσα οντολογιών που υποστηρίζεται είναι η OWL η οποία αποτελεί στις μέρες μας την κυρίαρχη γλώσσα ορισμού οντολογιών. Η λειτουργικότητα που παρέχει το GraphOnto μπορεί να δειχθεί μέσω δύο διαφορετικών πρισμάτων. Αφενός αποτελεί έναν πλήρη και γενικό γραφικό συντάκτη OWL-DL οντολογιών ο οποίος καλύπτει την πλήρη σημασιολογία της OWL. Αφετέρου, χρησιμοποιείται μέσα στο DS-MIRF πλαίσιο ως ένα συστατικό που επιτρέπει το σημασιολογικό υπομνηματισμό περιεχομένου πολυμέσων κάνοντας χρήση και επεκτείνοντας την οντολογική υποδομή του. Ειδικότερα, υποστηρίζει το σημασιολογικό υπομνηματισμό (annotation) οπτικοακουστικών αντικειμένων με τη βοήθεια της ανώτερης οντολογίας (upper ontology) του DS-MIRF που περιγράφει πλήρως το MPEG-7 MDS. Επιπρόσθετα, επιτρέπει τη χρήση και τον ορισμό οντολογιών περιοχής (domain ontologies) οι οποίες ενσωματώνονται στο πλαίσιο και επεκτείνουν την ανώτερη οντολογία με γνώση περιοχής. Αυτές οι οντολογίες περιοχής μαζί με τις σημασιολογικές περιγραφές πολυμέσων οι οποίες ορίζονται με βάση αυτές τις οντολογίες, μπορούν να μετασχηματιστούν μέσω των κανόνων μετασχηματισμού του DS-MIRF σε MPEG-7 σημασιολογικές περιγραφές περιεχομένου πολυμέσων. Αυτές οι MPEG-7 περιγραφές δύνανται να αποθηκευθούν είτε σε έγκυρα MPEG-7 αρχεία είτε στην MPEG-7 αποθήκη μεταδεδομένων του DS-MIRF.

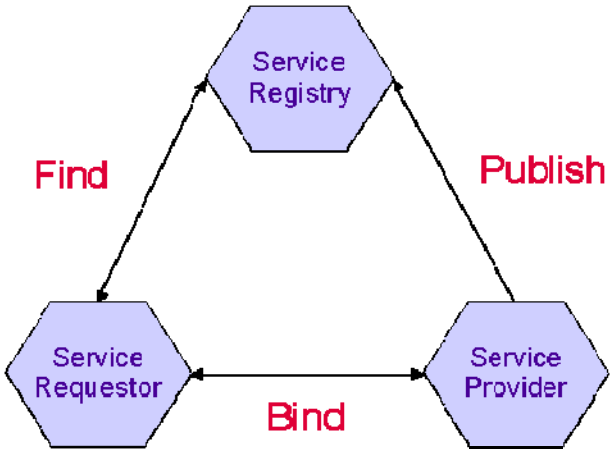
Παράλληλα, το GraphOnto σύστημα περιλαμβάνει το GraphOnto API που είναι ένα JAVA API που παρέχει προγραμματιστική πρόσβαση για πλήρη διαχείριση (πρόσθεση, επεξεργασία, διαγραφή) όλων των δομών μιας OWL οντολογίας. Το API αυτό χρησιμοποιείται μέσα στο DS-MIRF πλαίσιο για την ανάπτυξη εφαρμογών ανάκτησης και φιλτραρίσματος οι οποίες βασίζονται σε οντολογίες.

3.5 Web Services

3.5.1 Γενικά

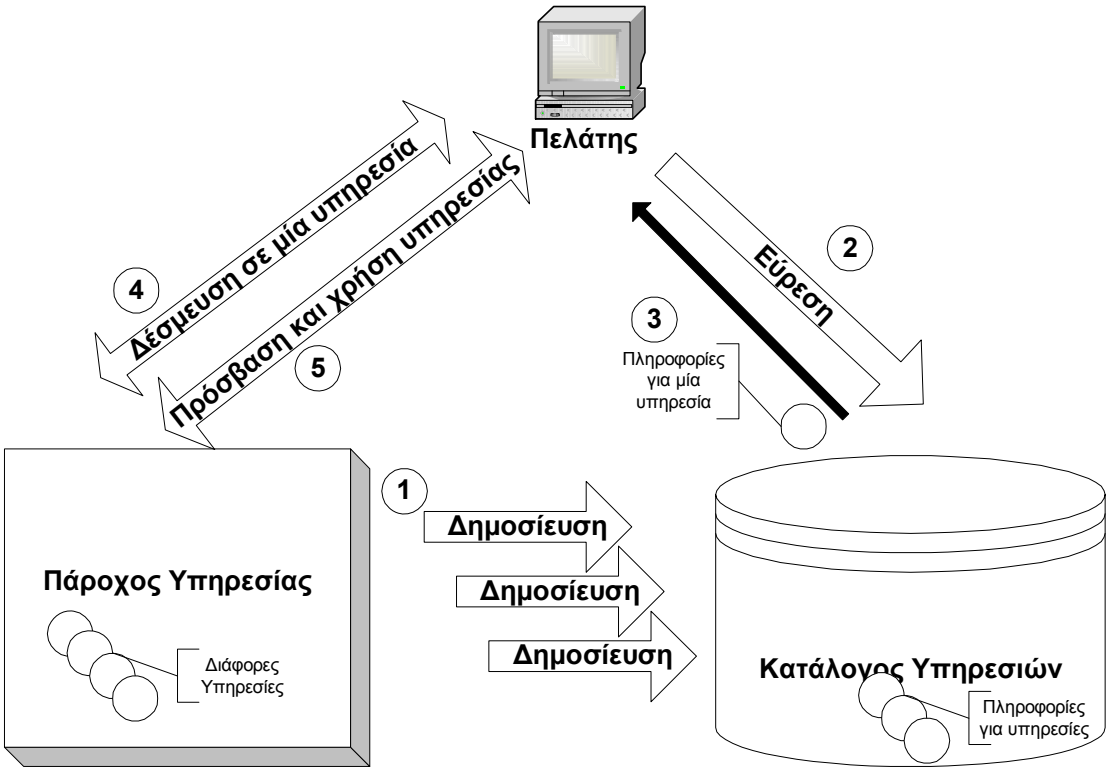
Τα web services είναι μια τεχνολογία που επιτρέπει στις εφαρμογές να επικοινωνούν μεταξύ τους ανεξαρτήτως πλατφόρμας και γλώσσας προγραμματισμού. Ένα web service είναι μια διεπαφή λογισμικού (software interface) που περιγράφει μια συλλογή από λειτουργίες, οι οποίες μπορούν να προσεγγιστούν από το δίκτυο μέσω πρότυπων μηνυμάτων XML. Χρησιμοποιεί πρότυπα βασισμένα στη γλώσσα XML, για να περιγράψει μία λειτουργία (operation) προς εκτέλεση και τα δεδομένα προς ανταλλαγή με κάποια άλλη εφαρμογή.

Η αρχιτεκτονική ενός web service μπορεί να περιγραφεί με το παρακάτω σχήμα [28]:



Εικόνα 7: Η αρχιτεκτονική ενός Web Service

Υπάρχουν οι εξής τρεις ρόλοι, service provider (φορέας παροχής υπηρεσιών), service requestor (ο αιτών των υπηρεσιών) και το service registry (κατάλογος υπηρεσιών). Επίσης υπάρχουν και τρεις λειτουργίες που αντιπροσωπεύουν τις αλληλεπιδράσεις μεταξύ τους, bind (δεσμεύω), publish (δημοσιεύω) και find (βρίσκω). Έτσι λοιπόν ο service provider δημοσιεύει μια περιγραφή υπηρεσιών (service description) στη Service Registry. Έπειτα ένας service requestor βρίσκει τη περιγραφή υπηρεσιών μέσω του service registry, και αν η περιγραφή υπηρεσιών είναι ικανοποιητική , δεσμεύει τον service provider για να χρησιμοποιήσει την υπηρεσία.



Εικόνα 8: Η αρχιτεκτονική ενός Web Service και οι ενέργειες που πραγματοποιούνται

Έτσι λοιπόν τα βήματα που πραγματοποιούνται από πλευρά του service provider είναι τα εξής:

- καθορισμός των υπηρεσιών που θα παρέχονται
- υλοποίηση της λειτουργίας πίσω από την υπηρεσία
- εγκατάσταση της εφαρμογής – παρόχου των υπηρεσιών
- δημοσίευση των web services σε μια υπηρεσία καταλόγου
- αναμονή για επεξεργασία αιτήσεων πελατών

Αντίστοιχα τα βήματα που πραγματοποιούνται από πλευρά service requestor είναι τα εξής:

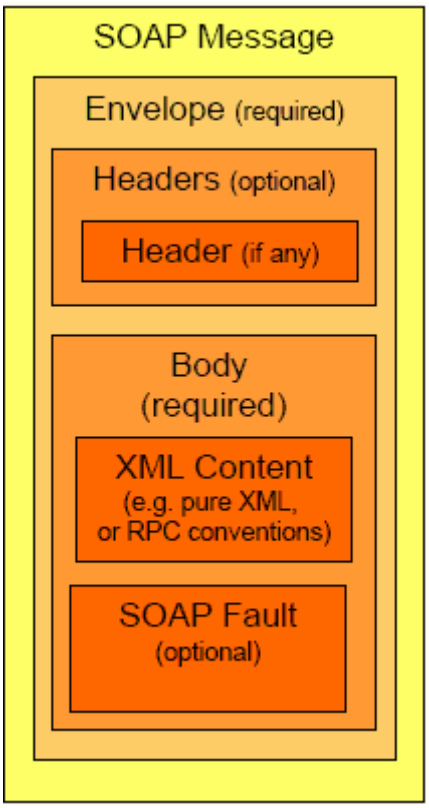
- προσδιορισμός των υπηρεσιών που θα απαιτηθούν
- εντοπισμός του web service ρωτώντας μία υπηρεσία καταλόγου
- αποστολή της αίτησης στην υπηρεσία
- λήψη της απάντησης από την υπηρεσία

Βασικές τεχνολογίες που χρησιμοποιούνται στα web services είναι το SOAP και το WSDL.

SOAP:

Το SOAP(Simple Object Access Protocol) είναι ένα απλό, ευέλικτο και επεκτάσιμο πρωτόκολλο ανταλλαγής μηνυμάτων σε μορφή XML . Είναι ανεξάρτητο από πλατφόρμα και γλώσσα προγραμματισμού οπότε μπορεί να χρησιμοποιηθεί για επικοινωνία μεταξύ εφαρμογών γραμμένων για διαφορετικές πλατφόρμες και σε διαφορετικές γλώσσες προγραμματισμού.

Η δομή ενός SOAP Message καθώς και η λειτουργικότητα κάθε υποενότητας που το αποτελεί φαίνεται στο παρακάτω σχήμα:



Εικόνα 9: Η δομή ενός SOAP Message

Σύμφωνα λοιπόν με το παραπάνω σχήμα μπορούμε να διακρίνουμε τα εξής:

- SOAP message: Είναι ένα XML έγγραφο
- Envelope: Το πραγματικό μήνυμα που θα επεξεργαστεί
- Headers: Προαιρετικό στοιχείο που καθορίζει συγκεκριμένες απαιτήσεις της εφαρμογής
- Body: Το περιεχόμενο του μηνύματος

- Το body μπορεί να έχει μόνο XML περιεχόμενο ή PRC πληροφορία (π.χ όνομα μεθόδων , παραμέτρων κτλ) κωδικοποιημένη σε XML
- Το body ενός μηνύματος απάντησης (response) μπορεί προαιρετικά να περιέχει SOAP Fault στοιχεία (συγκεκριμένους κωδικούς λάθους μαζί με περιγραφή)

WSDL:

Το WSDL (Web Services Description Language) είναι ένα σχήμα XML για τη περιγραφή δικτυακών υπηρεσιών. Βοηθάει δηλαδή τους παροχείς υπηρεσιών να περιγράψουν τη βασική μορφή των αιτήσεων και απαντήσεων των υπηρεσιών πάνω από διαφορετικά πρωτόκολλα και κωδικοποιήσεις. Επομένως λοιπόν η WSDL χρησιμοποιείται για να περιγράψει τι μπορεί να κάνει ένα web service, που βρίσκεται και πως να το καλέσει κανείς. Η δομή του WSDL καθώς και η λειτουργικότητα κάθε υποενότητας που το αποτελεί φαίνεται στο παρακάτω σχήμα:

Εικόνα 10: Η δομή του WSDL

Σύμφωνα λοιπόν με το παραπάνω σχήμα διακρίνονται τα εξής:

- Definitions: Το root element όλων των WSDL εγγράφων. Ορίζει το όνομα του web service, περιέχει πολλαπλά namespaces και περιέχει τα στοιχεία του service που θα περιγραφούν

- Types: Περιγράφει όλους τους τύπους δεδομένων μεταξύ πελάτη και εξυπηρετή.
- Message: Περιέχει τα μηνύματα που ένα service λαμβάνει και στέλνει.
- PortType: Είναι ένας συνδυασμός μηνυμάτων και λειτουργιών
- Binding: Περιγράφει το πρωτόκολλο επικοινωνίας που θα χρησιμοποιηθεί.
- Service: Καθορίζει το URL στο οποίο μπορεί να βρεθεί το εκάστοτε service.

3.5.2 Axis Web Service

Το Axis είναι ουσιαστικά μια SOAP engine , ένα πλαίσιο για να κατασκευάζουμε SOAP processors όπως clients, servers, gateways κτλ. Αλλά αυτή δεν είναι η μοναδική λειτουργικότητα του Axis. Επίσης περιέχει:

- έναν απλό stand – alone server
- έναν server ο οποίος μπορεί να χρησιμοποιηθεί ως plug in σε servlet engines όπως ο Tomcat.
- εκτεταμένη υποστήριξη για τη Web Services Description Language (WSDL)
- εργαλεία που παράγουν java classes από WSDL

Για να υλοποιήσουμε web services στο Axis επίσης χρειαζόμαστε ένα WSDD (Web Service Deployment Descriptor) . Ένα WSDD περιέχει τη πληροφορία εκείνη που επιθυμούμε να κάνουμε “deploy” στο Axis, δηλαδή που θέλουμε να έχουμε διαθέσιμη στο Axis engine.

Έτσι λοιπόν, σε γενικές γραμμές χρησιμοποιώντας αυτή τη τεχνολογία δημιουργήσαμε τα web services που προέκυψαν από την εφαρμογή **OntoNL.**

3.6 Η τεχνολογία XML Beans

Το XML Beans [51] είναι μια τεχνολογία για τη πρόσβαση σε XML έγγραφα και δεδομένα με την αναπαράσταση τους σε JAVA τύπους. Το XML Beans παρέχει διάφορους τρόπους ώστε να εισχωρήσει στην XML, συμπεριλαμβανομένων:

- Μέσω του XML Schema που έχει συνταχθεί για να παράγει Java τύπους που αντιπροσωπεύουν τους τύπους του σχήματος. Με αυτό το τρόπο είναι εφικτή η πρόσβαση σε Java δομές που αντιπροσωπεύουν XML με συναρτήσεις τύπου setFoo, getFoo.

- Ένα μοντέλο δρομέων (cursor model) μέσω του οποίου μπορεί να γίνει πλήρως η διάσχιση της XML πληροφορίας
- Υποστήριξη για τα DOM XML

Στα πλαίσια της παρούσας διπλωματικής εργασίας, η συγκεκριμένη τεχνολογία αξιοποιήθηκε ως εξής. Έχοντας το XML Schema για τη γλώσσα ερωτήσεων MP7QL ουσιαστικά δημιουργήσαμε ένα JAVA API σύμφωνα με το οποίο μπορούσαμε να δημιουργήσουμε – διαβάσουμε ένα Mp7ql query με αυτόματο τρόπο και σύμφωνα πάντα με τους Java τύπους.

3.7 Ανακεφαλαίωση

Στο συγκεκριμένο κεφάλαιο εξετάσαμε – περιγράψαμε τις διάφορες τεχνολογίες που χρησιμοποιήθηκαν κατά την υλοποίηση της συγκεκριμένης εργασίας και πιο συγκεκριμένα αναφερθήκαμε στο Stanford POS Tagger, στο JWordnet, στη SPARQL, στο GraphOnto και στα Axis Web Services . Στο επόμενο κεφάλαιο θα περιγράψουμε την αρχιτεκτονική του **OntoNL,** θα αναλύσουμε τις διάφορες υπομονάδες της και θα εξετάσουμε τη λειτουργικότητα της κάθε μίας.

ΚΕΦΑΛΑΙΟ 4
Η ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΟΥ ONTONL

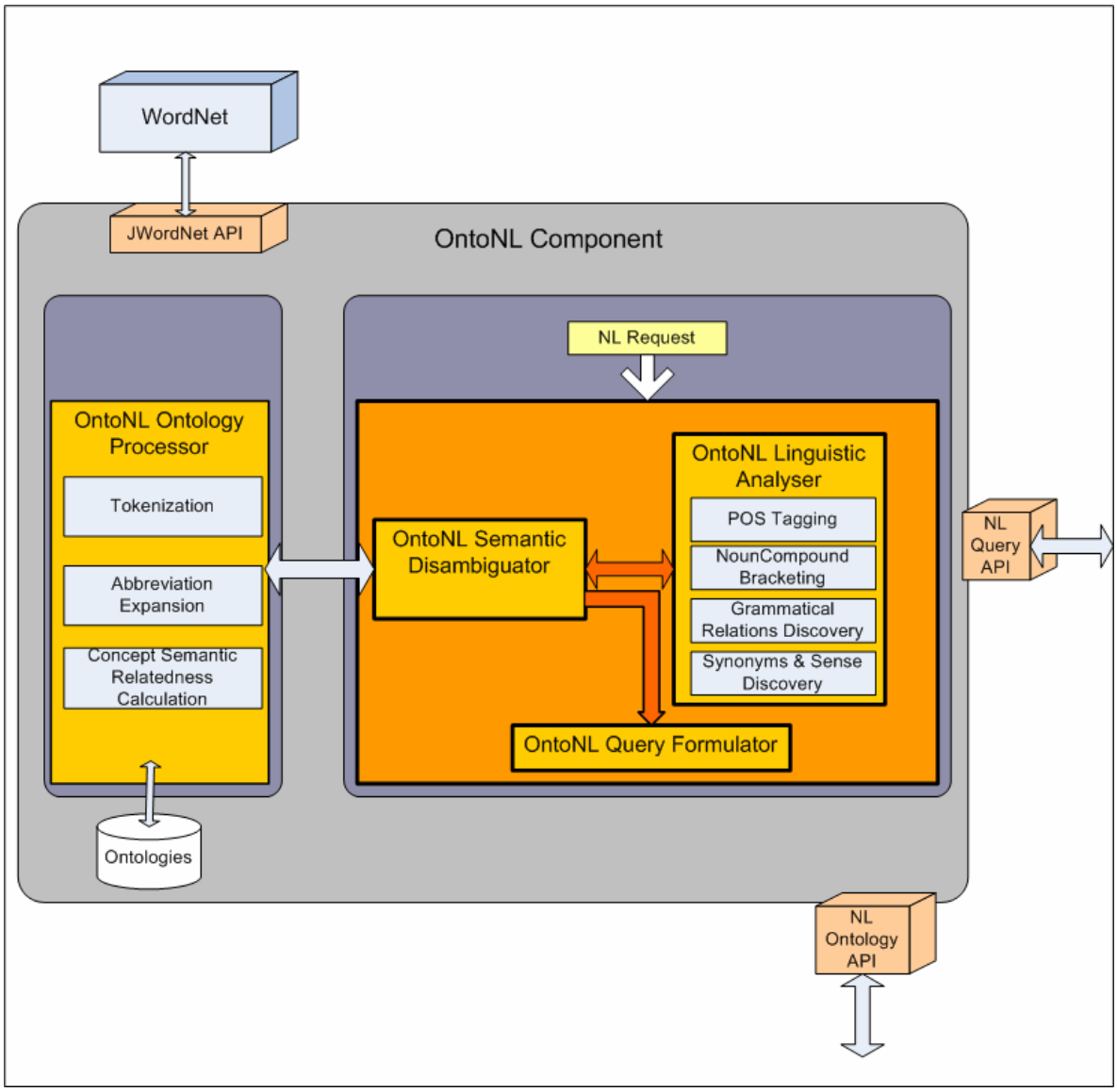
4.1 Εισαγωγή

Στο προηγούμενο κεφάλαιο εξετάσαμε τις διάφορες τεχνολογίες που χρησιμοποιήθηκαν κατά την υλοποίηση της παρούσας διπλωματικής εργασίας. Στο κεφάλαιο αυτό θα παρουσιαστεί η αρχιτεκτονική του **OntoNL Freamwork** [1] και θα περιγραφούν οι διάφορες υπομονάδες του. Σκοπός μας είναι να παράγουμε επαναχρησιμοποιήσιμες γενικευμένες μεθοδολογίες που θα διευκολύνουν τη χρήση διεπαφών φυσικής γλώσσας για αλληλεπιδράσεις χρηστών – βάσεων γνώσεις (knowledge repositories) ανεξαρτήτως οντολογιών περιοχής (domain ontologies).

Όπως είναι φυσικό αφού μιλάμε για αλληλεπιδράσεις φυσικού λόγου, υπάρχει πάντα η πιθανότητα να εμφανιστούν διαφόρων μορφών ασάφειες στο αίτημα του χρήστη. Έτσι λοιπόν, δίνεται έμφαση στο να μειωθούνε αυτές οι ασάφειες όσο το δυνατόν περισσότερο με την ανάπτυξη διάφορων μεθοδολογιών που θα αναλυθούνε παρακάτω.

4.2 Γενική Αρχιτεκτονική

Η αρχιτεκτονική του **OntoNL** σύμφωνα με το [1] αποτελείται από τέσσερα βασικά κομμάτια (components), το **OntoNL Ontology Processor**, το **OntoNL Linguistic Analyzer,** το **OntoNL Semantic Disambiguator** και το **OntoNL Query Formulator** (βλ. εικόνα 11)



Εικόνα 11: Η αρχιτεκτονική του OntoNL

Κάθε ένα από αυτά τα components αποτελούνται με τη σειρά τους από άλλες μικρότερες υποενότητες. Ακολουθεί λοιπόν η ανάλυση αυτών των components καθώς και η χρησιμότητα – λειτουργικότητα τους.

4.3 Η υπομονάδα OntoNL Επεξεργαστή Οντολογίας (Ontology Processor)

Όπως φαίνεται και στην εικόνα 11, ο **OntoNL Ontology Processor** εμπεριέχει τις υπομονάδες **Tokenization**, **Abbreviation Expansion** και **Concept Semantic Relatedness Calculation** που κάθε μία από αυτές επιτελεί διαφορετική λειτουργία και θα αναλύσουμε κάθε μία από αυτές.

Κοινός απώτερος σκοπός και των τριών αυτών υπομονάδων είναι ότι προσπαθούν να επεξεργαστούν την οντολογία με τέτοιο τρόπο έτσι ώστε να είναι πιο εύκολο προς

το σύστημα να επικοινωνήσει με αυτή σε μελλοντικές αλληλεπιδράσεις. Η οντολογία ουσιαστικά με διάφορες μεθόδους που έχουν αναπτυχθεί στη συγκεκριμένη μονάδα του **OntoNL Ontology Processor** επιχειρείται να γίνει πιο "φιλική" προς χρήση και ο λόγος θα φανεί αμέσως παρακάτω.

Το **Tokenization** process είναι μια προσπάθεια να αναλύσουμε την εκάστοτε λέξη εάν αυτό είναι δυνατό στα συνθετικά της (εάν αυτά υπάρχουν). Αυτό γίνεται εφικτό εάν στη λέξη παρατηρήσουμε αλλαγές στα γράμματα (κεφαλαία - μικρά) ή αν υπάρχουν μέσα στη λέξη διάφοροι χαρακτήρες όπως οι χαρακτήρες **"."** ή **"-"** ή **"_"** που λειτουργούν ως διαχωριστικά ανάμεσα στις περιεχόμενες λέξεις. Ο λόγος ύπαρξης αυτής της ενότητας γίνεται άμεσα αντιληπτός αν σκεφτούμε ότι στις οντολογίες συνήθως τα ονόματα των κλάσεων για παράδειγμα είναι σύνθετα έτσι ώστε με την ονοματολογία τους να υποδηλώνουν και τη λειτουργικότητα τους (π.χ SoccerTeam που υποδηλώνει μία σύνθετη έννοια, δηλαδή μια ποδοσφαιρική ομάδα – Soccer Team-). Για να μπορέσει λοιπόν η ονοματολογία των κλάσεων να έρθει πιο «κοντά» στο φυσικό λόγο γίνεται η προαναφερθείσα επεξεργασία.

Το **Abbreviation Expansion** process είναι μια διαδικασία σύμφωνα με την οποία συμβουλευόμαστε το λεξικό της εφαρμογής TypeFast abbreviation expander (http://www.topshareware.com/TypeFast-download-3297.htm) με σκοπό να αντιστοιχίσουμε τις λέξεις που προέκυψαν από τη διαδικασία του **Tokenization** με αυτές που μας συμβουλεύει το λεξικό της εφαρμογής εάν αυτό είναι εφικτό. Ο λόγος είναι ότι στην ονοματολογία που χρησιμοποιείται στις οντολογίες συχνά χρησιμοποιούνται διάφορες συντομογραφίες που όμως δε μπορούν να αντιστοιχηθούν στο φυσικό λόγο (π.χ socc – θέλοντας να εννοηθεί soccer). Με αυτή τη μεθοδολογία λοιπόν ξεπερνιέται αυτό το εμπόδιο αφού από το συγκεκριμένο λεξικό μπορούμε να βρούμε ποιες είναι οι πιθανές λέξεις που προτείνονται για κάθε πιθανή συντομογραφία.

Η διαδικασία **Concept Semantic Relatedness Calculation** είναι η προσπάθεια να υπολογιστεί ο βαθμός συσχέτισης μιας κλάσης με όλες τις υπόλοιπες, δηλαδή το πόσο πολύ μπορεί να συσχετίζεται μια κλάση με μια άλλη μέσα στην οντολογία σύμφωνα πάντα με τον αλγόριθμο **ONTONL Semantic Relatedness Measure Algorithm** [2,3,4] που έχοντας ως είσοδο την εκάστοτε οντολογία περιοχής (domain ontology) και διάφορα βάρη που προκύπτουν πειραματικά υπολογίζει παράγοντες οι οποίοι αναδεικνύουν το βαθμό συσχέτισης. Σε γενικές γραμμές, ο συγκεκριμένος αλγόριθμος αποτελείται από 3 μετρικές, τη μετρική ιδιοτήτων, τη μετρική

εννοιολογικής απόστασης και τη μετρική κοινών εννοιών. Περισσότερες λεπτομέρειες για τον αλγόριθμο μπορούν να βρεθούν στα [1,2,3,4]. Σκοπός αυτός του σταδίου είναι να συμβουλευόμαστε τα αποτελέσματα του αλγορίθμου σε περιπτώσεις που εμφανίζονται ασάφειες στο αίτημα του χρήστη με απώτερο σκοπό να ξεπεραστούν αυτές οι ασάφειες και να αντιμετωπιστεί η πρόταση έχοντας πλέον αντικαταστήσει - τροποποιήσει τα σημεία που εμφανίζονται οι ασάφειες αυτές με τα πιο συσχετιζόμενα αποτελέσματα που συμβουλευόμαστε από τον αλγόριθμο.

Έτσι λοιπόν συνοψίζοντας, γίνεται αντιληπτό ότι στο component **OntoNL Ontology Processor** έχει αναπτυχθεί μεθοδολογία μέσα από διαφορετικά στάδια (Tokenization, Abbreviation Expansion, Concept Relatedness Calculation) που επεξεργάζεται την εκάστοτε οντολογία με σκοπό να παρέχει δομές που προσεγγίζουν την οντολογία από πλευράς φυσικού λόγου αφού προνοεί τόσο για ενδεχόμενες ασάφειες που μπορεί να εμφανίζονται στο αίτημα του χρήστη όσο και για γλωσσολογικές "συγχύσεις" που καθιστούν την αλληλεπίδραση χρήστη – οντολογίας αδύνατη.

4.4 Η υπομονάδα γλωσσολογικής ανάλυσης (OntoNL Linguistic Analyzer)

Το component Linguistic Analyzer όπως μπορεί να διακριθεί και στην εικόνα 11 αποτελείται από τις μεθοδολογίες **POS Tagging**, **Noun Compound Bracketing**, **Grammatical Relations Discovery** και **Synonyms and Sense Discovery**.

Επίσης, στο **OntoNL Linguistic Analyzer** υπάρχει ένας μηχανισμός μετατροπής του αιτήματος του χρήστη (**Request Conversion Mechanism**) στον οποίο γίνεται η προσπάθεια να παραμείνει στο αίτημα μόνο η χρήσιμη πληροφορία που προσδίδει σημαντικότητα σε αυτό. Αυτός ο μηχανισμός είναι το πρώτο στάδιο επεξεργασίας στο **OntoNL Linguistic Analyzer** και στη συνέχεια ακολουθούνε οι διάφορες προαναφερθείσες μεθοδολογίες.

Γενικότερα στην υπομονάδα γλωσσολογικής ανάλυσης αναπτύχθηκαν μέθοδοι για τη συντακτική και γλωσσολογική αποσαφήνιση της πρότασης φυσικού λόγου. Γίνεται επίσης προσπάθεια να απομονωθεί με διάφορες τεχνικές το μέρος της πρότασης που περιέχει τη χρήσιμη πληροφορία. Έτσι λοιπόν ακολουθεί μια πρώτη, σύντομη αναφορά σε κάθε μία από αυτές τις υπομονάδες.

Στο στάδιο **Request Conversion**, το **OntoNL Framework** [1] ορίζει 3 διαφορετικού τύπου προτάσεις. Πιο συγκεκριμένα:

- **Αίτημα για δεδομένα** (π.χ. Show me the goals scored in the game between Italy and France)
- **WH-questions** (π.χ. What was the score in the game between Italy and France?)
- **Yes/No questions** (π.χ. Were there any goals in the game between Italy and France?)

Στην είσοδο του μηχανισμού request conversion, γίνεται ανίχνευση του τύπου στον οποίο ανήκει η πρόταση και γίνεται η ανάθεση του σε ένα δείκτη. Αυτός ο δείκτης τελικά, όπως θα δειχθεί και παρακάτω, εξάγεται μέσω του **NL Query API** (εικόνα 11), έτσι ώστε να έχει τη δυνατότητα το σύστημα ανάλογα με το τύπο ερώτησης να παρέχει το κατάλληλο τύπο απάντησης, αφού αναφερόμαστε σε question – answering συστήματα. Μετά το convertion οι προτάσεις θα γίνουν:

1. the goals scored in the game between Italy and France
2. the score in the game between Italy and France
3. any goals in the game between Italy and France

Περισσότερες λεπτομέρειες για το συγκεκριμένο στάδιο αναφέρονται στο κεφάλαιο 5 που περιγράφεται η υλοποίηση της αρχιτεκτονικής.

Στην υπομονάδα **POS Tagging** γίνεται χρήση του POS Tagger [http: // nlp.stanford.edu/ software/tagger.shtml] που έχει αναλυθεί στο κεφάλαιο 3 με σκοπό να πάρουμε μια δενδροειδή αναπαράσταση από την οποία μπορούμε να γνωρίζουμε τι μέρος του λόγου είναι κάθε λέξη της πρότασης καθώς και την αλληλουχία τους μέσα σε αυτήν.

Στην υπομονάδα **Noun Compound Bracketing** γίνεται η προσπάθεια να αναγνωριστούν τα noun compounds, δηλαδή να ομαδοποιηθούν λέξεις που ενδεχομένως να αντιστοιχίζονται στις λέξεις που προέκυψαν από το **Abbreviation Expansion** mechanism του component **OntoNL Ontology Processor**. Στη προσπάθεια μας να κάνουμε την οντολογία πιο φιλική προς το σύστημα είναι απαραίτητο να κρατάμε λέξεις του φυσικού λόγου που συνδέονται άρρηκτα μεταξύ τους έτσι ώστε ο συνδυασμός τους να λαμβάνει μέρος στα μετέπειτα στάδια επεξεργασίας του **OntoNL** ως ένα ενιαίο συντακτικό σύνολο. Λέξεις που στον φυσικό λόγο είναι ξεχωριστές, μπορούν συνδυαζόμενες κατάλληλα να αποτελέσουν όρο που μπορεί να αντιστοιχηθεί στην οντολογία.

Στην υπομονάδα **Grammatical Relations Discovery** γίνεται η συντακτική ανάλυση της πρότασης και ότι αυτό συνεπάγεται [1]. Η ανακάλυψη των γραμματικών σχέσεων μέσα σε μία πρόταση θα βοηθήσει στο να δημιουργηθεί ένα κατάλληλο γλωσσικό μοντέλο που θα βοηθήσει το σύστημα να ανακτήσει τα σωστά δεδομένα από το repository. Με άλλα λόγια, αποτελεί τη σημασιολογική βάση από την οποία θα γίνει εξαγωγή πληροφορίας

Τέλος, στην υπομονάδα **Synonyms & Sense Discovery** μπορούμε να ανακτήσουμε τα διάφορα συνώνυμα που έχει μία λέξη. Ο λόγος για τον οποίο χρειαζόμαστε τα συνώνυμα των λέξεων είναι γιατί ο χρήστης στο φυσικό λόγο ενδεχομένως να χρησιμοποιεί όρους που να μην αντιστοιχίζονται στην οντολογία, αλλά υπάρχει πιθανότητα κάποιο από τα συνώνυμα αυτής της λέξης να αποτελεί ικανό όρο ώστε να γίνει η αντιστοίχηση.

4.5 Η υπομονάδα εννοιολογικής αποσαφήνισης (Semantic Disambiguator)

Γενικότερα σε αυτό το στάδιο γίνεται η εννοιολογική αποσαφήνιση των όρων που έχουν εμφανιστεί, δηλαδή η προσπάθεια να αντιστοιχίσουμε τους όρους αυτούς με ανάλογους της οντολογίας. Στα πλαίσια αυτής της επεξεργασίας, γίνεται επίσης προσπάθεια να αντιμετωπιστούν – ξεπεραστούν διάφορες ασάφειες που εμφανίζονται στο αίτημα του χρήστη. Οι τύποι των ασαφειών που μπορεί να εμφανιστούν είναι οι εξής:

- Τύπος ασάφειας 1
    - “players of soccer team Barcelona”
- Τύπος ασάφειας 2
    - “players of Barcelona” (Barcelona = city, soccer team, etc.)
- Τύπος ασάφειας 3
    - “mvp of Barcelona versus Real” (mvp = ?, Barcelona = city?, soccer team? .., Real = soccer team?, person? ..)

Στο τύπο ασάφειας 1 οι ασάφειες μπορούν να αντιμετωπιστούν χρησιμοποιώντας μόνο τη γνώση της οντολογίας, αφού λέξεις όπως player, soccer team μπορούν να αντιστοιχηθούν σε κλάσεις.

Στο τύπο ασάφειας 2 υπάρχει μεγαλύτερος βαθμός ασάφειας αφού η λέξη Barcelona δε μπορεί να αντιστοιχηθεί σε κάποιο όρο της οντολογίας. Ο τρόπος με τον

οποίο αντιμετωπίζονται τέτοιου είδους ασάφειες απαιτεί πιο συστηματική μεταχείριση, αφού χρειάζεται να συμβουλευτούμε βαθμούς συσχέτισης ανάμεσα σε κλάσεις οντολογιών.

Στο τύπο ασάφειας 3 αντιμετωπίζουμε ασάφειες για τις οποίες δε γνωρίζουμε οτιδήποτε αφού οι λέξεις που χρησιμοποιούνται ή είναι instances κλάσεων ή είναι απλές λέξεις, με αποτέλεσμα να μη μπορούμε να ανακτήσουμε από την οντολογία πληροφορία που να μας βοηθάει να τις αντιμετωπίσουμε.

4.6 Η υπομονάδα του συντάκτη ερωτήσεων (Query Formulator)

Σε αυτό το στάδιο γίνεται η τελική επεξεργασία του **OntoNL**. Ουσιαστικά έχουν υλοποιηθεί μέθοδοι οι οποίοι μετατρέπουν σε SPARQL τη δομή που έχει προκύψει από τα στάδια επεξεργασίας της γλωσσολογικής και εννοιολογικής αποσαφήνισης.

Επιλέγουμε τη SPARQL ως γλώσσα διατύπωσης ερωτήσεων για να αντιπροσωπεύσουμε τις ερωτήσεις φυσικής γλώσσας μετά από τη συντακτική και σημασιολογική αποσαφήνιση, δεδομένου ότι η SPARQL ορίζεται στο W3C's RDF data model και μπορεί να απευθυνθεί σε οποιαδήποτε πηγή δεδομένων μπορεί να αντιστοιχηθεί σε RDF

Γενικότερα, δημιουργείται ένα query το οποίο αντιπροσωπεύει το αίτημα του χρήστη κάνοντας αναφορά τόσο σε όρους του φυσικού λόγου όσο και σε όρους της οντολογίας αντίστοιχα. Πιο συγκεκριμένα στο SPARQL query που δημιουργείται από το component **OntoNL Query Formulator** παρουσιάζεται ένα «μονοπάτι» στο οποίο κλάσεις (concepts) της οντολογίας συνδέονται μέσω object properties με άλλες κλάσεις ή μέσω data properties με δεδομένα Το query αυτό στη συνέχεια είναι έγκυρο έτσι ώστε να μπορεί να ερωτηθεί σε μία βάση δεδομένων ώστε να ανακτήσουμε αποτελέσματα , και επίσης μπορεί να χρησιμοποιηθεί ως πρότυπο query που μπορεί να μετατραπεί με τη βοήθεια οποιαδήποτε άλλης query language. Μια τέτοια υλοποίηση έχει πραγματοποιηθεί στα πλαίσια αυτής της διπλωματικής εργασίας και αναλύεται στο κεφάλαιο 6 ,στο οποίο εξηγείται η μετατροπή του SPARQL query σε Mp7Ql query.

4.7 Το NL Query API και το NL Ontology API

Το OntoNL Framework προσφέρει υποστήριξη για την αποσαφήνιση φυσικής γλώσσας σε βάσεις γνώσης. Το **NL Query API** παίρνει σαν είσοδο ένα αίτημα σε φυσικό λόγο και ύστερα από διάφορα στάδια που μόλις αναλύθηκαν παράγει ως έξοδο έναν αριθμό από βεβαρυμμένα (weighted) SPARQL queries (τα βάρη προκύπτουν από το στάδιο αποσαφήνισης και εξαρτώνται από τις ασάφειες που παρουσιάζονται) βασισμένα στην οντολογία που χρησιμοποιήθηκε για αποσαφήνιση καθώς και το τύπο της απάντησης που παρέχεται, έτσι όπως αυτός έχει προκύψει από την επεξεργασία του Request Conversion.

Το **NL Ontology API** περιέχει το σύνολο των συναρτήσεων που είναι υπεύθυνες για τη διαχείριση της εκάστοτε οντολογίας. Συναρτήσεις όπως insertOntology( ), storeOntology( ), getClasses( ) και έναν αριθμό από διάφορες συναρτήσεις που επεξεργάζονται την εκάστοτε οντολογία.

4.8 Ανακεφαλαίωση

Συνοψίζοντας, στο συγκεκριμένο κεφάλαιο περιγράφθηκε συνοπτικά η αρχιτεκτονική του **OntoNL** πλαισίου, είδαμε από ποιες υπομονάδες αποτελείται και ποια είναι η λειτουργικότητα της κάθε μίας ξεχωριστά. Λεπτομέρειες για το πλήρες μοντέλο της Αρχιτεκτονικής υπάρχουν στο [1] Στο επόμενο κεφάλαιο θα εξετάσουμε πολύ πιο αναλυτικά και με τη χρήση παραδειγμάτων τα στάδια υλοποίησης του **OntoNL**

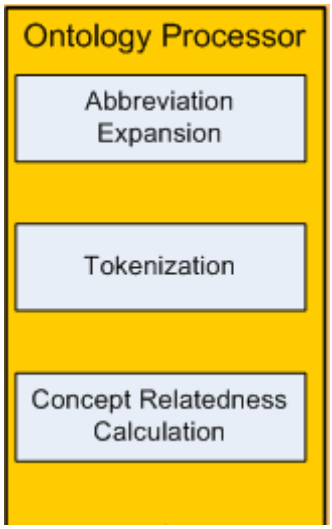
ΚΕΦΑΛΑΙΟ 5
ΥΛΟΠΟΙΗΣΗ ΤΟΥ ONTONL FRAMEWORK

5.1 Εισαγωγή

Στο προηγούμενο κεφάλαιο εξετάσαμε την αρχιτεκτονική του **OntoNL** [1] πλαισίου και είδαμε σε γενικές γραμμές τις υπομονάδες που την αποτελούν καθώς και τη λειτουργικότητα τους. Στο παρών κεφάλαιο, θα δούμε αναλυτικότερα αυτά τα στάδια, πως αυτά υλοποιούνται καθώς και παραδείγματα που θα κάνουν τη κατανόηση τους ευκολότερη.

5.2 Υλοποίηση του Επεξεργαστή Οντολογίας (Ontology Processor)

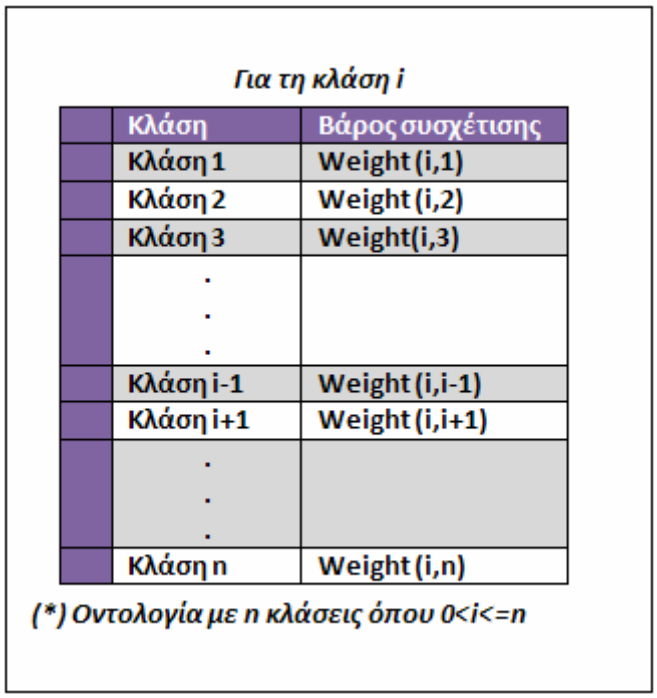
Αρχικά, αξίζει να σημειωθεί πως το component **Ontology Processor** ανήκει στη φάση του deployment. Αυτό σημαίνει ότι κατά τη διάρκεια αυτού του σταδίου ,που γίνεται offline, το αποτέλεσμα είναι η δημιουργία κάποιων αρχείων τα οποία ναι μεν απαιτούνε κάποιο χρόνο για να δημιουργηθούνε αλλά αφού δημιουργηθούνε μπορούν να χρησιμοποιηθούν στη φάση του runtime από την εφαρμογή για όσες φορές κριθεί απαραίτητο. Είναι προτιμότερο η επεξεργασία της οντολογίας να γίνει μία φορά και όταν η εφαρμογή δε χρησιμοποιείται παρά να γίνεται κάθε φορά που ο χρήστης επιθυμεί να χρησιμοποιήσει την εφαρμογή. Το μόνο που απαιτείται είναι να είναι διαθέσιμη η οντολογία περιοχής (domain ontology) στην οποία θα γίνει η επεξεργασία και η οποία στη συνέχεια θα χρησιμοποιηθεί και από το OntoNL.



Εικόνα 12: Το Component Ontology Processor

Αναλυτικότερα, στην υποενότητα **Concept Relatedness Calculation** δημιουργούνται δύο αρχεία, το πρώτο έχει σχέση με το βαθμό συσχέτισης μεταξύ των κλάσεων και είναι αποτέλεσμα του αλγορίθμου **OntoNL Semantic Relatedness Measure Algorithm** [2,3,4] που αναφέρθηκε στο κεφάλαιο 4 και το δεύτερο σχετίζεται με το βέλτιστο μονοπάτι μεταξύ δύο κλάσεων μέσω object properties, κάτι που είναι απαραίτητο και πολύ σημαντικό για τη δημιουργία του SPARQL query, αφού η γνώση του μονοπατιού που υποδεικνύει τη μετάβαση από μια κλάση σε μία άλλη κρίνεται αναγκαία όπως θα φανεί και παρακάτω. Απαραίτητα για τη δημιουργία αυτών των αρχείων είναι φυσικά η οντολογία η οποία θα χρησιμοποιηθεί από την εφαρμογή και κάποιο ontology API το οποίο να μεταχειρίζεται την οντολογία (GraphOnto API).

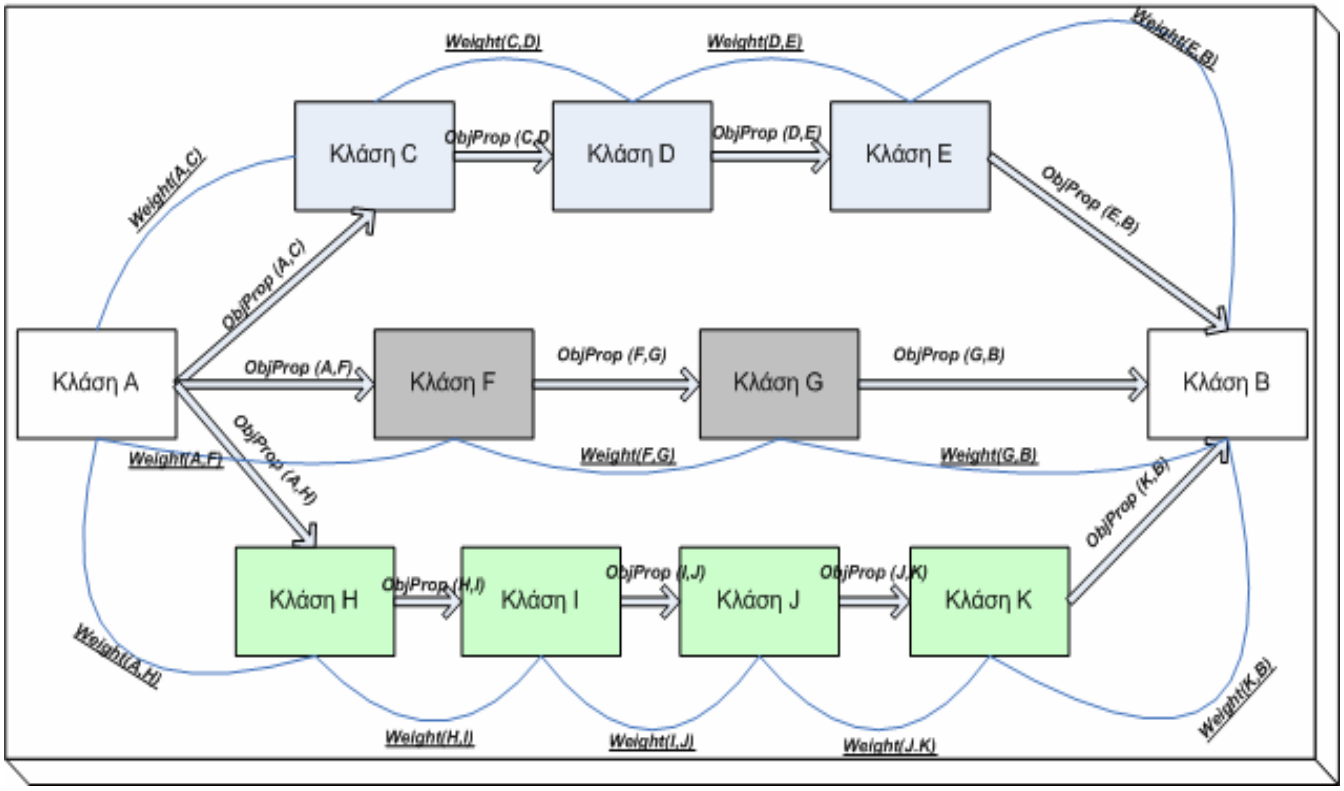
Πιο συγκεκριμένα, η δομή του πρώτου αρχείου είναι αυτή που φαίνεται στο παρακάτω σχήμα και δείχνει ότι για κάθε κλάση της οντολογίας έχει υπολογιστεί ο βαθμός συσχέτισης της με τις υπόλοιπες. Ουσιαστικά δηλαδή, έχει αποθηκευτεί σε ένα αρχείο κάθε πιθανός συνδυασμός κλάσης με όλες τις υπόλοιπες και ποιο είναι το βάρος συσχέτισης (έχει προκύψει από τον αλγόριθμο) που τις χαρακτηρίζει.



Εικόνα 13: Αναπαράσταση δομής αρχείου με βάρη συσχέτισης στο στάδιο Ontology Processor

Για τη δημιουργία του δεύτερου αρχείου στο στάδιο **Concept Relatedness Calculation** απαιτείται η δημιουργία του πρώτου αρχείου αφού είναι απαραίτητα τα βάρη συσχέτισης που ήδη έχουν υπολογιστεί και αποθηκευτεί σε αυτό. Για κάθε κλάση της οντολογίας υπάρχουν όλοι οι πιθανοί συνδυασμοί της με τις υπόλοιπες κλάσεις με τη διάφορα ότι εδώ δεν αποθηκεύεται ο βαθμός συσχέτισης αλλά η διαδρομή μέσω object properties με το μεγαλύτερο κανονικοποιημένο βάρος. Αναλυτικότερα:

Σε ένα υποθετικό παράδειγμα, έστω ότι υπάρχει η κλάση Α και η κλάση Β και υπάρχουν 3 διαφορετικοί τρόποι για να γίνει μετάβαση από τη κλάση Α στη κλάση Β μέσω object properties (οι ενδιάμεσοι σταθμοί είναι και αυτοί κλάσεις της οντολογίας). Αυτοί φαίνονται παρακάτω:



Εικόνα 14: Αναπαράσταση μεταβάσεων μεταξύ κλάσεων μέσω object properties

Στο παραπάνω παράδειγμα λοιπόν φαίνονται 3 διαφορετικοί τρόποι μετάβασης μέσω object properties από τη κλάση A στη κλάση B. Με βελάκια παρουσιάζονται τα αντίστοιχα object properties που συνδέουν τις εκάστοτε κλάσεις ενώ με καμπύλες γραμμές αναπαρίστανται τα βάρη συσχέτισης ανάμεσα στις κλάσεις που είναι ήδη υπολογισμένα από προηγούμενο στάδιο επεξεργασίας.

Για να επιλεχθεί ποια από τις 3 διαδρομές είναι η καλύτερη γίνεται το εξής (σε αυτό το σημείο γίνεται αξιοποίηση του πρώτου αρχείου):

Επιλέγεται η διαδρομή η οποία έχει μεγαλύτερο "σχετικό" βάρος, δηλαδή η διαδρομή που το άθροισμα των βαρών ανάμεσα στις αντίστοιχες κλάσεις δια του αριθμού των ενδιάμεσα κλάσεων είναι ο μεγαλύτερος Τα βάρη ανάμεσα στις κλάσεις είναι ήδη γνωστά και υπολογισμένα. Επομένως αν

```
w1 = ((weight A,C)+ weight(C,D) +weight(D,E) +weight(E,B))/3

w2 = ((weight A,F)+(weight F,G) + weight(G,B))/2

w3 = ((weight A,H)+(weight H,I) + weight(I,J) +weight(J,K) +weight(K,B))/4
```

επιλέγεται και αποθηκεύεται η διαδρομή με το μεγαλύτερο σχετικό βάρος w. Εννοείται πως αν υπάρχει διαδρομή που συνολικά υπάρχουν 2 κλάσεις ,δηλαδή η

κλάση Α να συνδέεται άμεσα με τη κλάση Β μέσω object property τότε επιλέγετε αυτή η διαδρομή αφού ο αριθμός των ενδιάμεσων κλάσεων είναι 0 και το σχετικό βάρος είναι άπειρο.

Επομένως, σύμφωνα με τα παραπάνω, η δομή του δεύτερου αρχείου θα είναι:

Για τη κλάση i

	Κλάση	Μονοπάτι μέσω Object properties
	Κλάση 1	Path (i,1)
	Κλάση 2	Path (i,2)
	Κλάση 3	Path(i,3)
	. . .	
	Κλάση i-1	Path (i,i-1)
	Κλάση i+1	Path (i,i+1)
	. . .	
	Κλάση n	Path (i,n)

(*) Οντολογία με n κλάσεις όπου 0<i<=n

Εικόνα 15: Αναπαράσταση δομής αρχείου με βέλτιστες διαδρομές μέσω object properties ανάμεσα στις κλάσεις της οντολογίας στο στάδιο Ontology Processor

Τέλος, από τη συνεργασία των υποενοτήτων **Tokenization** και **Abbreviation Expansion,** δημιουργείται ένα αρχείο που σχετίζεται με τη γλωσσολογική αντιμετώπιση της οντολογίας.

Πιο συγκεκριμένα, έχει ήδη αναφερθεί ότι στο στάδιο προεργασίας της οντολογίας γίνεται μια προσπάθεια να έρθει η ονοματολογία της οντολογίας πιο κοντά στο φυσικό λόγο. Δηλαδή οι κλάσεις της οντολογίας που μπορεί να έχουν σύνθετα ονόματα ή και συντομογραφίες προσπαθούν να αποσαφηνιστούν με τέτοιο τρόπο ώστε ο χρήστης χρησιμοποιώντας παρόμοιους όρους στο φυσικό λόγο να μπορέσει να καταστήσει σαφές ότι μιλάει για κάποιο concept της οντολογίας. Συγκεκριμένα η διαδικασία που ακολουθείται είναι η εξής:

Αρχικά στο στάδιο **Tokenization** γίνεται προσπάθεια να αναλυθεί το όνομα κάθε κλάσης της οντολογίας σε λέξεις που μπορεί αυτή να περιέχει, αν φυσικά πρόκειται για κάποιο όνομα που αποτελεί σύνθετη λέξη. Επομένως, σε αυτό το στάδιο, γίνεται

προσπάθεια να χωριστεί το σύνθετο όνομα στις λέξεις οι οποίες την αποτελούν. Έτσι λοιπόν ο διαχωρισμός αυτός γίνεται αν μέσα στη λέξη παρουσιαστούν οι ειδικοί χαρακτήρες **"."** ή **"-"** ή **"_"** και επίσης αν ύστερα από κάποια σειρά γραμμάτων που είναι "μικρά" εμφανιστεί κάποιο κεφαλαίο (ή και αντίστροφα) που υποδηλώνει ότι ξεκινάει μια καινούρια λέξη .

```
Program List Tokenize (String className)
    List returnedVector;
    Begin
    String [] tokens= className.split("-");
    For all terms i of tokens
    {
      String temp=tokens[i];
      String[] tokens2=temp.split("_");
      For all terms j of tokens2 {
        String temp2=tokens2[j];
        String[] tokens3=temp2.split("_");
        For all terms k of tokens3 {
        String temp3=tokens3[k];
        Vector words=upperCaseSeparation(temp3);
     }
   }
  }
 return returnedVector;
End
```

Ακολουθούν κάποια παραδείγματα που εξηγούν τη παραπάνω μεθοδολογία. Αρχικά παρουσιάζεται η λέξη που αντιπροσωπεύει τη κλάση της εκάστοτε οντολογίας και τελικά η λέξη αυτή μετά το στάδιο επεξεργασίας **Tokenization.**

Στάδιο Tokenization

Αρχικά	**Τελικά**
SoccerTeam	Soccer Team
SoccerTeamObject	Soccer Team Object
FoulAction	Foul Action
PenaltyKick	Penalty Kick
Player	Player

Εικόνα 16:Παράδειγμα αποτελεσμάτων από τη φάση Tokenization

Φυσικά υπάρχουν και οι περιπτώσεις που η λέξη δεν είναι σύνθετη όπως το τελευταίο παράδειγμα με τη λέξη Player που τελικά προκύπτει το ίδιο αποτέλεσμα.

Έτσι λοιπόν, και μόνο από αυτό το στάδιο, έχει γίνει κάποιο είδος πρώτης αποσαφήνισης αφού αν ο χρηστής αναφέρει στο φυσικό λόγο τους όρους soccer team γίνεται αμέσως αντιληπτό ότι αναφέρεται στο concept SoccerTeam της οντολογίας.

Όλα τα παραπάνω γίνονται σύμφωνα πάντα με το μηχανισμό του **Tokenization**. Αφού έχει τελειώσει αυτού του είδους η επεξεργασία, ακολουθεί το στάδιο του **Abbreviation Expansion**. Σε αυτό το στάδιο γίνεται αξιοποίηση των λέξεων που προέκυψαν από το **Tokenization** και συμβουλευόμενοι ένα θησαυρό λέξεων που περιέχει συντομογραφίες και πως μπορούν αυτές να ερμηνευτούν καταλήγουμε στη τελική μορφή της λεξικολογικής ανάλυσης των ονομάτων των κλάσεων.

Συγκεκριμένα ο θησαυρός λέξεων που χρησιμοποιείται είναι η βιβλιοθήκη που χρησιμοποιεί η εφαρμογή TypeFast abbreviation expander (http://www.topshareware .com/TypeFast-download-3297.htm). Έτσι λοιπόν γίνεται χρήση αυτής της βιβλιοθήκης και για κάθε συνδυασμό λέξεων που έχει προκύψει από το στάδιο του **Tokenization** δημιουργούνται οι συνδυασμοί που προτείνονται από τη συγκεκριμένη εφαρμογή.

```
Program List Expand (String word)
    List listOfWordsReturned;
    Begin
    String[] typeFastList=typeFastConsult(word);
    If(typeFastList.size()>0){
      For all terms j of typeFastList{
         String temp2=typeFastList[j];
          ListOfWordsReturned.add(temp2);
      }
     }

 return listOfWordsReturned;
End
```

Ένα παράδειγμα αυτής της εφαρμογής είναι ότι για τη λέξη socc προτείνεται η λέξη soccer και για τη λέξη obj προτείνεται τη λέξη object.

Για μια υποθετική κλάση με το όνομα SoccTeam, στο πρώτο στάδιο (Tokenization) που έχει ήδη αναφερθεί θα χωριζόταν το όνομα αυτό στις λέξεις Socc Team.

Στο δεύτερο στάδιο συμβουλευόμενοι πάντα το θησαυρό λέξεων παρατηρείται ότι για τη λέξη Socc προτείνεται η λέξη Soccer ενώ για τη λέξη Team δεν υπάρχει καμία πρόταση.

Έτσι λοιπόν για το όνομα κλάσης SoccTeam υπάρχουν οι εξής δυο εκδοχές:

Στάδιο Abbreviation Expansion

Αρχικά	**Τελικά**
Socc Team	Socc Team
	Soccer Team

Εικόνα 17: Παράδειγμα Abbreviation Expansion για τις λέξεις Socc Team

Με αυτό το τρόπο αν ο χρήστης στο φυσικό λόγο αναφέρει τους όρους soccer team αμέσως γίνεται αντιληπτό ότι αναφέρεται στο concept της οντολογίας με όνομα SoccTeam, γεγονός που δε θα ήταν σε καμία περίπτωση εφικτό αν δεν υπήρχε αυτού του είδους η επεξεργασία.

Όμοια για τη κλάση PlayerObj:

Στάδιο Abbreviation Expansion

Αρχικά	**Τελικά**
Player Obj	Player Obj
	Player Object

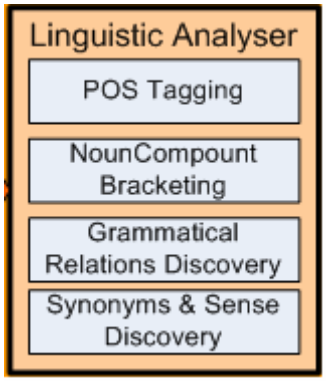
Εικόνα 18:Παράδειγμα Abbreviation Expansion για τις λέξεις Player Obj

Συνοψίζοντας, στο **OntoNL Ontology Processor**, γίνεται επεξεργασία της οντολογίας που απώτερο σκοπό έχει να βοηθήσει το σύστημα να αντιμετωπίσει καλύτερα τα μελλοντικά αιτήματα του χρήστη. Γίνονται λεξικολογικές αναλύσεις στα ονόματα των κλάσεων, έτσι ώστε αν ο χρήστης δε χρησιμοποιήσει ίδιους όρους με αυτούς της οντολογίας αλλά παραπλήσιους, να γίνεται αντιληπτό ότι αναφέρεται σε αυτούς. Επίσης, υπολογίζονται οι βαθμοί συσχέτισης μεταξύ κλάσεων, γεγονός που θα αποδειχθεί πολύ χρήσιμο κατά την αντιμετώπιση των ασαφειών στο μελλοντικό στάδιο της αποσαφήνισης του αιτήματος του χρήστη.

5.3 Υλοποίηση του OntoNL Γλωσσικού Αναλυτή (Linguistic Analyzer)

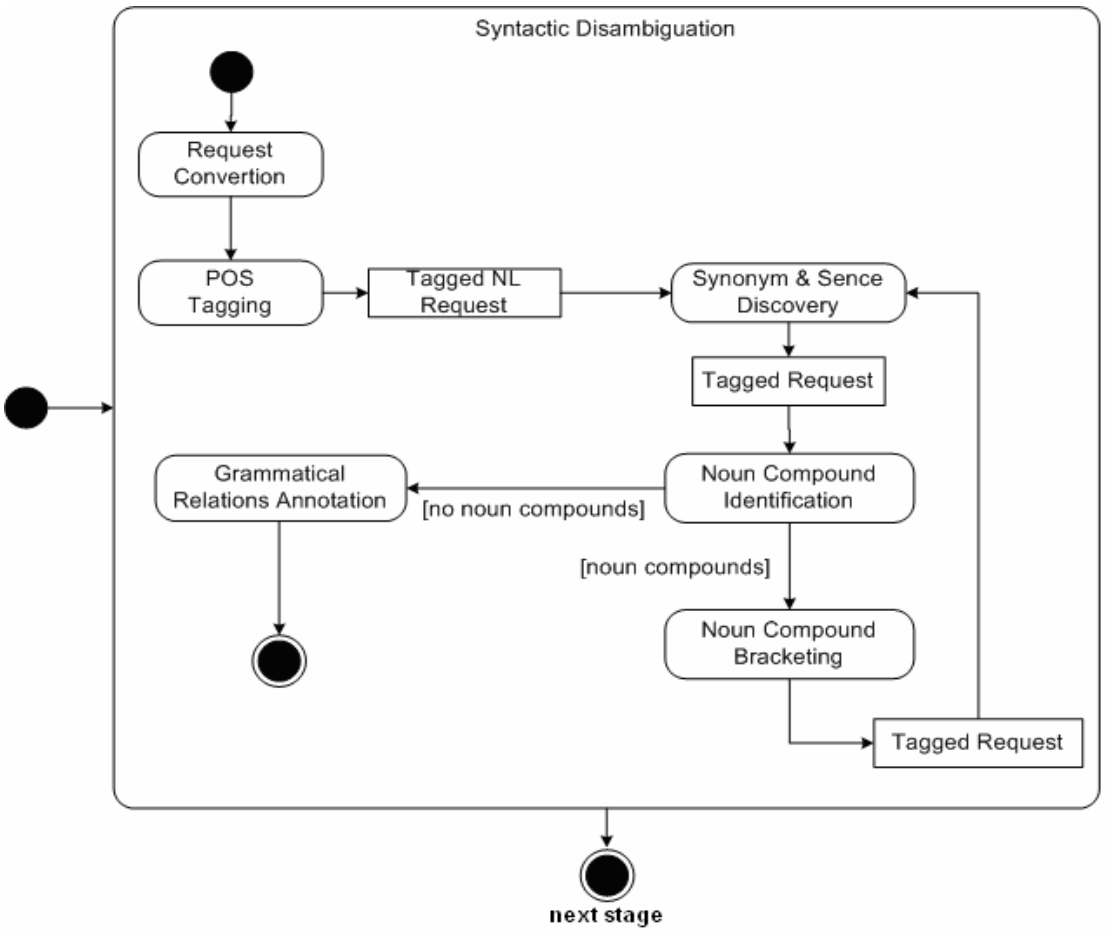
Σε αυτό το component γίνεται η γλωσσολογική ανάλυση της πρότασης μέσα από διάφορα στάδια επεξεργασίας και δημιουργείται μία δομή η οποία όπως θα εξηγηθεί παρακάτω χρησιμοποιείται από το επόμενο component της αρχιτεκτονικής του OntoNL, το **Semantic Disambiguator**. Κατά τη διάρκεια του σταδίου γλωσσολογικής ανάλυσης, τα διάφορα στάδια που λαμβάνουν μέρος είναι τα **POS**

Tagging, **Noun Compound Bracketing**, **Grammatical Relations Discovery** και το **Synonyms & Sense Discovery**.



Εικόνα 19:Το component Linguistic Analyzer

Ο τρόπος με τον οποία αυτά συνδέονται και ποια είναι η σειρά με τα οποία αυτά πραγματοποιούνται φαίνεται στο παρακάτω activity diagram που δείχνει τις ενέργειες που λαμβάνουν μέρος κατά το στάδιο της γλωσσολογικής ανάλυσης, το οποίο θα μπορούσε επίσης να ονομαστεί και στάδιο συντακτικής αποσαφήνισης (Syntactic Disambiguation) αφού σε αυτό το στάδιο γίνεται η συντακτική ανάλυση της πρότασης. Έτσι λοιπόν:



Εικόνα 20: Activity diagram του Linguistic Analyzer (Syntactic Disambiguation)[1]

Ακολουθεί η ανάλυση των ενεργειών που πραγματοποιούνται στο κάθε ένα από αυτά τα στάδια

5.3.1 Μετατροπή αιτήματος (Request Convertion)

Αυτό είναι το πρώτο στάδιο της επεξεργασίας της πρότασης. Γίνεται προσπάθεια να απομονωθεί η χρήσιμη πληροφορία της πρότασης και να «κοπεί» πληροφορία που είναι άχρηστη και δε προσδίδει σημαντικότητα στο περιεχόμενο της. Λέξεις ή φράσεις που στο φυσικό λόγο είναι πολύ συνηθισμένες , στο στάδιο αποσαφήνισης περισσότερο μπερδεύουν παρά διευκολύνουν την επεξεργασία της. Έτσι λοιπόν το γεγονός αυτό αποτελεί λόγο για να υπάρχει μία ειδική αντιμετώπιση των προτάσεων.

Υπάρχει ένα στάδιο προεπεξεργασίας της πρότασης, στο οποίο γίνεται η προσπάθεια να ανιχνευτεί τι είναι περιττό και να αφαιρεθεί έτσι ώστε όχι μόνο να γίνει η πρόταση μικρότερη που σημαίνει λιγότερη πολυπλοκότητα για τα επόμενα στάδια επεξεργασίας αλλά και να μείνει στη πρόταση όσο το δυνατόν περισσότερη

πληροφορία που είναι χρήσιμη ώστε τα αποτελέσματα που θα ανακτηθούν να είναι πιο σωστά και συγκεκριμένα. Φυσικά μπορεί να υπάρξουν περιπτώσεις που στο συγκεκριμένο στάδιο η πρόταση να μην υφίσταται καμία αλλαγή και να παραμένει η ίδια.

Κριτήριο για το στάδιο αποκοπής λέξεων μέσα σε προτάσεις είναι το γεγονός ότι η παρούσα υλοποίηση αφορά ένα σύστημα ερωτήσεων – απαντήσεων (question answering system), επομένως είναι σύνηθες αλλά συνάμα και άχρηστο ως προς το σύστημα ,ο χρήστης να χρησιμοποιεί εκφράσεις οι οποίες ουσιαστικά εισάγουν το αίτημα του, αλλά δεν προσδίδουν καθόλου σημαντικότητα σε αυτό (Give me, Show me κτλ). Έτσι λοιπόν, σύμφωνα με τα παραπάνω, γίνεται επεξεργασία της πρότασης με βάση αυτό το κριτήριο.

Ενδεικτικά στην υλοποίηση της εφαρμογής που έχει πραγματοποιηθεί στο τομέα του ποδοσφαίρου κόβονται οι λέξεις που εισάγουν τη πρόταση όπως give, find, show, retrieve, get, list, know, want. Επίσης αυτές οι λέξεις προαιρετικά μπορεί να ακολουθούνται από τις αντωνυμίες me, you, him, her , us, them. Επομένως, η πρόταση '**Give me** goals scored by Ronaldo' θα γίνει 'goals scored by Ronaldo'.

Αναλυτικότερα, στη πρόταση '**Give me** goals scored by Ronaldo' για παράδειγμα, υπάρχει ένα ρήμα, το give που ακολουθείται από μια αντωνυμία me και τα οποία μαζί ουσιαστικά αποτελούν το κομμάτι της πρότασης που δεν ενημερώνει κάτι το σύστημα αλλά είναι τρόπος εισαγωγής μιας πρότασης στο φυσικό λόγο. Έτσι λοιπόν γίνεται αντιληπτό ότι τα προαναφερθέντα μπορεί να είναι απαραίτητα για το φυσικό λόγο αλλά δε προσδίδουν σημαντικότητα όσον αφορά το τρόπο αντιμετώπισης τους από το σύστημα . Επομένως, όταν ζητείται '**Give me** goals scored by Ronaldo' ουσιαστικά αυτό που πραγματικά θέλει ο χρήστης είναι το 'goals scored by Ronaldo' κάτι και το οποίο μένει μετά την αποκοπή (convertion) της πρότασης.

Σε αυτό το στάδιο, ξεχωρίζουμε 3 διαφορετικού τύπου προτάσεις, στις οποίες υπάρχει και διαφορετική αντιμετώπιση. Πιο συγκεκριμένα:

- **Αίτημα για δεδομένα** (π.χ. Show me the goals scored in the game between Italy and France)
- **WH-questions** (π.χ. What was the score in the game between Italy and France?)

  Στη φιλοσοφία, είναι γνωστό ότι τα wh-questions προσδιορίζουν το υποκείμενο του αιτήματος με βάση το τύπο της εισόδου. Έτσι λοιπόν παρουσιάζεται η σημασιολογία του αιτήματος για κάθε τύπο wh-question.

When?	Time
Where?	Place
Who?	Person
Why?	Reason
How?	Manner
What?	Object/Idea/Action
Which (one)?	Choice of alternatives
Whose?	Possession
Whom?	Person (objective formal)
How much?	Price, amount (non-count)
How many?	Quantity (count)
How long?	Duration
How often?	Frequency
How far?	Distance
What kind (of)?	Description

Εικόνα 21: Σημασιολογίες των wh questions

- **Yes/No questions** (π.χ. Were there any goals in the game between Italy and France?)

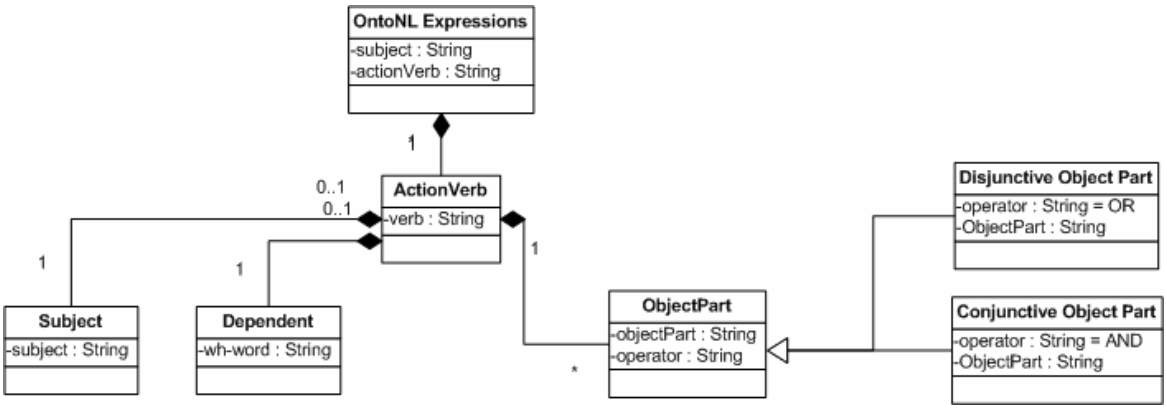
Στην είσοδο του μηχανισμού request conversion, γίνεται ανίχνευση του τύπου στον οποίο ανήκει η πρόταση και γίνεται η ανάθεση του σε ένα δείκτη. Αυτός ο δείκτης τελικά, όπως θα δειχθεί και παρακάτω, εξάγεται μέσω του **NL Query API** (εικόνα 11), έτσι ώστε να έχει τη δυνατότητα το σύστημα ανάλογα με το τύπο ερώτησης να παρέχει το κατάλληλο τύπο απάντησης, αφού αναφερόμαστε σε question – answering συστήματα. Μετά το convertion οι προτάσεις θα γίνουν:

1. the goals scored in the game between Italy and France
2. the score in the game between Italy and France
3. any goals in the game between Italy and France

Το OntoNL γλωσσικό πρότυπο (OntoNL Language Model) [1]

Το γλωσσικό πρότυπο (language model) περιγράφεται χρησιμοποιώντας ένα διάγραμμα κατηγορίας UML[49] . Η εικόνα 22 παρουσιάζει τη δομή που περιέχει τις λέξεις που αποτελούν το υποκείμενο (subject), το ρήμα δράσης (action verb) και το

αντικείμενο (object) μια έκφρασης OntoNL (OntoNL Expression). Το αντικείμενο του αιτήματος του χρήστη περιέχει τις πραγματικές πληροφορίες που ανακτώνται από την αποθήκη. Λαμβάνοντας υπόψιν και τις πληροφορίες από τις εξαρτώμενες (dependent) προτάσεις μπορεί να υπάρξει ένας μετασχηματισμός σε μια νέα πρόταση χωρίς χάσιμο της πραγματικής πληροφορίας, δεδομένου ότι υπάρχει μια συγκεκριμένη γραμματική για τις εκφράσεις, όπου το αντικείμενο θα παίξει το ρόλο του υποκειμένου



Εικόνα 22: Η συντακτική δομή μιας ανεξάρτητης πρότασης [1]

Στη συνέχεια, και αφού η πρόταση έχει επεξεργαστεί σύμφωνα με τα παραπάνω, περνάει στο επόμενο στάδιο, το Part of Speech (POS) Tagging.

5.3.2 Ανάθεση των μερών του λόγου (Part of Speech (POS) Tagging)

Στην αρχιτεκτονική του **OntoNL ,**και πιο συγκεκριμένα στο στάδιο POS Tagging, χρησιμοποιείται ο Stanford Log – Linear POS Tagger (nlp.stanford.edu/ software/tagger.shtml), ο οποίος είναι ικανός να ανιχνεύει τα μέρη του λόγου από τα οποία αποτελείται η πρόταση , δηλαδή βρίσκει ποιες λέξεις είναι ουσιαστικά, ποιες ρήματα, επίθετα, αντωνυμίες κ.τ.λ. Επίσης γίνεται κάποιου είδους ταξινόμηση και δημιουργείται μια αναπαράσταση της πρότασης σε δενδροειδή μορφή. Αυτό το στάδιο λοιπόν είναι πολύ σημαντικό, αφού απορρέουν πολύ χρήσιμα συμπεράσματα που αφορούν τη πρόταση και ποιο συγκεκριμένα τη δομή της και την αλληλουχία των λέξεων μέσα σε αυτή. Ακολουθούν παραδείγματα προτάσεων μαζί με κάποια σχόλια, όπου αυτό κρίθηκε απαραίτητο, που έχουν "περάσει" από τον Tagger έτσι ώστε να γίνει πιο εύκολη η κατανόηση του συγκεκριμένου σταδίου επεξεργασίας. Επίσης

υπενθυμίζεται στον αναγνώστη ότι στα παραδείγματα που ακολουθούν η πρόταση έχει ήδη «περάσει» από το στάδιο **Request Conversion**

Το set από ετικέτες (tagset) που χρησιμοποιεί το OntoNL για αναπαράσταση της πρότασης είναι ένα υποσύνολο του Penn Treebank Tagset και το οποίο παρουσιάζεται στο πίνακα που ακολουθεί.

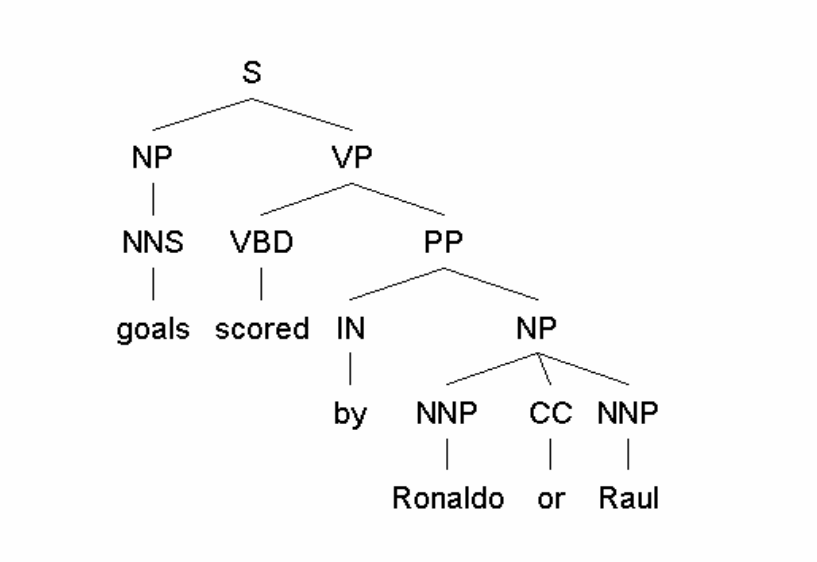
1	Coordinating conjunction	CC
2	Cardinal number	CD
3	Determiner	DT
4	Existential there	EX
5	Foreign word	FW
6	Preposition or subordinating conjunction	IN
7	Adjective	JJ
8	Adjective, comparative	JJR
9	Adjective, superlative	JJS
10	List item marker	LS
11	Modal	MD
12	Noun, singular or mass	NN
13	Noun, plural	NNS
14	Proper noun, singular	NNP
15	Proper noun, plural	NPS
16	Predeterminer	PDT
17	Possessive ending	POS
18	Personal pronoun	PP
19	Possessive pronoun	PP$
20	Adverb	RB
21	Adverb, comparative	RBR
22	Adverb, superlative	RBS
23	Particle	RP
24	Symbol	SYM
25	to	TO
26	Interjection	UH
27	Verb, base form	VB
28	Verb, phrase	VP
29	Verb, past tense	VBD
30	Verb, gerund or present participle	VBG
31	Verb, past participle	VBN
32	Verb, non-3rd person singular present	VBP
33	Verb, 3rd person singular present	VBZ
34	Wh-determiner	WDT
35	Wh-pronoun	WP
36	Possessive wh-pronoun	WP$
37	Wh-adverb	WRB
38	Wh-Noun Phrase	WHNP

Εικόνα 23: Το Penn Treebank Tagset

Επίσης, αξίζει να σημειωθεί η μεθοδολογία που χρησιμοποιείται για την αναπαράσταση των προτάσεων υπακούει στις γραμματικές Probabilistic Context Free Grammars [56]

- ΄Give me goals scored by Ronaldo or Raul’

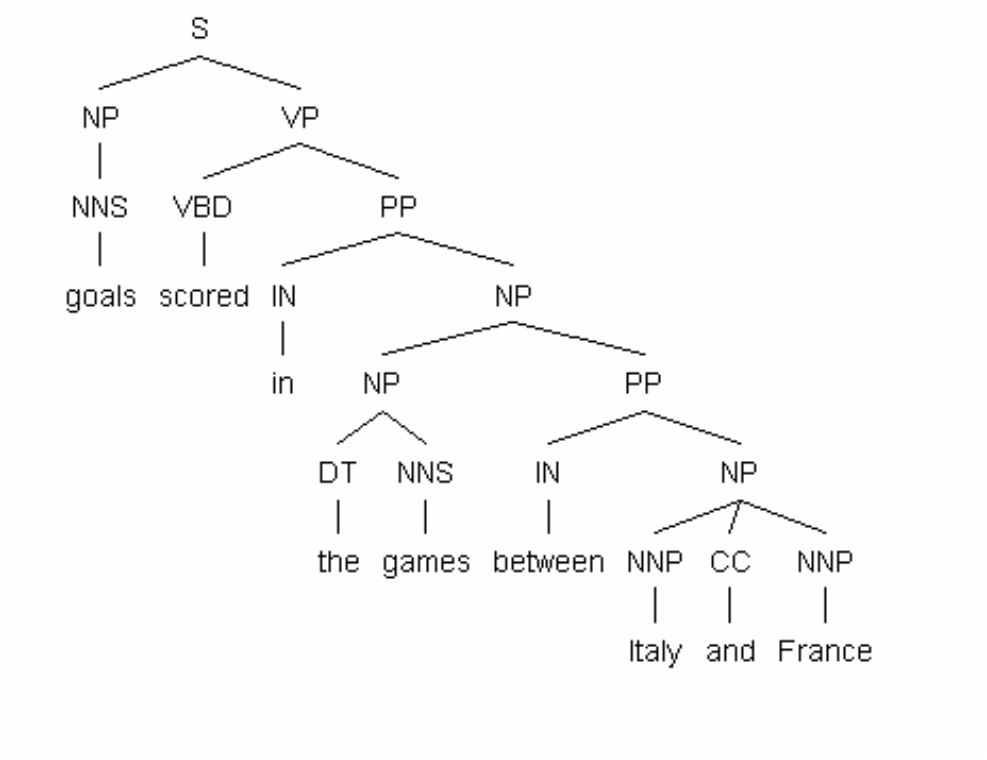
Η δενδροειδής μορφή της πρότασης που δείχνει τα μέρη του λόγου αλλά και τους συσχετισμούς μεταξύ τους είναι η εξής:



Εικόνα 24:Παράδειγμα Tagged Tree για τη πρόταση goals scored by Ronaldo

Ένα ακόμη παράδειγμα είναι το εξής:

- ‘List me goals scored in the games between Italy and France’

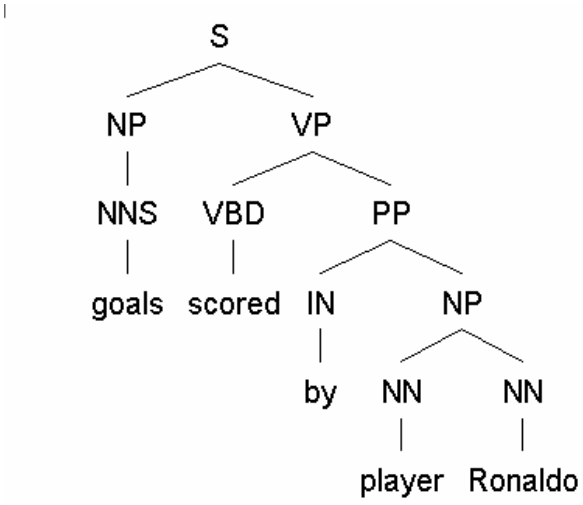


Εικόνα 25:Παράδειγμα Tagged Tree για τη πρόταση goals scored in the games between Italy and France

Πιο συγκεκριμένα, η αναπαράσταση των προτάσεων γίνεται ως εξής: Μία πρόταση στη δενδροειδή μορφή ξεκινάει με το σύμβολο S που συμβολίζει το sentence, δηλαδή ολόκληρη τη πρόταση. Στη συνέχεια η πρόταση διαιρείται σε 2 κύρια μέρη, το NP και το VP, δηλαδή το κομμάτι που περιέχει το noun phrase και το κομμάτι που περιέχει το verb phrase. Έτσι λοιπόν, είναι γνωστό πως τα υποκείμενα της πρότασης βρίσκονται στο NP κομμάτι ενώ τα αντικείμενα της πρότασης θα βρίσκονται στο VP κομμάτι αφού αυτό περιέχει το τμήμα της πρότασης στο οποίο βρίσκεται το ρήμα και ότι το ακολουθεί. Υπάρχουν φυσικά περιπτώσεις που η πρόταση δεν περιέχει ρήμα (π.χ goals of Ronaldo) όπου η πρόταση είναι ένα NP κομμάτι.

Στη συνεχεία, κάθε μια από αυτές τις μεγάλες υποενότητες NP και VP υποδιαιρούνται, με συγκεκριμένη πάντα μεθοδολογία και συμβολισμό, σε μικρότερα μέρη τα οποία με τη σειρά τους υποδιαιρούνται σε μικρότερα ώσπου να φτάσουμε στα φύλλα του δέντρου που είναι οι λέξεις που αποτελούν τη πρόταση.

Για να γίνουν κατανοητά τα παραπάνω γίνεται ανάλυση του δέντρου που προκύπτει από τη πρόταση ‘Give me goals scored by player Ronaldo’



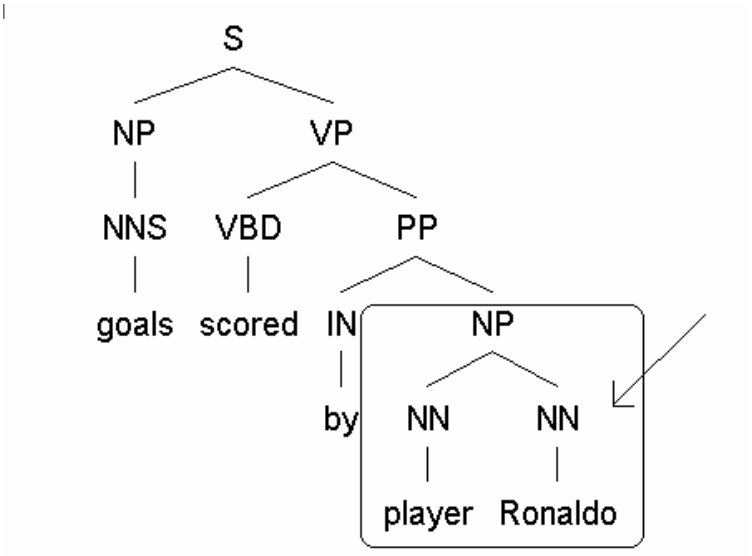
Εικόνα 26:Παράδειγμα Tagged Tree για τη πρόταση goals scored by player Ronaldo

Όπως φαίνεται στο παραπάνω δέντρο υπάρχει το S που είναι το root του δέντρου και το οποίο χωρίζεται σε δυο μέρη, το NP (noun phrase) και το VP (verb phrase). Έτσι λοιπόν, είναι γνωστό ότι στο NP βρίσκεται το υποκείμενο της πρότασης (goals) και στο VP βρίσκεται το αντικείμενο (συγκεκριμένα αφού η πρόταση είναι παθητική είναι κατηγορούμενο) της πρότασης (player Ronaldo).

Το NP έχει ως παιδί το NNS (plural noun) και αυτό με τη σειρά του έχει ως παιδί το φύλλο goals . Επομένως όσον αφορά το NP κομμάτι συμπεραίνουμε ότι περιέχει ένα ουσιαστικό, πληθυντικού αριθμού με τη τιμή goals.

Το VP έχει δυο παιδιά, το VBD (verb past tense) και το PP (preposition phrase). Το VBD έχει ως παιδί το φύλλο scored ενώ το PP έχει ως παιδιά το IN (preposition) και το NP (noun phrase). Το IN έχει ως παιδί το φύλλο by και το NP έχει 2 NN (singular noun) που όμοια έχουν ως παιδιά τα φύλλα player και Ronaldo. Έτσι λοιπόν γίνεται γνωστό ότι το VP περιέχει ένα ρήμα σε παρελθοντικό χρόνο (scored), μια πρόθεση (by) και 2 ουσιαστικά σε ενικό αριθμό (player, Ronaldo). Σε γενικές γραμμές αυτή είναι η μεθοδολογία που ακολουθείται κατά την ανάλυση της πρότασης.

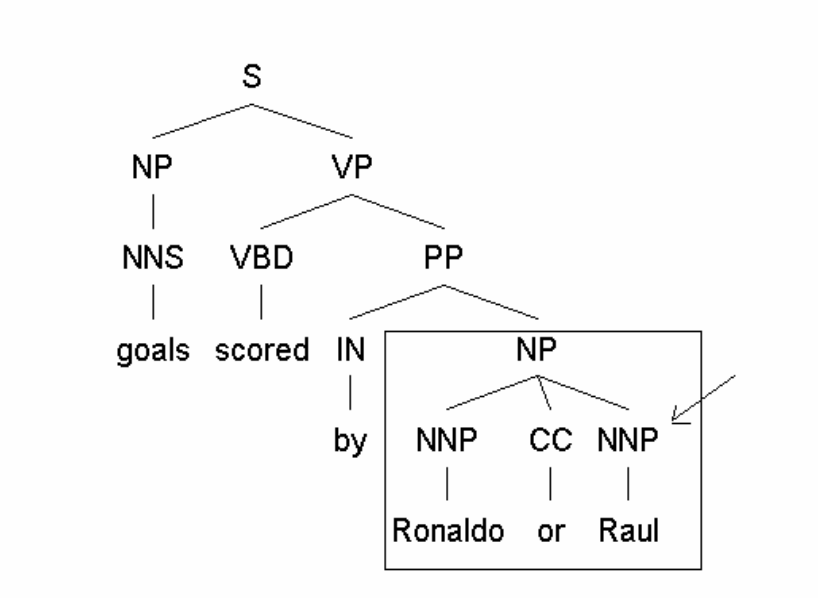
Κάτι που επίσης αξίζει να σημειωθεί, είναι ότι με τη συγκεκριμένη μεθοδολογία μπορούν να βγουν συμπεράσματα για το ποιες λέξεις «απευθύνονται» σε ποιες έτσι ώστε να γίνεται ευκολότερα η μετάβαση στο μελλοντικό στάδιο της αποσαφήνισης. Δηλαδή στη πρόταση που μόλις αναλύθηκε (Give me goals scored by player Ronaldo) παρατηρείται ότι οι λέξεις player, Ronaldo βρίσκονται κάτω από το ίδιο NP (noun phrase)



Εικόνα 27:Παράδειγμα Tagged Tree για τη πρόταση goals scored by player Ronaldo στο οποίο φαίνεται ο συσχετισμός των λέξεων player, Ronaldo

επομένως είναι φανερό ότι το player απευθύνεται στο Ronaldo, δηλαδή όπως θα δειχθεί παρακάτω, φανερώνεται με αυτό το τρόπο ότι το Ronaldo θα είναι value στη κλάση Player της οντολογίας, γεγονός που κάνει το στάδιο αποσαφήνισης που ακολουθεί πολύ πιο εύκολο και συγκεκριμένο.

Τέλος, αυτό που επίσης πρέπει να αναφερθεί, είναι η ειδική περίπτωση που στη πρόταση εμφανίζονται operators (με τον όρο operators εννοούνται οι διαζευκτικοί όροι που μπορεί να παρουσιαστούν σε μία πρόταση, οι οποίοι είναι οι **ή(or)**, **και (and)**). Για τη πρόταση 'Give me goals scored by Ronaldo or Raul' δημιουργείται το παρακάτω δένδρο



Εικόνα 28:Παράδειγμα Tagged Tree για τη πρόταση goals scored by Ronaldo or Raul στο οποίο φαίνεται η διάσπαση της πρότασης όταν υπάρχει λογικός τελεστής

στο οποίο συναντώνται όλοι οι συμβολισμοί που έχουν εμφανιστεί και στα παραπάνω παραδείγματα. με τη διαφορά ότι υπάρχει και ο συμβολισμός CC (operator) και ο οποίος χρησιμοποιείται όταν υπάρχουν λέξεις που είναι operators (OR, AND). Ο τρόπος με τον οποίο χρησιμοποιείται ο συμβολισμός αυτός είναι τέτοιος έτσι ώστε από το NP στο οποίο ανήκει να γίνεται αντιληπτό ποιες τιμές βρίσκονται ανάμεσα από τον operator. Έτσι λοιπόν παρατηρείται ότι στο ίδιο NP κομμάτι που βρίσκεται ο operator OR βρίσκονται και τα ουσιαστικά Ronaldo και Raul, επομένως είναι φανερό ότι ο operator "απευθύνεται" σε αυτές τις 2 λέξεις . Αποτέλεσμα όλων αυτών είναι ότι σε μελλοντικό στάδιο ανάλυσης θα είναι γνωστό ότι ζητούνται από το σύστημα τα goals scored by Ronaldo OR goals scored by Raul με αποτέλεσμα τη καλύτερη επεξεργασία της πρότασης και πιο σωστά αποτελέσματα

5.3.3 Εύρεση συνωνύμων και εννοιών (Synonyms and Sense Discovery)

Σε αυτό το στάδιο, όπως μπορεί να καταλάβει κανείς και από το τίτλο του, γίνεται προσπάθεια να ανακαλυφθούν τα συνώνυμα και οι έννοιες των λέξεων. Για να το επιτευχθεί ο παραπάνω σκοπός χρησιμοποιείται ο θησαυρός λέξεων και συνωνύμων

WordnetData, ο οποίος προσπελαύνεται μέσω του JWordent API (κεφάλαιο 3) . Αποθηκεύονται σε συγκεκριμένες δομές όλα τα συνώνυμα και οι έννοιες της κάθε λέξης που εμφανίζεται στη πρόταση. Ο λόγος για τον οποίο υπάρχει αυτό το στάδιο είναι πολύ συγκεκριμένος. Υπάρχει πάντα η περίπτωση ο χρήστης στο αίτημα που θα απευθύνει στη εφαρμογή να χρησιμοποιεί λέξεις οι οποίες όμως να μην μπορούν να αντιστοιχηθούν στην οντολογία όχι γιατί δεν υπάρχουν εννοιολογικά σε αυτή αλλά γιατί ενδεχομένως στην οντολογία να χρησιμοποιείται άλλος όρος που όμως έχει την ίδια σημασία. Κάτι που είναι απολύτως φυσιολογικό αφού ο χρήστης δεν είναι υποχρεωμένος να γνωρίζει την ονοματολογία που ίσως να χρησιμοποιείται στην εκάστοτε οντολογία. Αποτέλεσμα αυτών φυσικά είναι να μην υπάρξουν σωστά αποτελέσματα.

Για την επίλυση αυτού του προβλήματος και για να μειωθεί η πιθανότητα να υπάρχει λάθος αντιμετώπιση της πρότασης, εξετάζονται από το σύστημα επίσης τα συνώνυμα των λέξεων που χρησιμοποιεί ο χρήστης και συγκεκριμένα εξετάζεται αν τα συνώνυμα αυτά αντιστοιχίζονται στην οντολογία. Έτσι λοιπόν αυξάνεται η πιθανότητα να αποσαφηνιστεί σωστότερα το αίτημα του χρήστη, αφού με αυτό το τρόπο αυξάνονται οι πιθανές ερμηνείες των λέξεων που αυτός χρησιμοποιεί στο αίτημα του.

Ένα παράδειγμα για να γίνουν κατανοητά τα όσα έχουν προαναφερθεί είναι ο χρήστης να χρησιμοποιήσει στο αίτημα του τη λέξη **finishes**. Για παράδειγμα ας υποθέσουμε τη πρόταση ‘Give me **finishes** of Ronaldo’ και επίσης ας υποθέσουμε ότι υπάρχει μία οντολογία που περιέχει ως κλάση τη κλάση **Goal**. Η λέξη **finish** δε θα μπορέσει να αντιστοιχηθεί με κάποιο concept της οντολογίας αφού δεν υπάρχει κλάση με τέτοιο όνομα. Όμως σε αυτό το στάδιο που αποθηκεύονται τα συνώνυμα των λέξεων παρατηρείται ότι η λέξη **finish** έχει ως συνώνυμο τη λέξη **goal**. Έτσι λοιπόν, αφού εξετάζονται και τα συνώνυμα, μπορεί να συσχετιστεί η λέξη **finish** που χρησιμοποίησε ο χρήστης στο φυσικό λόγο με τη κλάση **Goal** της οντολογίας. Στη προσπάθεια λοιπόν του συστήματος να φέρει «πιο κοντά» την οντολογία στο χρήστη, μια από τις μεθόδους που χρησιμοποιείται είναι και η προαναφερθείσα αφού με αυτό το τρόπο το σύστημα προσπαθεί να «μαντέψει» τι θα ήθελε να πει ο χρήστης και να το μεταφέρει στην οντολογία.

Φυσικά υπάρχουν περιπτώσεις που κάποια από τις λέξεις που χρησιμοποιούνται δεν έχει συνώνυμα ή και να έχει συνώνυμα αυτά δεν αντιστοιχίζονται με όρους της οντολογίας. Αυτό δε λειτουργεί αρνητικά στην όλη διαδικασία. Απλά στη περίπτωση

που υπάρχουν συνώνυμα και δεν έχει αντιστοιχηθεί η συγκεκριμένη λέξη με κάποιον όρο της οντολογίας περιοχής (domain ontology), τα συνώνυμα βοηθάνε ώστε να υπάρχει ένας επιπλέον έλεγχος στο μελλοντικό στάδιο της αποσαφήνισης.

Συνοψίζοντας, σε αυτό το στάδιο δημιουργούνται δομές στις οποίες αποθηκεύονται τα συνώνυμα των λέξεων που εμφανίζονται στη πρόταση. Κάτι που θα φανεί πολύ χρήσιμο σε μελλοντικά στάδια επεξεργασίας.

```
Program List getSynonymsAndSenses (String word)
    List listOfWordsReturned;
  Begin
    String[] senses=getSenses(word);
    If(senses.size()>0){
      For all terms i of senses{
         String temp=senses[i];
         String[] synonyms=getSynonyms[temp];
         For all terms j of synonyms{
           String temp2=synonyms[j];
           listOfWordsReturned.add(temp2);
         }
       }
     }
    Else
      listOfWordsReturned.add("Not specified");
    return listOfWordsReturned;
  End
```

Ένα παράδειγμα αυτού του σταδίου παρουσιάζεται με τη πρόταση '*Give me* goals of Ronaldo'. Για αυτή τη πρόταση υπάρχουν τα εξής συνώνυμα:

Goal:

End
Destination
Aim
Object
Objective
Target
Bourn
Intent
Basket
Hoop
Net
Terminus
Score

Ronaldo: *Not Specified*

Ερμηνεύοντας λοιπόν το παραπάνω παράδειγμα, παρατηρείται ότι αφού το *Give me* "κόβεται" στο στάδιο του **Request Conversion,** τα ουσιαστικά που απομένουν στη πρόταση είναι τα goals, Ronaldo και επομένως για αυτά αναζητούνται τα συνώνυμα. Η λέξη Goals έχει δεκατρία συνώνυμα ενώ η λέξη Ronaldo δεν έχει κανένα. Ποια από αυτά τα συνώνυμα θα αξιοποιηθούν, ίσως και κανένα, θα φανεί στο μελλοντικό στάδιο της αποσαφήνισης.

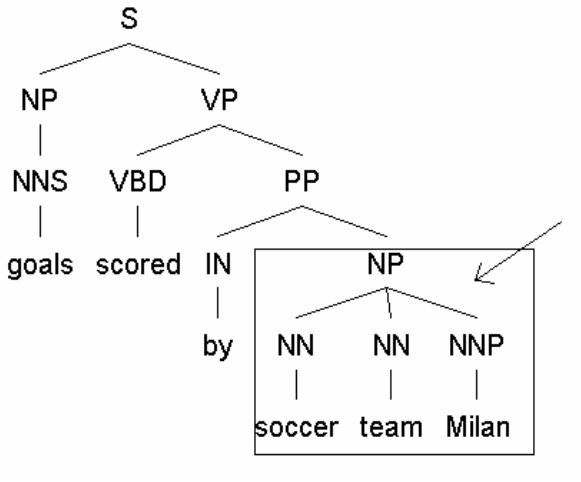
5.3.4 Αναγνώριση και κατηγοριοποίηση των ενώσεων ουσιαστικού (Noun Compound Identification & Noun Compound Bracketing)

Σε αυτά τα δύο στάδια επεξεργασίας, δημιουργούνται κάποιες δομές στις οποίες αποθηκεύονται τα ουσιαστικά εκείνα της πρότασης τα οποία έχουν άμεση συσχέτιση μεταξύ τους. Στο τρέχων κεφάλαιο και συγκεκριμένα στην ενότητα 5.2 αναλύθηκε η λειτουργικότητα των **Tokenization** και **Abbreviation Expansion**. Αναφέρθηκε ότι αποτέλεσμα αυτών των σταδίων είναι η δημιουργία ενός αρχείου στο οποίο έχουν αποθηκευτεί για κάθε όνομα κλάσης της εκάστοτε οντολογίας κάποιοι πιθανοί εναλλακτικοί τρόποι που θα μπορούσε ο χρήστης να αναφερθεί σε αυτές. Υπενθυμίζεται στον αναγνώστη, ότι για μια υποθετική κλάση με το όνομα soccTeam, προτείνονται ως εναλλακτική ονοματολογία οι λέξεις Soccer Team. Αυτό που μένει τώρα από πλευράς συστήματος - φυσικού λόγου, είναι να εντοπιστούν στο αίτημα του χρήστη τα ουσιαστικά εκείνα αποτελούν μαζί μια ενιαία έννοια, να αποθηκευτούν σε μία δομή και στη συνέχεια σε κάποιο από τα επόμενα στάδια επεξεργασίας στο οποίο θα γίνεται αντιστοίχηση του φυσικού λόγου με όρους της οντολογίας να αξιοποιηθούν και αυτά.

Έτσι λοιπόν, στα στάδια που εξετάζονται (Noun Compound Identification & Noun Compound Bracketing), αποθηκεύονται σε δομές ομαδοποιημένα ουσιαστικά. Πρόβλημα αποτελεί το πως γίνεται φανερό ποιες από τις λέξεις που εμφανίζονται στη πρόταση συνδέονται με τέτοιο τρόπο ώστε να φανερώνεται ότι συσχετίζονται άμεσα μεταξύ τους, δηλαδή ότι έχουν μία τέτοιου είδους εξάρτηση που η μία λέξη απευθύνεται στην άλλη με τρόπο που όλες μαζί να μας δίνουν ένα ενιαίο νόημα. Η μεθοδολογία που ακολουθείται από το σύστημα, είναι να εξετάζεται το δέντρο που προκύπτει όταν γίνεται parse η πρόταση.

Έχει ήδη αναλυθεί στην υποενότητα **Part of Speech (POS) Tagging** ότι στο δέντρο που προκύπτει ανιχνεύονται ποιες λέξεις βρίσκονται κάτω από το ίδιο NP έτσι ώστε να γίνεται φανερό ποιες συσχετίζονται άμεσα μεταξύ τους. Για παράδειγμα ,

στη πρόταση 'Give me goals scored by soccer team Milan' το δένδρο που θα προκύψει όταν γίνει parse η συγκεκριμένη πρόταση στο Tagger είναι το εξής:



Εικόνα 29:Παράδειγμα Tagged Tree για τη πρόταση goals scored by soccer team Milan στο οποίο φαίνονται τα ουσιαστικά που συνδέονται άμεσα μεταξύ τους και γίνεται έλεγχος για το αν είναι noun compound

Παρατηρείται ότι κάτω από το ίδιο NP βρίσκονται 3 λέξεις, οι soccer, team και Milan. Αυτές οι λέξεις συσχετίζονται άμεσα μεταξύ τους και θα πρέπει να ληφθούν υπόψιν στο στάδιο Noun Compound Identification & Noun Compound Bracketing. Φυσικά λαμβάνονται όλοι οι πιθανοί συνδυασμοί (και ανά δύο και ανά τρεις) που μπορούν να υπάρξουν μεταξύ αυτών των τριών λέξεων ως ομαδοποιημένα σύνολα ουσιαστικών.

Έτσι λοιπόν , ένας από αυτούς τους συνδυασμούς θα είναι και ο Soccer Team κάτι που θα φανεί πολύ χρήσιμο στα επόμενα στάδια αποσαφήνισης, αφού αυτός ο όρος θα αντιστοιχηθεί μέσω του αρχείου που ήδη έχει δημιουργηθεί στο στάδιο **Ontology Processor** και αφορά την ονοματολογία της οντολογίας στη κλάση SoccerTeam της οντολογίας. Αξίζει να σημειωθεί ότι αυτό το στάδιο λειτουργεί βοηθητικά στη προσπάθεια του συστήματος να φέρει την οντολογία πιο κοντά στο χρήστη. Φυσικά θα υπάρχουν πολλές περιπτώσεις που ο συνδυασμός των λέξεων που αποθηκεύεται στις δομές δε θα αντιστοιχίζεται σε κανέναν όρο της οντολογίας, κάτι που όμως δεν επιβαρύνει το σύστημα. Είναι μια επεξεργασία που γίνεται πάντα χωρίς να υπάρχει ο κίνδυνος ότι θα προκύψουν λανθασμένα συμπεράσματα αφού λειτουργεί συναινετικά.

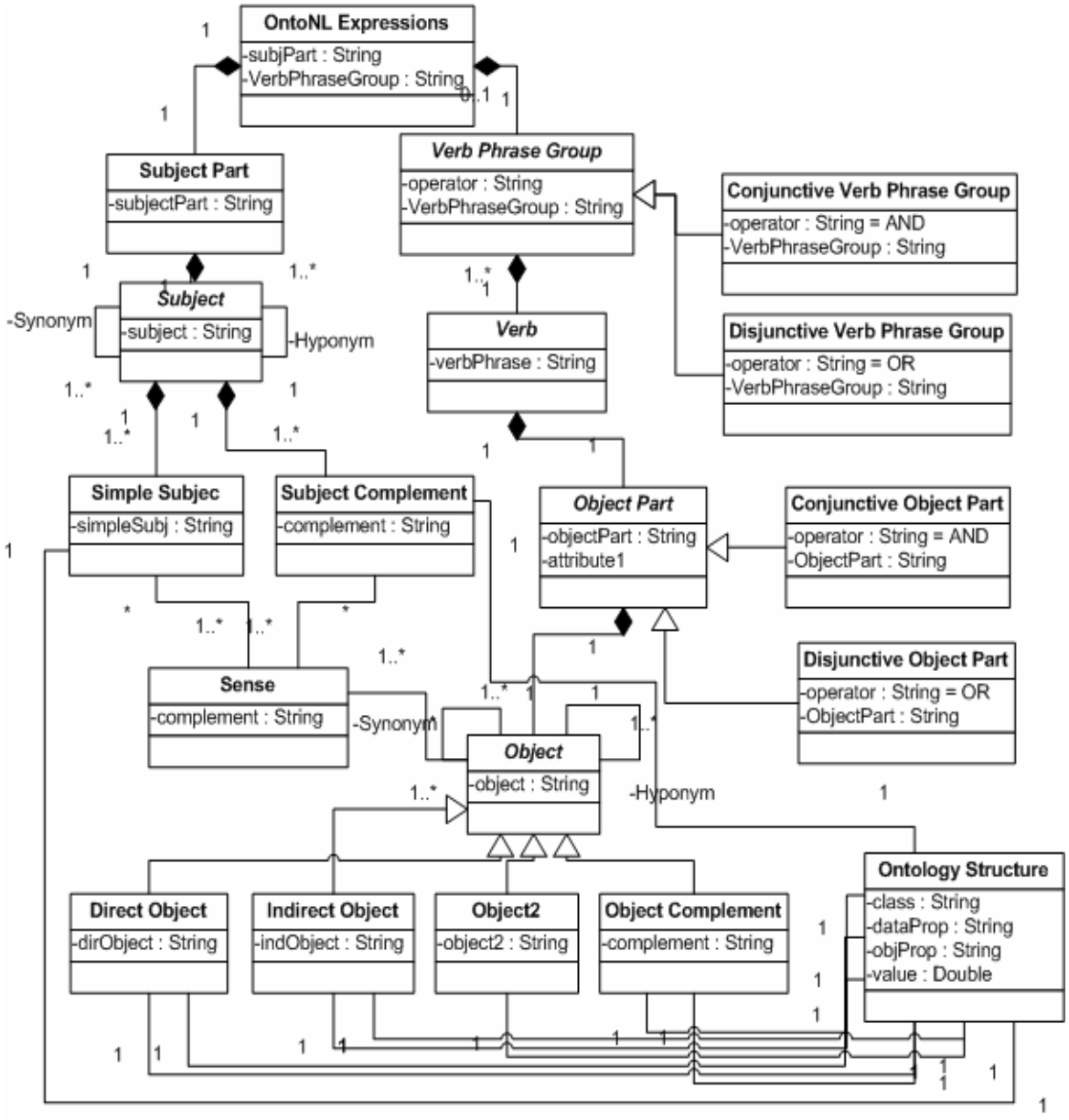
5.3.5 Σχολιασμός γραμματικών σχέσεων (Grammatical Relations Annotation)

Σε αυτό το στάδιο, όπως δηλώνει και ο τίτλος του, γίνεται η προσπάθεια να ανιχνευτούνε οι γραμματικές σχέσεις που συνδέουν τα μέρη που αποτελούν τη πρόταση. Η ανακάλυψη των γραμματικών σχέσεων μέσα σε μία πρόταση θα συνδράμει στη δημιουργία ενός κατάλληλου γλωσσικού μοντέλου που θα βοηθήσει το σύστημα να ανακτήσει τα σωστά δεδομένα από το repository. Με άλλα λόγια, αποτελεί τη σημασιολογική βάση από την οποία θα γίνει εξαγωγή πληροφορίας.

Το OntoNL γλωσσικό πρότυπο (OntoNL Language Model) [1]

Σύμφωνα πάντα με την αρχιτεκτονική του **OntoNL**[1] πλαισίου, δημιουργείται μια δομή χρησιμοποιώντας ένα πρότυπο διάγραμμα που φαίνεται στην εικόνα 30 με τις πληροφορίες που προέρχονται από όλα τα βήματα κατά τη διάρκεια της γλωσσικής αποσαφήνισης

Σε αυτό το πρότυπο διάγραμμα υπάρχουν κατηγορίες που αντιπροσωπεύουν τις γραμματικές σχέσεις που συνδέονται με σχέσεις. Παρατηρούμε ότι υπάρχουν δομές λέξεων που αποτελούν τις βασικές δομές της πρότασης, όπως το υποκείμενο (subject) και το αντικείμενο (object) και υπάρχουν συμπληρώματα (complements) και ειδικές περιπτώσεις των αντικειμένων που τα συνοδεύουν.



Εικόνα 30: Το OntoNL Language Model που εξάγεται από τη γλωσσολογική ανάλυση [1]

Οι εκφράσεις OntoNL είναι η γενική κατηγορία που συνοψίζει τις περιπτώσεις των πιθανών γραμματικών εξαρτήσεων μέσα σε μια έκφραση. Αποτελείται από ένα Subject Part και ενδεχομένως από ένα Verb Phrase Group. Το Subject Part αποτελείται από ένα ή περισσότερα Subject που συνδέονται μεταξύ τους με λογικούς τελεστές. Το Verb Phrase Group είναι μια αφηρημένη (abstract) κατηγορία και έχει μια IS-A σχέση με τα συνδετικά (Conjunctive) και τα διαχωριστικά (Disjunctive) Verb Phrase Group.

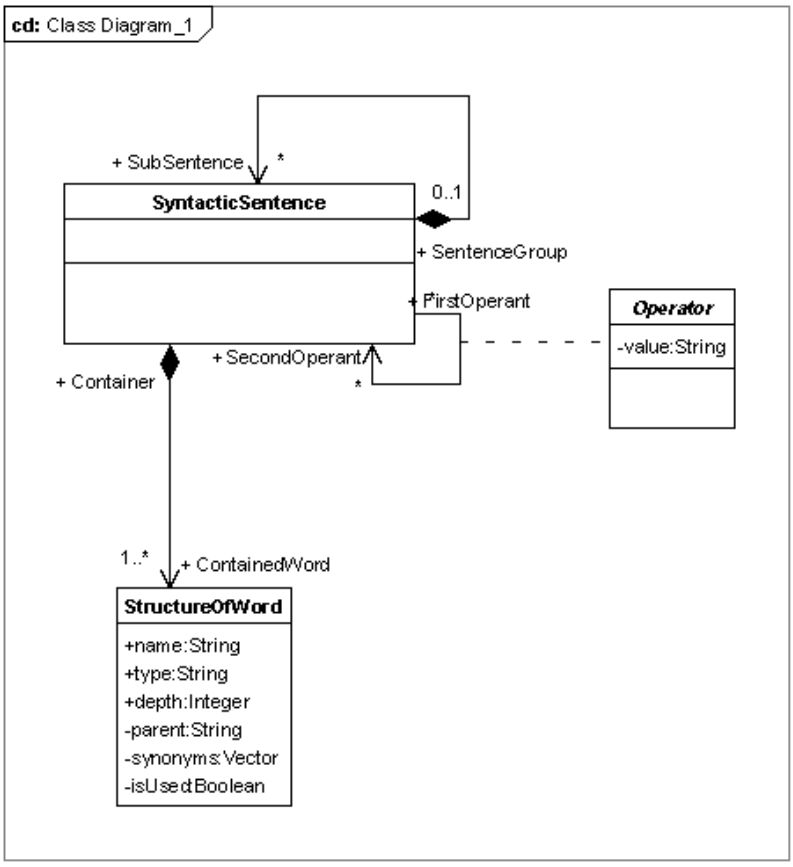
Το Verb Phrase Group αποτελείται από ένα ή περισσότερα ρήματα. Το ρήμα αποτελείται από ένα μοναδικό Object Part, μια αφηρημένη κλάση ενός ή

περισσότερων αντικειμένων. Επίσης, το Object Part έχει μια IS-A σχέση με τα συνδετικά (Conjunctive) και τα διαχωριστικά (Disjunctive) Object Part.

Αυτός σημαίνει ότι μετά από το ρήμα μπορούμε να συναντήσουμε περισσότερα από ένα Object Part συνδεδεμένα μεταξύ τους με τελεστές σύζευξης ή διάζευξης. Το αντικείμενο μπορεί να είναι ένα άμεσο αντικείμενο (Direct Object), ένα έμμεσο αντικείμενο (Indirect Object), ένα συμπλήρωμα αντικειμένου (Object Complement) ή ένας συνδυασμός αυτών. Αξίζει να σημειωθεί ότι για κάθε μία δομή που έχει περιγραφεί, αποθηκεύονται τα συνώνυμα των λέξεων που αυτή περιέχει στη κλάση Sense.

Τέλος υπάρχει ένα Ontology Structure που αφορά το στάδιο της αποσαφήνισης των εννοιών και θα αναλυθεί παρακάτω.

Έτσι λοιπόν, έχοντας αναλυθεί η δομή-λειτουργικότητα του **OntoNL Language Model** [1], σε αυτό το στάδιο, και σε στάδιο υλοποίησης πλέον, δημιουργείται μια δομή (εικόνα 31) , στην οποία ουσιαστικά γίνεται η αναπαράσταση της συντακτικής ανάλυσης της πρότασης και του τρόπου με τον οποίο οι λέξεις συσχετίζονται μεταξύ τους. Αυτή η δομή στη συνέχεια, ανάλογα με τη πληροφορία που περιέχει θα εμπλουτίσει αντίστοιχα το OntoNL Language Model.



Εικόνα 31:Η δομή SyntacticSentence

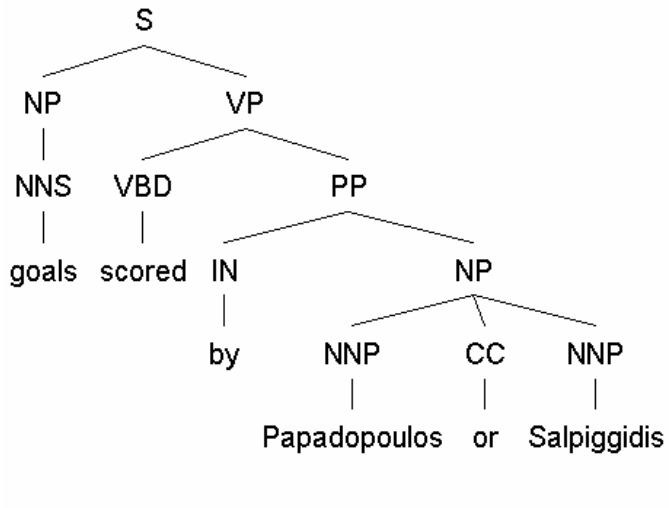
Το SyntacticSentence (συντακτική πρόταση) είναι μια εσωτερική δομή η οποία δημιουργείται και χρησιμοποιείται κατά το στάδιο της γλωσσολογικής ανάλυσης (Linguistic Analysis) για να αναπαραστήσει της γραμματικές σχέσεις ανάμεσα στις λέξεις που αποτελούν το αίτημα του χρήστη.

Όπως φαίνεται και από το παραπάνω class diagram ένα SyntacticSentence μπορεί να εμπεριέχει ένα ή περισσότερα SyntacticSentence (SentenceGroup). Επίσης ένα SyntacticSentence συνδέεται με ένα άλλο SyntacticSentence μέσω operator (υπάρχει η κλάση Operator που έχει ως attribute το String value που θα είναι OR ή AND).

Τέλος, κάθε SyntacticSentence περιέχει ένα ή περισσότερα StructureOfWord που , όπως δηλώνει η ονομασία τους, αντιπροσωπεύουν ολόκληρη τη δομή μιας λέξης που βρίσκεται στο φυσικό λόγο. Δηλαδή περιέχει την ίδια τη λέξη (name), το τύπο της (type) που δηλώνει αν αυτή η λέξη είναι subject, subject complement, object ή object complement, σε ποιο βάθος βρέθηκε μέσα στο δέντρο (depth), ποιος είναι ο πατέρας της λέξης (parent) που προκύπτει από το συντακτικό δέντρο και δηλώνει αν η λέξη είναι ουσιαστικό, ρήμα, επίθετο κτλ, ένα Vector με τα συνώνυμα της λέξης (αν υπάρχουν) και τέλος μια βοηθητική Boolean μεταβλητή isUsed που βοηθάει να γίνει αντιληπτό αν το συγκεκριμένο StructureOfWord έχει λάβει μέρος σε διάφορους ελέγχους που λαμβάνουν μέρος στο συγκεκριμένο στάδιο.

Αποτέλεσμα όλων αυτών είναι η ανάπτυξη μιας μεθοδολογίας στην οποία έχουν υλοποιηθεί τεχνικές που εξετάζοντας το tagged tree που προκύπτει από το στάδιο **Part of Speech (POS) Tagging** να μπορούν να εμπλουτίσουν τη δομή που μόλις αναλύθηκε.

Παράδειγμα μίας τέτοιας ανάλυσης είναι το εξής (List me goals scored by Papadopoylos or Salpiggidis):



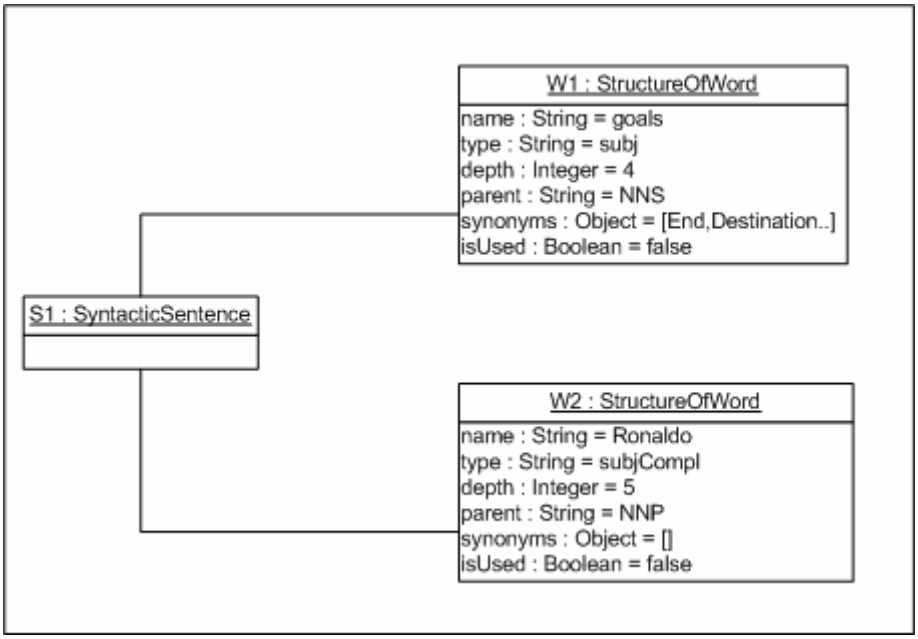
Εικόνα 32:Παράδειγμα Tagged Tree για τη πρόταση goals scored by Papadopoylos or Salpiggidis

Από αυτή και μόνο την αναπαράσταση, όπως έχει αναφερθεί και σε προηγούμενη υποενότητα, μπορούν να εξαχθούν κάποια συμπεράσματα. Έτσι λοιπόν έχουν υλοποιηθεί διάφορες αναδρομικές συναρτήσεις συμφωνά με τις οποίες για παράδειγμα η αναζήτηση υποκειμένου της πρότασης γίνεται στο NP κομμάτι του παραπάνω δένδρου (goals). Όμοια για τα αντικείμενα της πρότασης (Papadopoylos , Salpiggidis) γίνεται το ίδιο για το VP κομμάτι της πρότασης. Επίσης αν προηγείται του υποκειμένου ή του αντικειμένου αντίστοιχα κάποια πρόθεση τότε είναι γνωστό ότι γίνεται αναφορά σε κάποιο subject complement ή object complement αντίστοιχα.

Επίσης, εκτός από τη συντακτική ανάλυση της πρότασης, έχουν αναπτυχθεί μέθοδοι οι οποίες όταν εμφανίζεται operator στη πρόταση αναλύουν σε υποπροτάσεις τη κύρια πρόταση έτσι ώστε να είναι πιο αναλυτικό και συγκεκριμένο αυτό που θα παραχθεί από μεριάς φυσικού λόγου στο στάδιο της εννοιολογικής αποσαφήνισης. Δηλαδή στο παραπάνω παράδειγμα θα δημιουργούνταν οι υποπροτάσεις ‘goals scored by Papadopoylos’ και ‘goals scored by Salpiggidis’ που θα ήταν δύο διαφορετικά SyntacticSentence που θα άνηκαν στο ίδιο SentenceGroup και θα συνδέονταν μεταξύ τους με τον operator OR αφού αυτός χρησιμοποιείται στη πρόταση .

Αφού λοιπόν έχει παρουσιαστεί - αναλυθεί η παραπάνω δομή SyntacticSentence που δημιουργείται στο συγκεκριμένο στάδιο, και είναι η τελευταία του **Linguistic Analyzer** αφού μετά γίνεται μετάβαση στο στάδιο **Semantic Disambiguator** που θα εξηγηθεί παρακάτω, παραθέτονται κάποια παραδείγματα δομών με σύντομο σχολιασμό που έχουν δημιουργηθεί από κάποιες ενδεικτικές προτάσεις και υποδεικνύουν τη λειτουργικότητα της δομής.

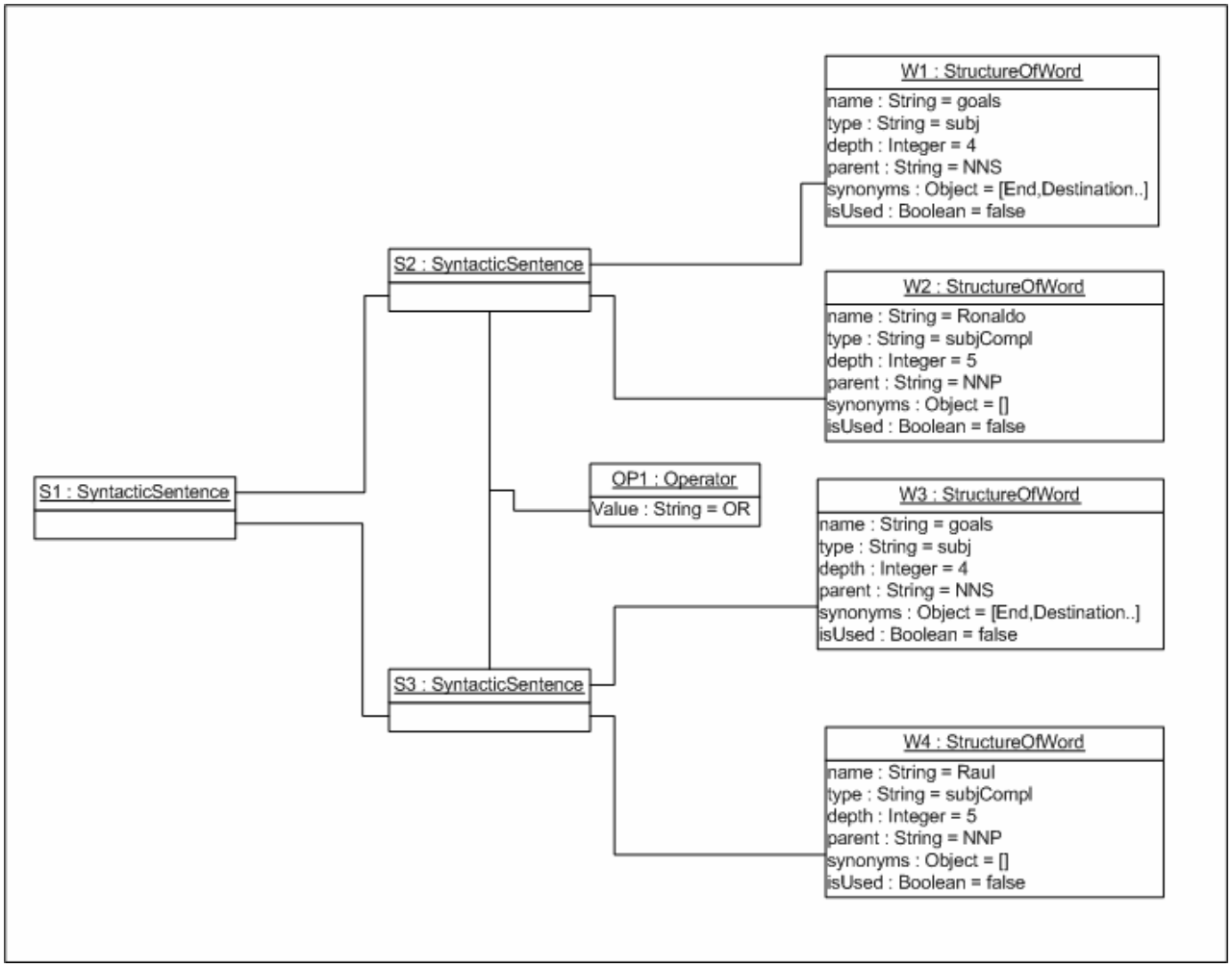
- 'Give me the goals of Ronaldo'



Εικόνα 33:Παράδειγμα SyntacticSentence για τη πρόταση goals of Ronaldo

Εδώ παρουσιάζεται μια απλή πρόταση στην οποία δεν εμφανίζεται operator. Υπάρχουν δύο ουσιαστικά τα οποία ανήκουν στο subject part της πρότασης αφού έτσι κι αλλιώς δεν υπάρχει ρήμα άρα δεν υπάρχει και object part. Έτσι λοιπόν δημιουργείται ένα SyntacticSentence με δύο StructureOfWord, ένα για κάθε ουσιαστικό. Το πεδία depth, parent και type συμπληρώνονται σύμφωνα με το tagged tree που έχει προκύψει από το στάδιο **Part Of Speech (POS) Tagging** ενώ η δομή synonyms συμπληρώνεται από τα αποτελέσματα του σταδίου **Synonyms and Sense Discovery**.

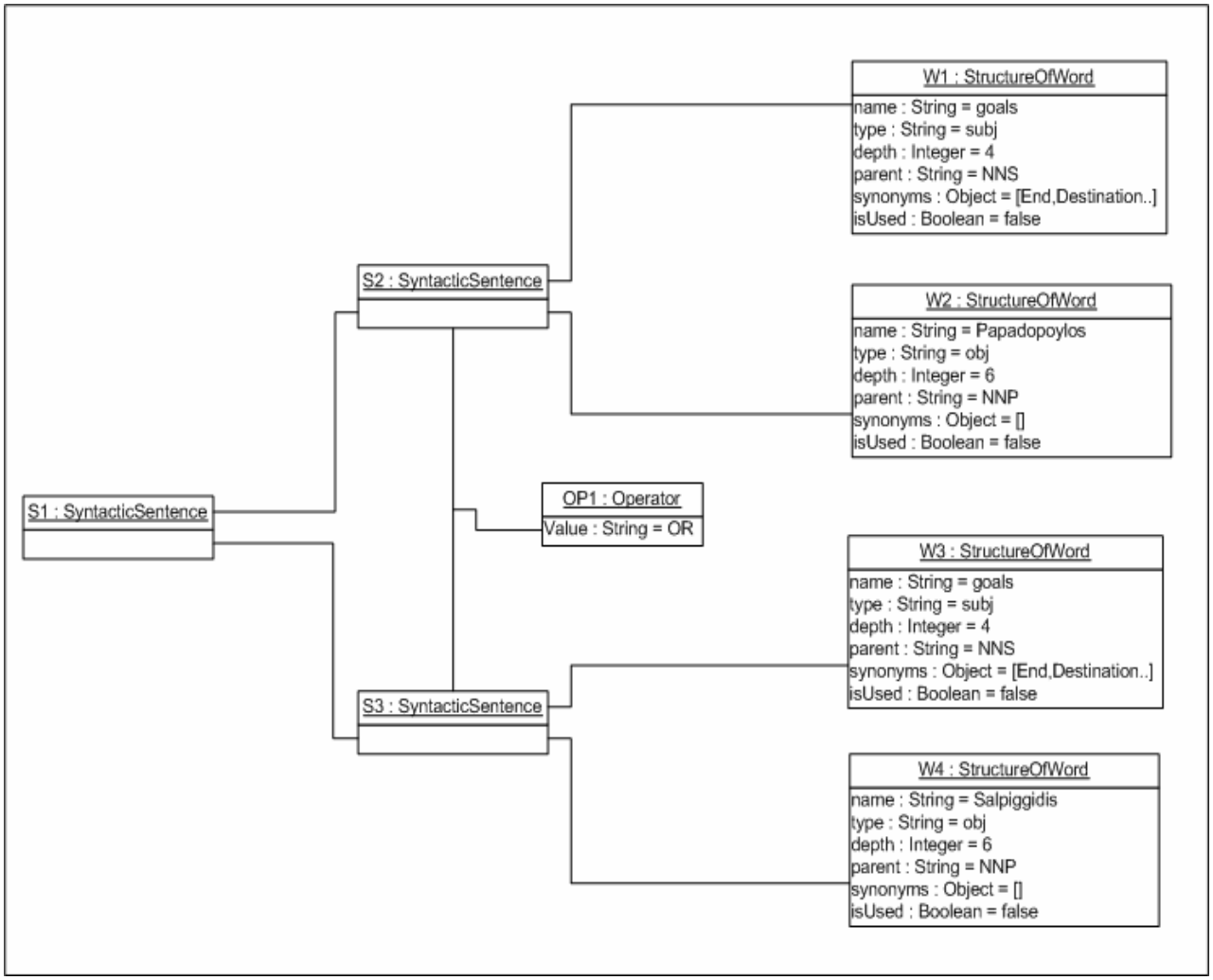
- List me goals of Ronaldo or Raul



Εικόνα 34:Παράδειγμα SyntacticSentence για τη πρόταση goals of Ronaldo or Raul

Στο παραπάνω παράδειγμα παρουσιάζεται μία πρόταση στην οποία εμφανίζεται operator (OR). Έτσι λοιπόν , όπως έχει ήδη αναφερθεί, γίνεται ανάλυση της κύριας πρότασης σε υποπροτάσεις και για κάθε μία από αυτές δημιουργείται ένα SyntacticSentence. Σε κάθε SyntacticSentence τοποθετούνται τα κατάλληλα StructureOfWord με τις λέξεις που έχουν προκύψει από την ανάλυση και συμπληρώνονται τα πεδία τους κατάλληλα.

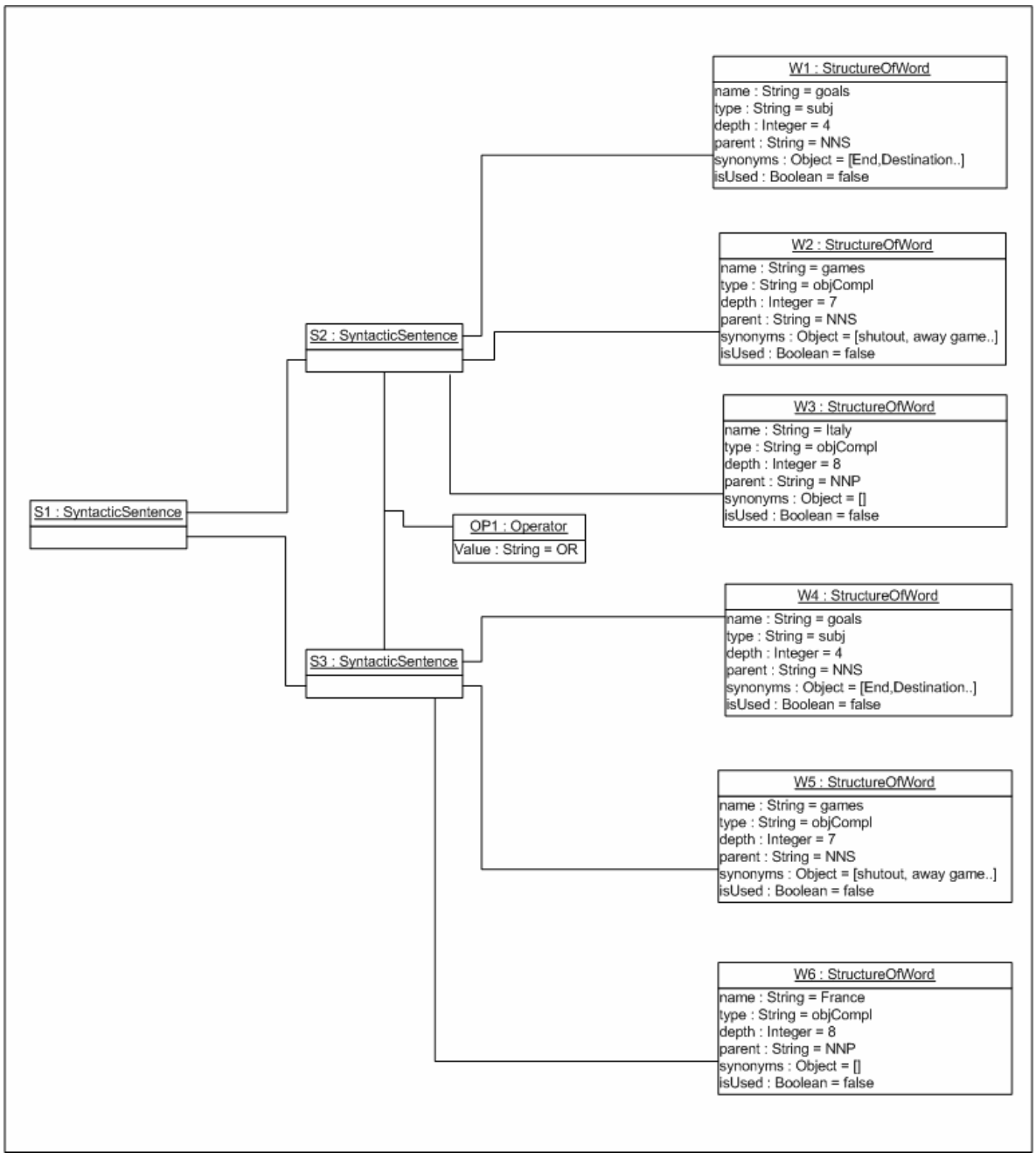
- List me goals scored by Papadopoylos or Salpiggidis



Εικόνα 35:Παράδειγμα SyntacticSentence για τη πρόταση goals scored by Papadopoylos or Salpiggidis

Σε αυτό το παράδειγμα εμφανίζεται ρήμα στη πρόταση. Επομένως η ανάλυση θα γίνει όπως και πριν με τη διαφορά ότι εδώ εκτός από υποκείμενο (goals) υπάρχουν και αντικείμενα (Salpiggidis, Papadopoylos) κάτι που φαίνεται φυσικά στα StructureOfWord που δημιουργούνται.

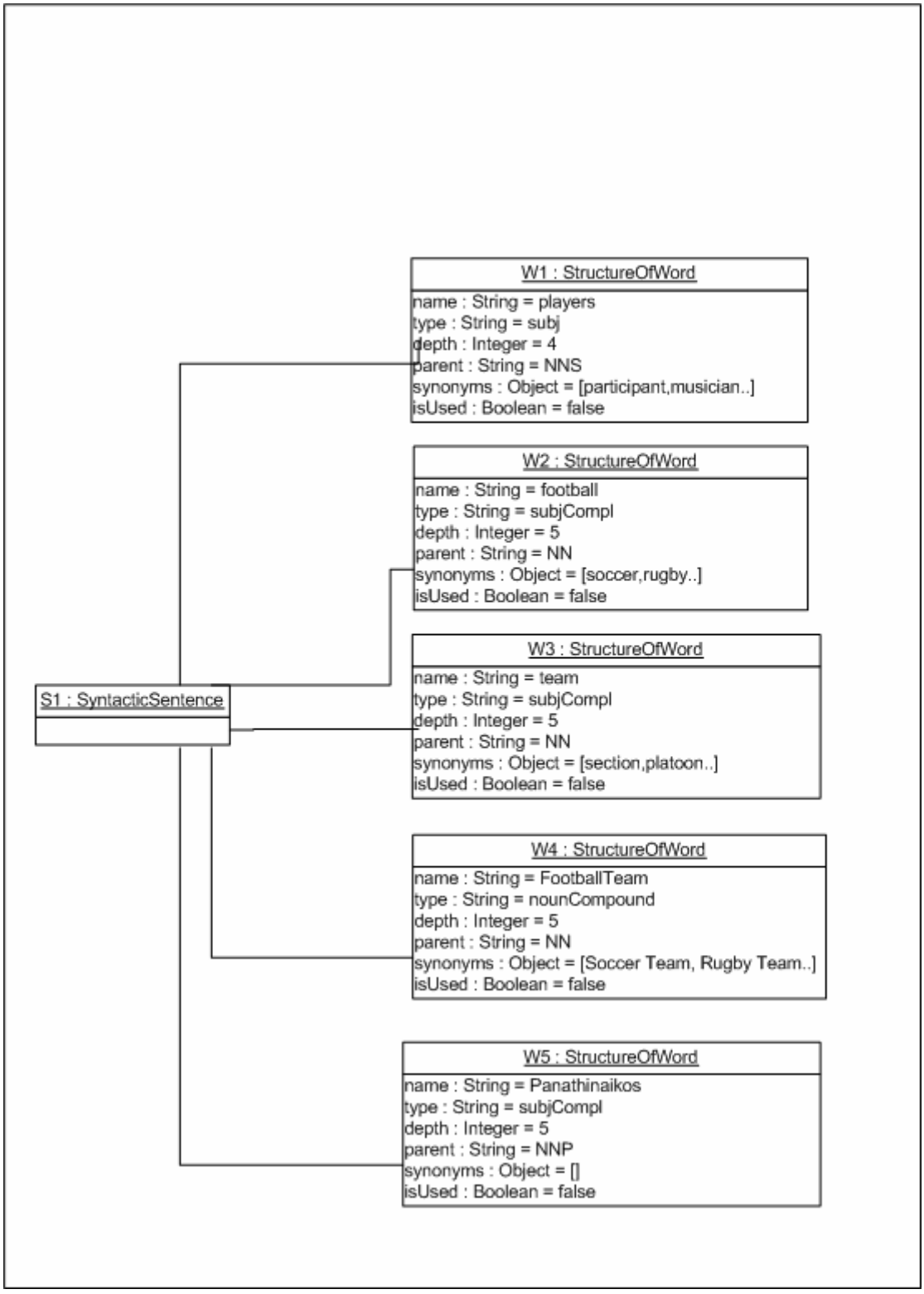
- Give me goals scored in the games between Italy and France



Εικόνα 36:Παράδειγμα SyntacticSentence για τη πρόταση goals scored in the games between Italy and France

Εδώ παρουσιάζεται ένα πιο πολύπλοκο παράδειγμα στο οποίο εκτός από operator εμφανίζονται πολλά ουσιαστικά μαζί (goals, game, Italy, France).

- List me players of football team Panathinaikos



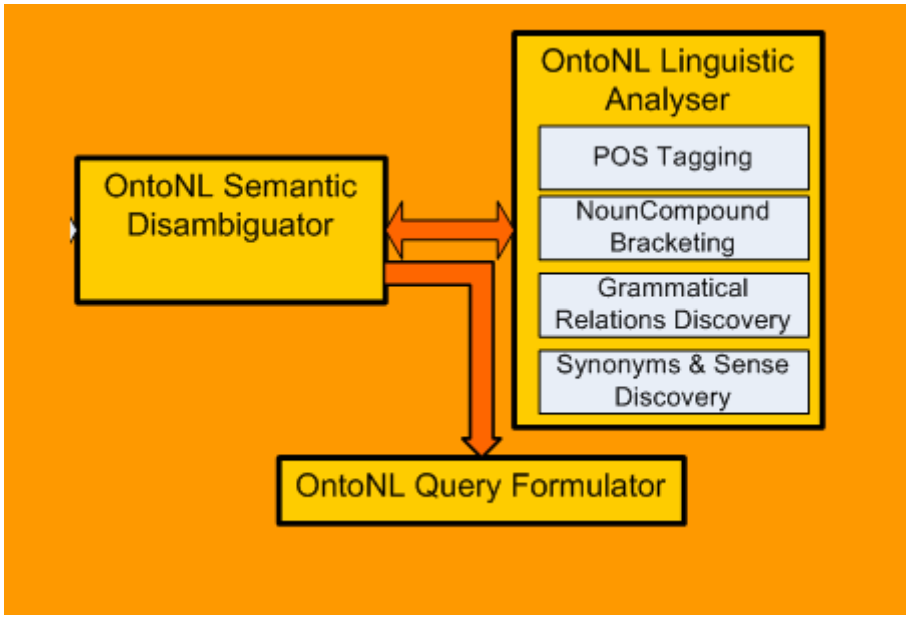
Εικόνα 37:Παράδειγμα SyntacticSentence με noun compounds

Εδώ παρουσιάζεται ένα παράδειγμα στο οποίο εμφανίζεται ένα noun compound. Παρατηρείται λοιπόν ότι εκτός από τις λέξεις που αποτελούν τη πρόταση λαμβάνεται υπόψιν στη συντακτική ανάλυση και το noun compound ουσιαστικό.

Έτσι λοιπόν σε αυτή την υποενότητα έγινε ανάλυση των ενδιάμεσων σταδίων που αποτελούν το **Linguistic Analyzer**, ποιες δομές αυτά δημιουργούν και πως επικοινωνούνε μεταξύ τους. Σε μορφή activity diagram εξηγήθηκαν οι ενέργειες που λαμβάνουν μέρος και με ποια σειρά αυτές πραγματοποιούνται. Στην επόμενη υποενότητα θα εξεταστεί το επόμενο στάδιο του **OntoNL Component**, το **Semantic Disambiguator**, πως αυτό εκμεταλλεύεται τη μέχρι τώρα επεξεργασία, ποια στάδια το αποτελούνε και ποιες δομές δημιουργούνται

5.4 Υλοποίηση της διαδικασίας εννοιολογικής αποσαφήνισης (OntoNL Semantic Disambiguator)

Στη συγκεκριμένη ενότητα γίνεται η εννοιολογική αποσαφήνιση της πρότασης. Ο εννοιολογικός αποσαφηνιστής (**Semantic Disambiguator)** επικοινωνεί με το γλωσσολογικό αναλυτή (**Linguistic Analyzer)**, χρησιμοποιεί τις δομές που αυτός δημιούργησε και μέσα από τα διάφορα στάδια επεξεργασίας κάνει την εννοιολογική αποσαφήνιση της πρότασης, δηλαδή προσπαθεί να αντιστοιχήσει τους όρους του φυσικού λόγου που έχουν υποστεί βέβαια κάποια επεξεργασία σε όρους της οντολογίας , έτσι ώστε στο επόμενο στάδιο που θα ακολουθήσει, τον συντάκτη ερωτήσεων (**Query Formulator)**, να παρουσιαστεί εκφρασμένο σε SPARQL το αίτημα του χρήστη. Φυσικά, επειδή πάντα υπάρχει η περίπτωση να υπάρχουν ελλείψεις από τα δεδομένα που έχει δώσει ο χρήστης στο σύστημα (ασάφειες), έχει αναπτυχθεί μεθοδολογία [1] που προτείνει λύσεις για τις συγκεκριμένες ελλείψεις και η οποία θα αναλυθεί παρακάτω.



Εικόνα 38: Το component Semantic Disambiguator και η επικοινωνία με τις άλλες υπομονάδες

Σε γενικές γραμμές, και όπως έχει ήδη αναφερθεί, στο αίτημα του χρήστη μπορούν να παρουσιαστούν διαφόρων επιπέδων ασάφειες. Για να γίνουν πιο κατανοητά αυτά τα επίπεδα ασαφειών, παρουσιάζονται κάποια παραδείγματα από το τομέα του ποδοσφαίρου που αναδεικνύουν τα είδη ασαφειών που είναι πιθανόν να παρουσιαστούν κατά την επεξεργασία του αιτήματος από το σύστημα.

1. Το αίτημα περιέχει λέξεις που μπορούν να επιλυθούνε – αντιστοιχηθούνε μόνο με τη γνώση της οντολογίας (ontological structures and semantics).
   ex.1 "… players of soccer team *Milan*"

   Σε αυτό το παράδειγμα οι λέξεις **players, soccer team** μπορούν να αντιστοιχηθούν σε κλάσεις της οντολογίας και η πληροφορία ότι η Milan είναι soccer team προέρχεται από την ήδη υπάρχουσα επεξεργασία της γλωσσολογικής ανάλυσης. Έτσι λοιπόν το αίτημα του χρήστη είναι πλήρως αποσαφηνισμένο λόγω της χρήσης της οντολογίας.

2. Ένα από τα subject part , object part του γλωσσικού μοντέλου δε μπορεί να αποσαφηνιστεί με τη χρήση της οντολογίας.
   ex.2 "…the players of Barcelona"

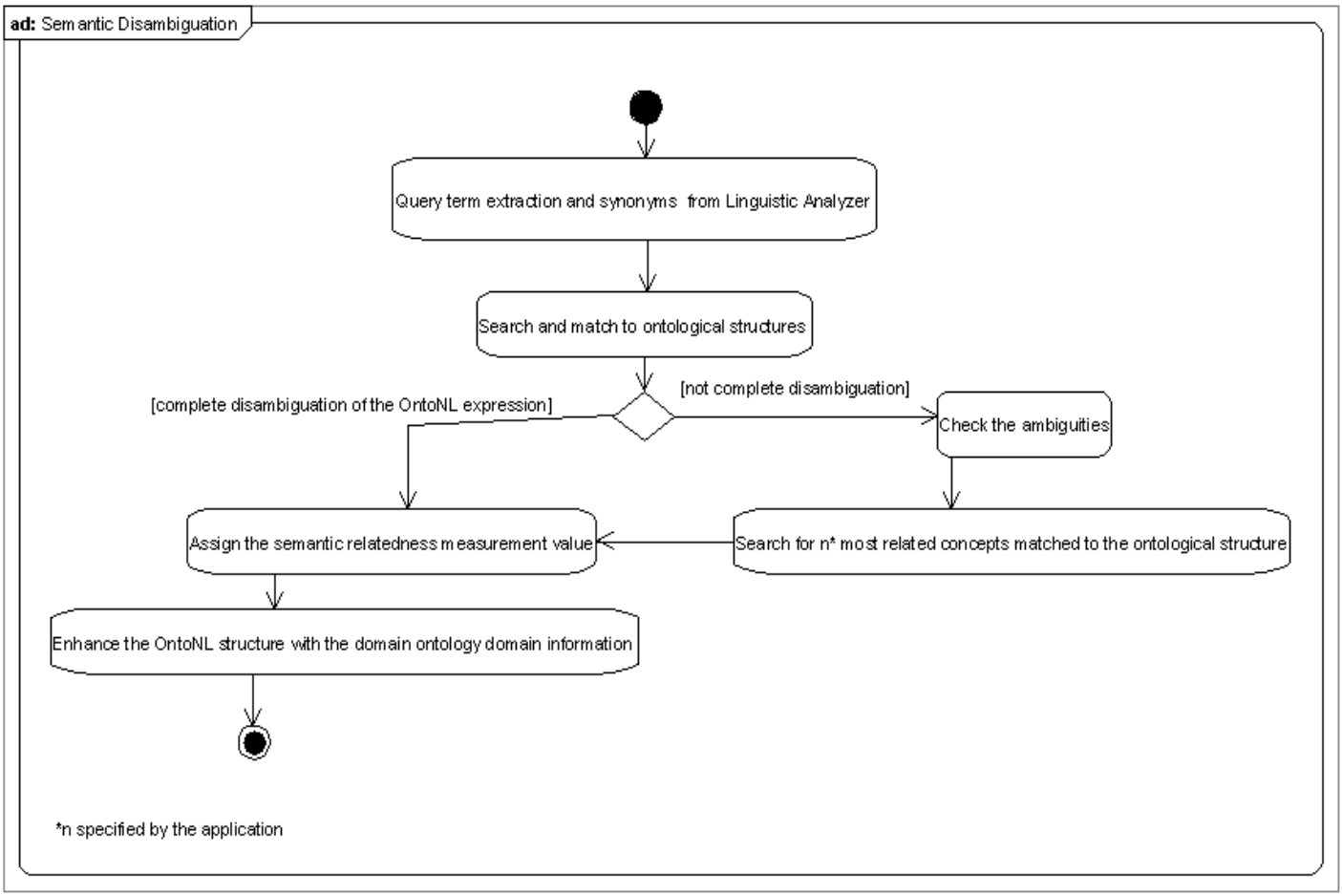
   Σε αυτό το παράδειγμα η λέξη **players** αντιστοιχίζεται μέσω της οντολογίας στην αντίστοιχη κλάση. Η ασάφεια παρατηρείται στη λέξη **Barcelona** την οποία το σύστημα την αντιμετωπίζει ως concept instance, χωρίς όμως να

γνωρίζει σε ποιο concept αντιστοιχίζεται. Στο component OntoNL Semantic Disambiguator αυτών των τύπων οι ασάφειες αντιμετωπίζονται και ο τρόπος θα παρουσιαστεί στη τρέχουσα ενότητα.

3. Ούτε το subject part ούτε το object part περιέχουν όρους που μπορούν να αποσαφηνιστούν με χρήση της οντολογίας.
   ex.3 "…information about Milan"

   Σε αυτό το παράδειγμα καμία από τις λέξεις **information**, **Milan** μπορούν να αποσαφηνιστούν με χρήση της οντολογίας. Ο συγκεκριμένος τύπος ασάφειας δε μπορεί να αντιμετωπιστεί.

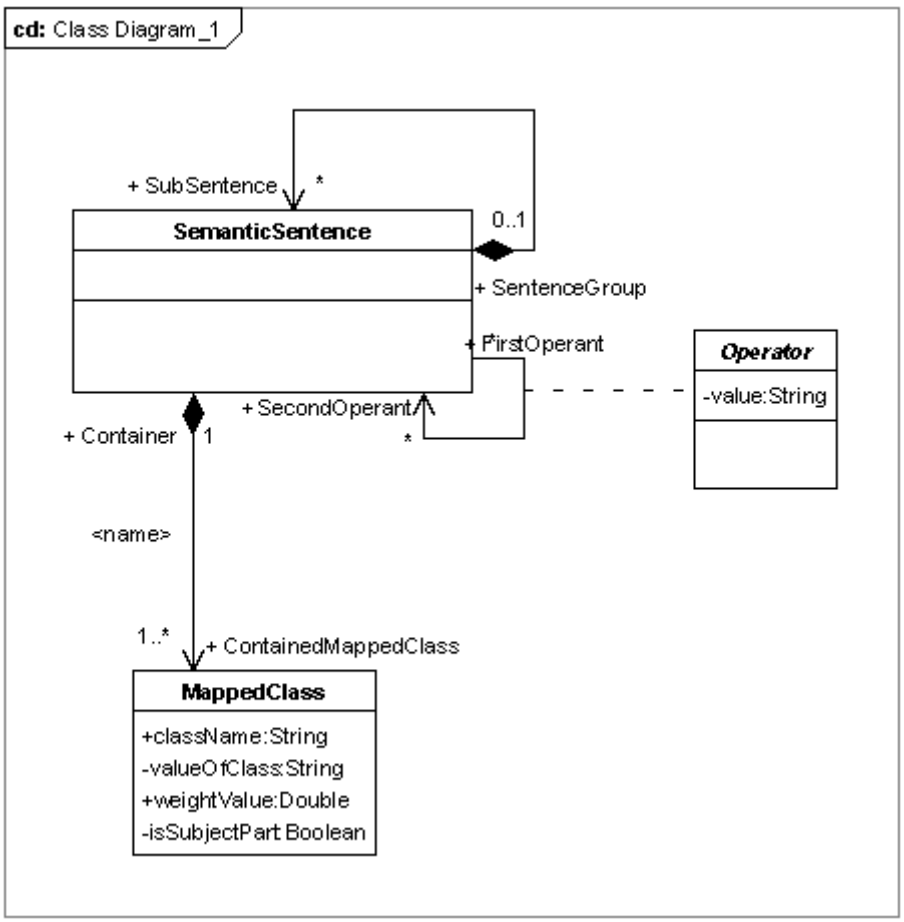
Στη συγκεκριμένη ενότητα του εννοιολογικού αποσαφηνιστή **(Semantic Disambiguator)** υπάρχουν διάφορα στάδια ελέγχου και επεξεργασίας. Ο τρόπος με τον οποίο αυτά συνδέονται και ποια είναι η σειρά με τα οποία αυτά εκτελούνται φαίνεται στο παρακάτω διάγραμμα ενεργειών (activity diagram) που δείχνει τις ενέργειες που λαμβάνουν μέρος κατά το στάδιο της εννοιολογικής αποσαφήνισης.



Εικόνα 39: Διάγραμμα ενεργειών του σταδίου εννοιολογικής αποσαφήνισης

Πριν λοιπόν αναλυθούν τα στάδια του εννοιολογικού αποσαφηνιστή (**Semantic Disambiguator)**, παρουσιάζεται η δομή που δημιουργείται στη συγκεκριμένη φάση, η οποία σταδιακά ανάλογα με τη φάση επεξεργασίας εμπλουτίζεται έτσι ώστε στο τέλος να είναι μία πλήρως εμπλουτισμένη δομή έτοιμη να μεταφραστεί σε SPARQL στο επόμενο στάδιο που ακολουθεί, τον συντάκτη ερωτήσεων **(Query Formulator)**.

Η δομή σε μορφή class diagram είναι η εξής:



Εικόνα 40: Η δομή SemanticSentence

Έτσι λοιπόν το SemanticSentence (εννοιολογική πρόταση) είναι μια εσωτερική δομή η οποία δημιουργείται και χρησιμοποιείται κατά το στάδιο της εννοιολογικής αποσαφήνισης (Semantic Disambiguation). Όπως φαίνεται και από το παραπάνω class diagram ένα SemanticSentence μπορεί να εμπεριέχει ένα ή περισσότερα SemanticSentence (SentenceGroup). Επίσης ένα SemanticSentence συνδέεται με ένα άλλο SemanticSentence μέσω operator (υπάρχει η κλάση Operator που έχει ως attribute το String value που θα είναι OR ή AND). Τέλος κάθε SemanticSentence περιέχει ένα ή περισσότερα MappedClass που ουσιαστικά αντιπροσωπεύουν με ποιο τρόπο έχει γίνει η αντιστοίχηση των λέξεων σε δομές που αφορούν την οντολογία.

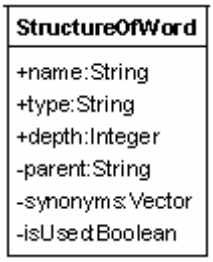
Συγκεκριμένα κάθε MappedClass περιέχει το όνομα της κλάσης που αντιστοιχίζεται στην οντολογία (className), τη τιμή που μπορεί να έχει αυτή η κλάση (valueOfClass) , το βάρος (weightValue) με το οποίο έχει εντοπιστεί η κλάση αυτή (αν βρίσκεται απευθείας από το φυσικό λόγο το βάρος είναι 1, αν η κλάση

βρίσκεται ύστερα από βοήθεια του αλγόριθμου τότε το βάρος θα είναι αυτό που δίνει ο αλγόριθμος –από 0 έως 1-) και τέλος μια βοηθητική boolean μεταβλητή isSubjectPart που χρειάζεται σε ότι αφορά το στάδιο της αποσαφήνισης και δηλώνει αν το MappedClass έχει προκύψει από το κομμάτι της πρότασης που βρίσκεται το υποκείμενο ή όχι.

Κατά το στάδιο **Search and match to ontological structures** (εικ. 39) το σύστημα προσπαθεί να αντιστοιχίσει τις δομές που έχουν δημιουργηθεί έως τώρα σε αντίστοιχους όρους της οντολογίας. Η κεντρική ιδέα είναι ότι για κάθε δομή StructureOfWord που έχει δημιουργηθεί από τον **Linguistic Analyzer** ελέγχεται αν η λέξη που αυτό περιέχει μπορεί να αντιστοιχηθεί σε κάποια κλάση της εκάστοτε οντολογίας. Αν αντιστοιχίζεται, στη δομή MappedClass που θα δημιουργηθεί θα τοποθετηθεί ως className αυτό το όνομα, αλλιώς αν δε μπορεί να αντιστοιχηθεί τότε τοποθετείται στο MappedClass ως valueOfClass. Αυτή η διαδικασία που μόλις αναφέρθηκε είναι πολύ περιληπτική και υπάρχουν πολλές περιπτώσεις που πρέπει να ληφθούν υπόψιν. Συγκεκριμένα η μεθοδολογία που ακολουθείται είναι η εξής:

Έλεγχος για ονόματα κλάσεων

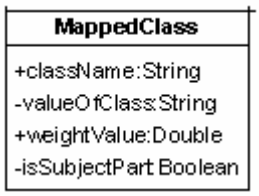
Για κάθε SyntacticSentence που υπάρχει από τη δομή που έχει δημιουργηθεί κατά το στάδιο γλωσσολογικής ανάλυσης, θα δημιουργηθεί ένα καινούργιο SemanticSentence και φυσικά θα διατηρηθεί και ο τρόπος που αυτά συνδέονταν μεταξύ τους με διάφορους operators αν αυτοί υπήρχαν. Κάθε δομή SyntacticSentence μπορεί να έχει μία ή περισσότερες δομές StructureOfWord (εικόνα 31). Έτσι λοιπόν εξετάζεται κάθε ένα από αυτά ξεχωριστά, και συγκεκριμένα ελέγχεται το name που αυτά έχουν.



Εικόνα 41: Η δομή StructureOfWord

Εξετάζεται λοιπόν αν το name αυτό μπορεί να αντιστοιχηθεί με κάποια από τις κλάσεις της οντολογίας. Συγκεκριμένα προσπελαύνεται το αρχείο που έχει

δημιουργηθεί από το component **OntoNL Ontology Processor**(εικ. 12) και στο οποίο περιέχεται όλη η προτεινόμενη ονοματολογία των κλάσεων της εκάστοτε οντολογίας. Αν είναι εφικτό να αντιστοιχηθεί με κάποια από τις κλάσεις, τότε αυτόματα δημιουργείται μια δομή MappedClass και τοποθετείται ως className το όνομα της κλάσης στο οποίο αντιστοιχήθηκε.



Εικόνα 42: Η δομή MappedClass

Επίσης η boolean (μεταβλητή που έχει ως τιμές μόνο τις true, false) μεταβλητή isUsed που έχει η συγκεκριμένη δομή StructureOfWord γίνεται true έτσι ώστε η συγκεκριμένη λέξη να μη πάρει μέρος στους υπόλοιπους ελέγχους που πρόκειται να γίνουν. Για παράδειγμα, εάν έχουμε μία οντολογία στην οποία ανήκει η κλάση Goal και επίσης έχουμε ένα StructureOfWord που έχει name τη λέξη Goal τότε γίνεται η παραπάνω αντιστοίχηση και δημιουργείται ένα MappedClass που έχει ως className το όνομα Goal.

Το δεύτερο στάδιο επεξεργασίας στη προσπάθεια του συστήματος να αντιστοιχίσει τα StructureOfWord σε όρους τις κλάσεις και με τη προϋπόθεση ότι δεν έχουν ήδη αντιστοιχηθεί (γίνεται αντιληπτό από τη μεταβλητή isUsed) είναι να ελέγχουμε τα συνώνυμα των λέξεων που έτσι κι αλλιώς έχουμε ήδη αποθηκευμένα. Έτσι επαναλαμβάνουμε τον ίδιο έλεγχο με τη διαφορά ότι τώρα εξετάζουμε τα συνώνυμα που περιέχει η δομή StructureOfWord. Αν αντιστοιχηθεί κάποιο από τα συνώνυμα, πάλι δημιουργούμε ένα MappedClass, τοποθετούμε σε αυτό className το όνομα της κλάσης στο οποίο αντιστοιχήθηκε και κάνουμε το isUsed του StructureOfWord ίσο με true. Για παράδειγμα εάν έχουμε ένα StructureOfWord με name τη λέξη Ending τότε από το πρώτο έλεγχο δε θα είχαμε καμία αντιστοίχηση, αλλά σε αυτό το στάδιο που κοιτάμε τα συνώνυμα, και αφού η λέξη Ending έχει ως συνώνυμο τη λέξη Goal, τότε θα γίνει επιθυμητή αντιστοίχηση.

Αξίζει να σημειωθεί για τα παραπάνω ότι στην ειδική περίπτωση που αντιστοιχηθεί σε κλάση της οντολογίας ένα StructureOfWord που είναι τύπου noun compound τότε ναι μεν ακολουθείται η ίδια διαδικασία με τη διαφορά όμως ότι

κάνουμε true και τις μεταβλητές isUsed των StructureOfWord που αντιπροσωπεύουν τις λέξεις που το αποτελούνε. Δηλαδή εάν έχουμε ένα StructureOfWord με name SoccerTeam και τύπο nouncompound που αντιστοιχίζετε στην οντολογία, τότε στα επόμενα στάδια δε χρειαζόμαστε να χρησιμοποιήσουμε ούτε τα StructureOfWord των λέξεων Soccer και Team μεμονωμένα και για αυτό κάνουμε true τις μεταβλητές isUsed που αυτά έχουν.

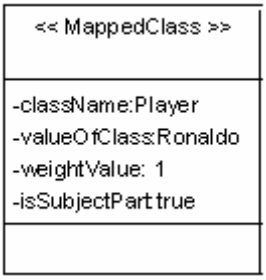
Επίσης στις παραπάνω αντιστοιχήσεις που αναλύθηκαν, αφού έχουμε άμεση αντιστοίχηση μεταξύ λέξης και κλάσης, κάνουμε το weightValue του αντίστοιχου MappedClass ίσο με 1, έτσι ώστε να καταλαβαίνουμε ότι η κλάση αυτή έχει αντιστοιχηθεί με απευθείας αναφορά στην οντολογία.

```
Program Void checkForClassNames (SyntacticSentence syntSent)
  SemanticSentence semSent;
  Begin
  For all terms i of syntSent{
    StructureOfWord sow=syntSent[i];
    String word=sow.getWord();
    If(checkIfClassWithSameNameExist(word)){
         MappedClass mc=new MappedClass();
         mc.setClassName(word);
         mc.setWeightValue(1);
         sow.setIsUsed(true);
         semSent.add(mc);
     }
    List synonyms=sow.getSynonyms();
    For all terms j of synonyms{
       String synonymsWord=synonyms[j];
       If(checkIfClassWithSameNameExist(synonymsWord)){
         MappedClass mc=new MappedClass();
         mc.setClassName(synonymsWord);
         mc.setWeightValue(1);
         sow.setIsUsed(true);
         semSent.add(mc);
       }
     }
   }
End
```

Έλεγχος για values κλάσεων

Αφού έχει προηγηθεί ο έλεγχος για ονόματα κλάσεων τώρα γίνεται έλεγχος για λέξεις που είναι values στις κλάσεις (όταν λέμε value στη κλάση εννοούμε OWL Class Instance). Πάλι εξετάζουμε κάθε StructureOfWord ξεχωριστά και η ιδιαιτερότητα είναι ότι εδώ κοιτάμε τις λέξεις που δεν έχουν χρησιμοποιηθεί σε

προηγούμενους ελέγχους. Δηλαδή, υποθετικά, στα StructureOfWord που είχαν τις λέξεις player, Ronaldo έχουμε ήδη χρησιμοποιήσει τη λέξη player ως className. Η λέξη Ronaldo που δεν έχει χρησιμοποιηθεί θα είναι valueOfClass στο ίδιο MappedClass αφού έχουν και το ίδιο depth. Επομένως θα έχουμε ένα MappedClass που θα έχει className το Player και valueOfClass το Ronaldo.

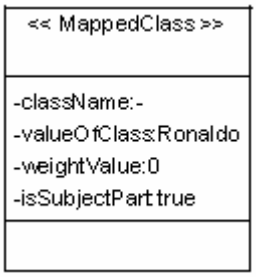


Εικόνα 43: Παράδειγμα MappedClass

```
Program Void checkForClassValues (SyntacticSentence syntSent,
                            MappedClass mc)
Begin
  For all terms i of syntSent{
    StructureOfWord sow=syntSent[i];
    String word=sow.getWord();
    If(sow.getisUsed()==false){
       mc.setValueOfClass(word);
     }
   }
End
```

Επίσης υπάρχει η περίπτωση σε ένα MappedClass να έχουμε μόνο valueOfClass και να μην υπάρχει κάποιο className. Για παράδειγμα αν είχαμε τη πρόταση goals of Ronaldo θα είχαμε δυο StructureOfWord, ένα με τη λέξη goals και ένα με τη λέξη Ronaldo που όμως έχουν διαφορετικό depth (το depth είναι attribute στο StructureOfWord –εικ.41-) κάτι που μας κάνει να συμπεράνουμε ότι το ένα δεν συνδέεται άμεσα με το άλλο. Άρα για το πρώτο StructureOfWord σύμφωνα με όλα τα παραπάνω θα είχαμε ένα MappedClass με className το Goal και κενό το valueOfClass και για δεύτερο StructureOfWord θα είχα ένα MappedClass με className κενό και valueOfClass το Ronaldo.

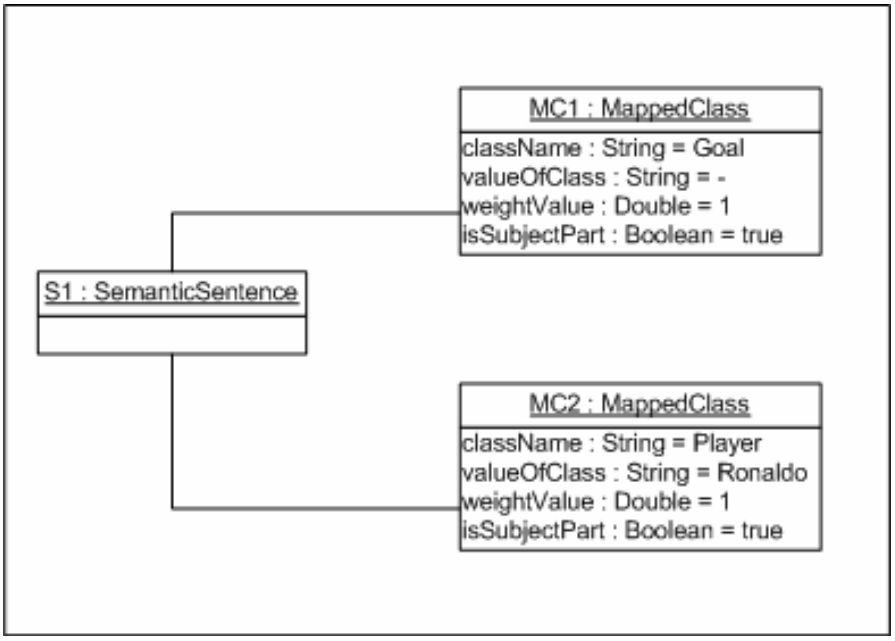
Αυτό είναι απόλυτα φυσικό και θα δούμε παρακάτω πως αντιμετωπίζονται τέτοιες περιπτώσεις στις οποίες παρουσιάζονται «ασάφειες» (ambiguities), δηλαδή περιπτώσεις στις οποίες έχουμε className κενό.



Εικόνα 44: Παράδειγμα MappedClass με ασάφεια

Ακολουθούνε κάποια χαρακτηριστικά παραδείγματα που θα καταστήσουν σαφή τα όσα έχουν αναλυθεί μέχρι τώρα για την αντιστοίχηση όρων στην οντολογία.

- 'Give me goals of player Ronaldo'

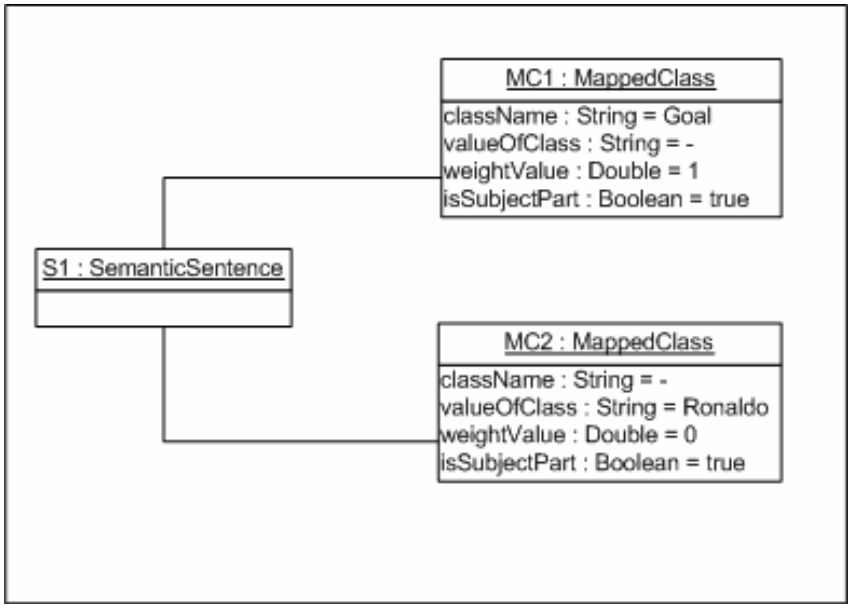


Εικόνα 45: Παράδειγμα SemanticSentence για τη πρόταση goals of player Ronaldo

Στο παραπάνω παράδειγμα παρατηρούμε ότι έχουμε ένα SemanticSentence που αποτελείται από 2 MappedClass. Κάθε ένα από αυτά τα MappedClass παρατηρούμε ότι έχει className με weightValue ίσο με 1 που σημαίνει ότι αυτές οι κλάσεις προέκυψαν από άμεση αναφορά στην οντολογία. Επίσης και τα δύο MappedClass έχουν το isSubjectPart true που σημαίνει ότι και τα δύο προέκυψαν από το κομμάτι της πρότασης που είναι το υποκείμενο (έτσι κι αλλιώς δεν έχουμε ρήμα στη πρόταση, άρα ούτε και αντικείμενα). Τέλος παρατηρούμε ότι το δεύτερο MappedClass έχει και ένα valueOfClass, τη λέξη Ronaldo. Αυτό δε σημαίνει απαραίτητα ότι το Ronaldo είναι concept instance της κλάσης Player αφού δεν έχουμε πρόσβαση στη βάση για να έχουμε διαθέσιμη τέτοια πληροφορία. Απλά προτείνεται ως concept instance, και αν

όντως υπάρχει στη βάση τότε στο μελλοντικό query που θα δημιουργηθεί και θα απευθυνθεί στη βάση δεδομένων θα υπάρξουν ανάλογα αποτελέσματα Η παραπάνω περίπτωση, αποτελεί μία ιδανική περίπτωση στην οποία στην οποία όλες οι ασάφειες (ambiguities) επιλύονται με χρήση μόνο της οντολογίας, με συνέπεια να μη χρειάζεται κάποια άλλη επεξεργασία και να είμαστε έτοιμοι να μετατρέψουμε το αίτημα του χρήστη σε SPARQL στο στάδιο του συντάκτη ερωτήσεων **(Query Formulator)**.

Ασάφειες, κάτι που είναι πολύ συχνό και σύνηθες, παρουσιάζονται όταν έχουμε δημιουργήσει κάποιο MappedClass στο οποίο ναι μεν έχουμε valueOfClass αλλά δεν έχουμε className. Τέτοιες περιπτώσεις εμφανίζονται όταν ο χρήστης δεν αναφέρει ακριβώς τι ζητάει. Ας υποθέσουμε λοιπόν τη πρόταση 'Give me goals of Ronaldo'. Σε αυτή τη πρόταση ο χρήστης δε διευκρινίζει τι είναι ο Ronaldo (ότι είναι παίχτης δηλαδή). Αποτέλεσμα αυτών είναι στο πρώτο στάδιο επεξεργασίας να δημιουργείται η παρακάτω δομή:



Εικόνα 46: Παράδειγμα SemanticSentence για τη πρόταση goals of Ronaldo

Παρατηρούμε ότι έχουμε ένα MappedClass που έχει className κενό. Αυτό αποτελεί πρόβλημα, αφού απαραίτητη προϋπόθεση για να περάσουμε στο στάδιο **Query Formulator** είναι πάντα στα MappedClass να υπάρχει className και προαιρετικά valueOfClass αφού στο SPARQL query που θα δημιουργήσουμε θα εμφανίζονται μεταβάσεις από κλάση σε κλάση που ενδεχομένως να έχουν και κάποιο value, αλλά απαραίτητα πρέπει να έχουν className.

Αυτές λοιπόν οι περιπτώσεις που εμφανίζουν ασάφειες είναι αυτές που θα επεξεργαστούν στο στάδιο **Check the Ambiguties** (εικ.39) έτσι ώστε να μπορέσει να τις λύσει και να μεταβούμε ομαλά στο επόμενο στάδιο του συντάκτη ερωτήσεων **(Query Formulator)**.

Πιο συγκεκριμένα, συμβουλευόμαστε το αρχείο που δημιουργήθηκε από τη μεθοδολογία **Concept Relatedness Calculation** του component **Ontology Processor**. Υπενθυμίζουμε ότι σε αυτό το αρχείο αποθηκεύσαμε για κάθε κλάση το βαθμό συσχέτισης της με όλες τις υπόλοιπες κλάσεις σύμφωνα με τα αποτελέσματα του αλγορίθμου. Έτσι λοιπόν αυτό που κάνουμε στη συγκεκριμένη φάση είναι να επιλέγουμε τις κλάσεις που σχετίζονται με τη κλάση που έχει ήδη εμφανιστεί στη πρόταση με βάρος συσχέτισης μεγαλύτερο από ένα κατώτατο όριο κατωφλιού (threshold). Αφού λοιπόν παίρνουμε αυτές τις κλάσεις δημιουργούμε ένα καινούργιο SemanticSentence που περιέχει όλους αυτούς τους συνδυασμούς.

```
Program Void solveTheAmbiguities (SemanticSentence semSent)

Begin
  For all terms i of semSent{
    MappedClass mc=semSent[i];
    If(chekIfAmbiguityExists(mc)==true){
         List listWithClasses=getNMostRelatedClasses();
         For all terms j of listWithClasses{
         String className=listWithClasses[j];
         Double weightValue=getWeightValue(className);
         MappedClass mc_;
         mc_.setClassName(className);
         mc_.setValueOfClass(mc.getValueOfClass());
         mc_.setWeightValue(weightValue);
         semSent.add(mc_);
     }
   }
End
```

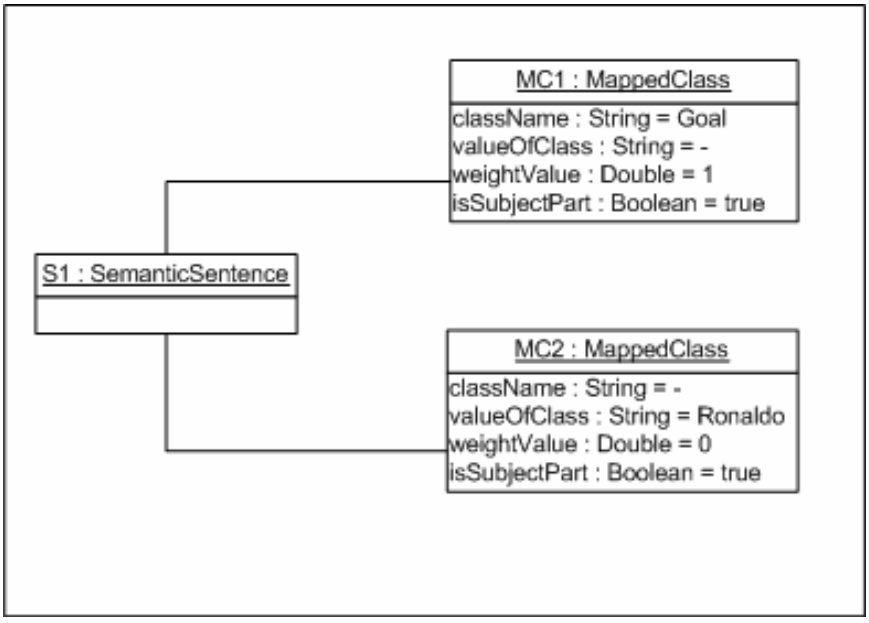
Συγκεκριμένα, στο προηγούμενο παράδειγμα (*Give me goals of Ronaldo*) παρατηρήσαμε ότι έχουμε περίπτωση ασάφειας. Έτσι λοιπόν αυτό που πρέπει να κάνουμε είναι να βρούμε τις πιο συσχετιζόμενες κλάσεις στη κλάση που ήδη έχει εμφανιστεί και είναι η κλάση Goal. Για το σκοπό αυτό κάνουμε χρήση ενός κατωφλιού (threshold), το οποίο εξαρτάται από το βαθμό συγγένειας των κλάσεων της οντολογίας. Συγκεκριμένα όσο πιο μεγάλος είναι ο βαθμός συγγένειας τόσο

μικρότερο είναι το κατώφλι δηλ. τόσες περισσότερες συσχετιζόμενες κλάσεις επιλέγουμε. Πιο συγκεκριμένα, το κατώφλι αυτό καθορίζεται από το μεγαλύτερο βαθμό συσχέτισης που αντιστοιχίζεται στην εκάστοτε κλάση της οντολογίας μειωμένο κατά ένα ποσοστό της τάξης του 0,4 (40%). Αυτό πρακτικά σημαίνει ότι το σύστημα επιστρέφει το 40% των πιο σχετικών κλάσεων στη συγκεκριμένη κλάση.

Έτσι λοιπόν για τη κλάση Goal της οντολογίας που αναφέρεται στο domain του ποδοσφαίρου [10], έχουμε ως τις πιο συσχετιζόμενες κλάσεις τις εξής:

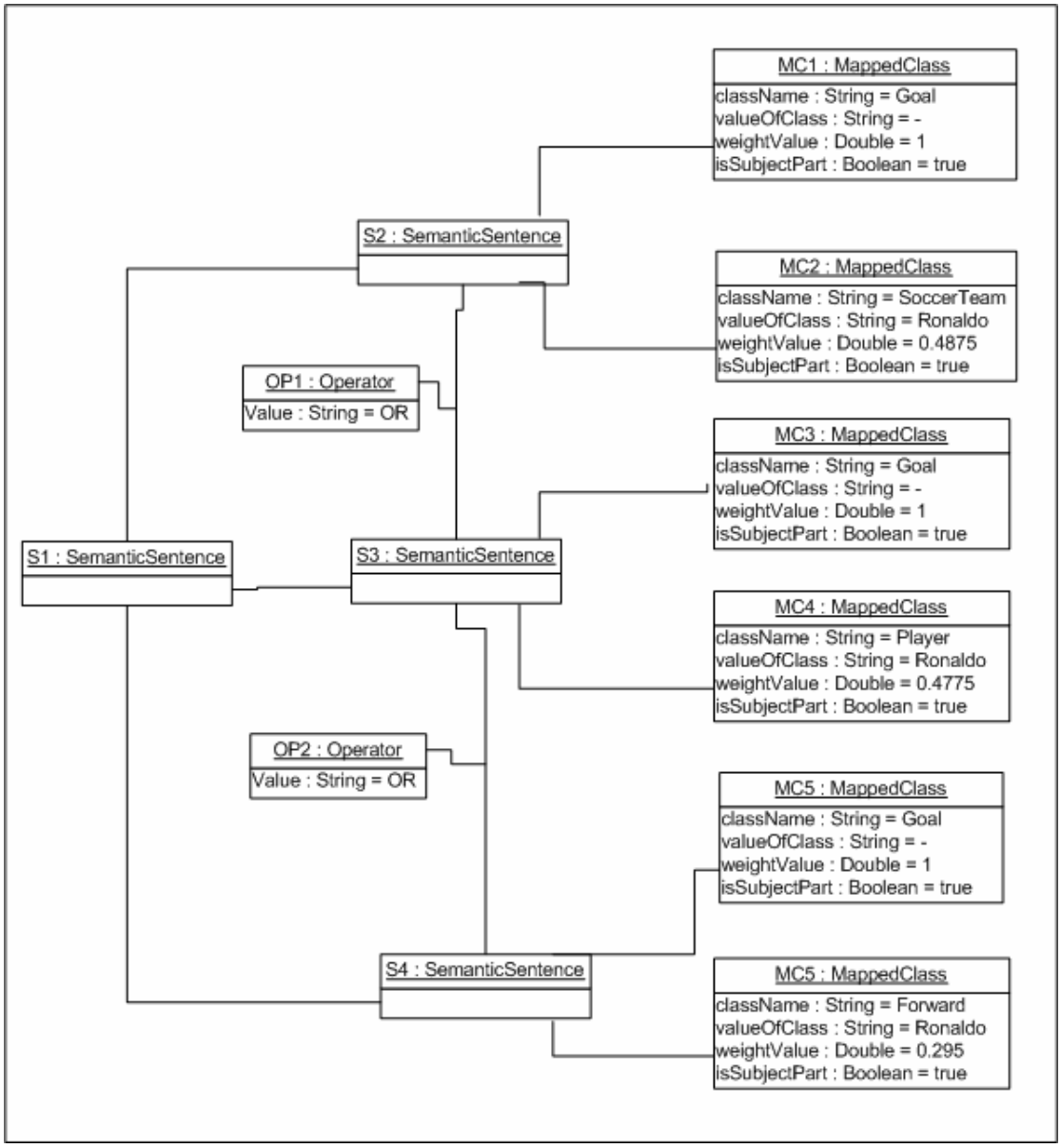
Algorithm's results consult	
most related classes to class **Goal**	
Class1: Forward	Weight: 0.295
Class2: Player	Weight: 0.4775
Class3:SoccerTeam	Weight: 0.485

Έτσι λοιπόν αφού έχουμε τις 3 πιο συσχετιζόμενες κλάσεις και ενώ η αρχική δομή ήταν η :



Εικόνα 47: Παράδειγμα SemanticSentence με ασάφεια στο υποκείμενο

η τελική μας δομή μετά το στάδιο **Check the Ambiguities** είναι η εξής:



Εικόνα 48: Παράδειγμα SemanticSentence χωρίς τις ασάφειες

Παρατηρούμε ότι σε αυτή τη νέα δομή έχουν δημιουργηθεί 3 SemanticSentence που ενώνονται μεταξύ τους με τον operator OR. Ουσιαστικά λοιπόν ζητάμε τα 'Goals of Forward Ronaldo or Goals of Player Ronaldo or Goals of SoccerTeam Ronaldo' και η ασάφεια που παρουσιαζόταν μέχρι τώρα έχει λυθεί.

Παρατηρούμε επίσης στη παραπάνω δομή ότι το weightValue στα MappedClass που δημιουργήθηκαν δεν είναι όλα 1. Βάζουμε 1 στα MappedClass τα οποία έχουν κλάση που προέκυψε με απευθείας αντιστοίχηση στην οντολογία (η κλάση Goal στο

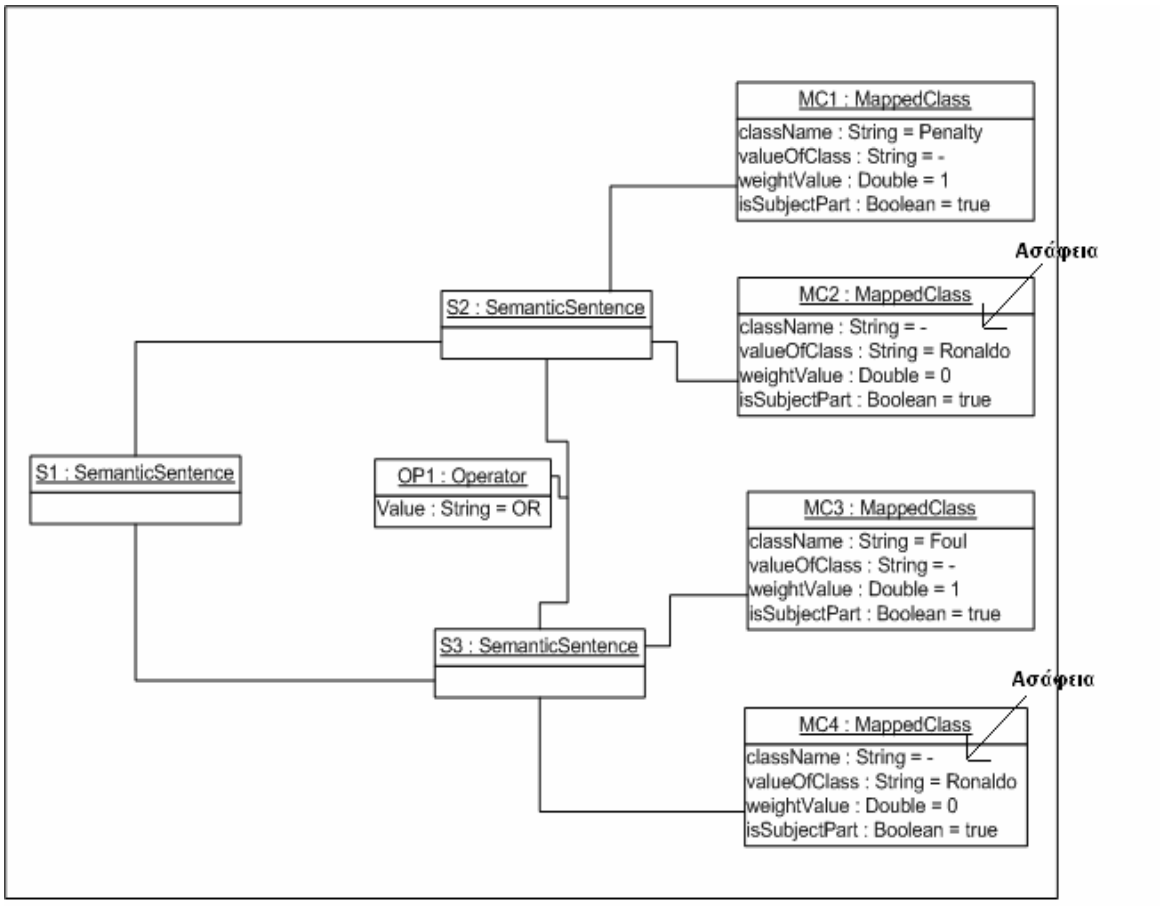
συγκεκριμένο παράδειγμα) ενώ σε διαφορετική περίπτωση βάζουμε το βάρος με το οποίο βρέθηκε αυτή η κλάση στο αρχείο που μας δίνει τους βαθμούς συσχέτισης.

Παρατηρούμε επίσης στο παραπάνω παράδειγμα ότι μετά από αυτή την επεξεργασία η δομή μας είναι πλήρης (αν και μεγαλύτερη) αφού δεν έχει κανένα MappedClass που να έχει κενό το className κάτι που σημαίνει ότι μπορούμε να περάσουμε στο επόμενο στάδιο επεξεργασίας, το **Query Formulator**.

Ακολουθεί ένα λίγο πιο πολύπλοκο παράδειγμα στο οποίο γίνεται εφαρμογή της προαναφερθείσας μεθοδολογίας.

- 'Give me penalties and fouls of Ronaldo'

Από το στάδιο **Search and match to ontological structures** (εικ.39) δημιουργείται η εξής δομή:



Εικόνα 49: Παράδειγμα SemanticSentence με ασάφειες και λογικό τελεστή

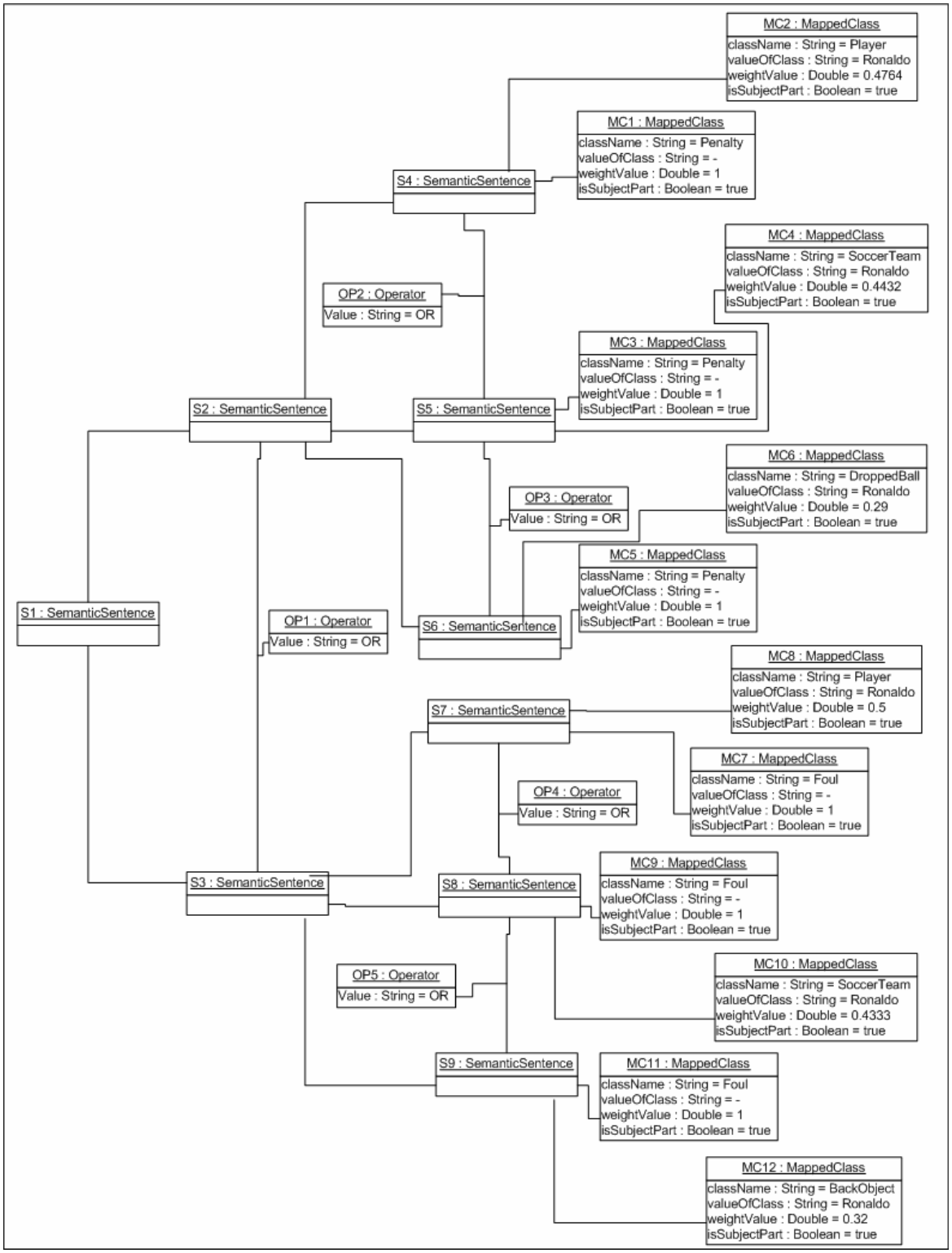
Και σε αυτή τη περίπτωση παρατηρούμε ότι έχουμε ασάφειες αφού υπάρχουν 2 MappedClass τα οποία έχουν κενό το className. Έτσι πρέπει να γίνει επεξεργασία μέσω του σταδίου **Ambiguity Resolution**. Έτσι λοιπόν, όμοια με προηγουμένως

συμβουλευόμαστε το αρχείο με τους βαθμούς συσχέτισης και έχουμε ως αποτελέσματα τα εξής:

Algorithm's results consult	
most related classes to class **Penalty**	
Class1: DroppedBall	Weight: 0.29
Class2: SoccerTeam	Weight: 0.4432
Class3:Player	Weight: 0.4764

Algorithm's results consult	
most related classes to class **Foul**	
Class1: BackObject	Weight: 0.32
Class2: SoccerTeam	Weight: 0.4333
Class3:Player	Weight: 0.5

Έτσι λοιπόν η τελική δομή που δημιουργείται μετά από αυτό το στάδιο είναι η εξής:



Εικόνα 50: Παράδειγμα SemanticSentence χωρίς ασάφειες

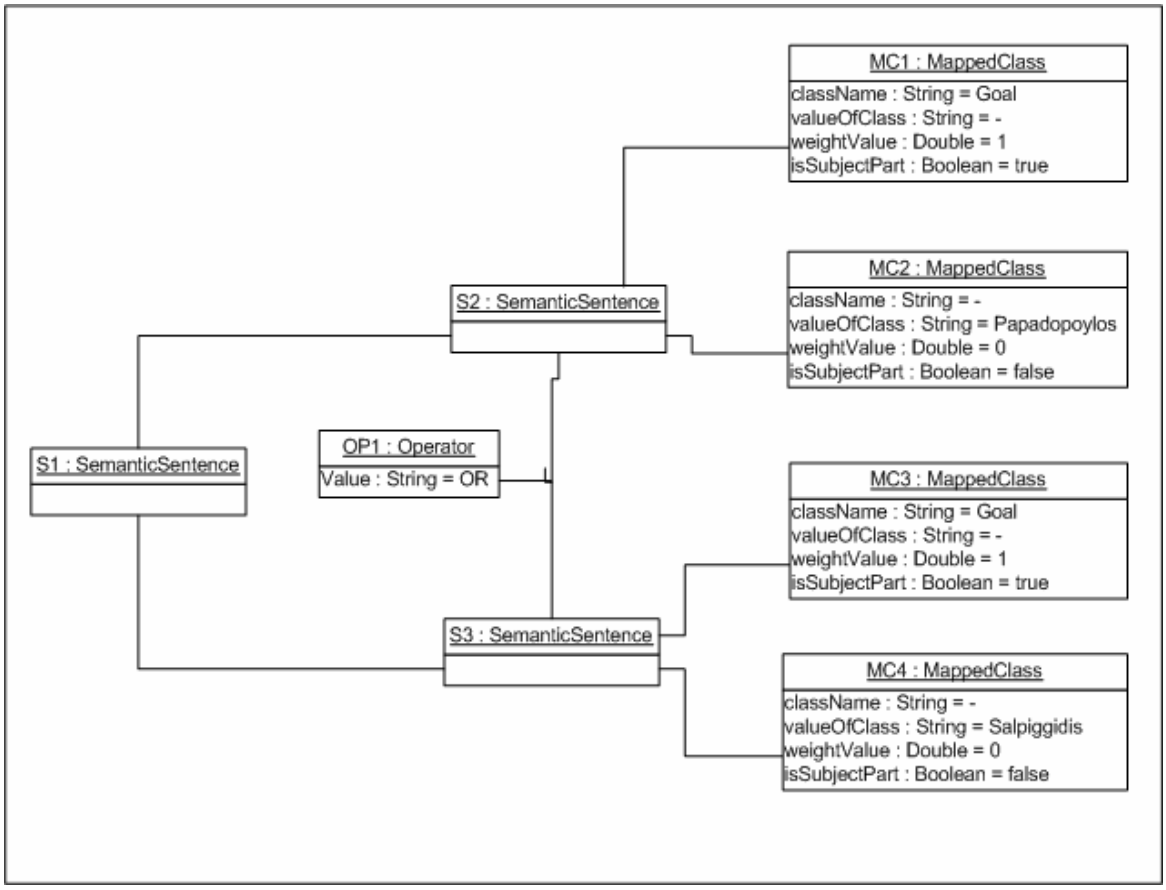
Παρατηρούμε ότι έχει λυθεί το πρόβλημα με τις ασάφειες και είμαστε έτοιμοι να προχωρήσουμε στο επόμενο στάδιο.

(*)Επέκταση του εννοιολογικού αποσαφηνιστή (OntoNL Semantic Disambiguator Extension)

Εκτός από τη διαδικασία που μόλις περιγράφηκε και σύμφωνα με την οποία συμβουλευόμαστε το αρχείο με τους βαθμούς συσχέτισης ώστε να επιλυθούν οι διάφορες ασάφειες, έχει σχεδιαστεί - υλοποιηθεί στα πλαίσια της παρούσας διπλωματικής εργασίας και κάποια επιπλέον μεθοδολογία που επεξεργάζεται το ρήμα της πρότασης προσπαθώντας να «αντλήσει» χρήσιμη πληροφορία από αυτό, με απώτερο σκοπό να αντιμετωπίσει τυχόν ασάφειες που παρουσιάζονται.

Πιο συγκεκριμένα υπάρχει περίπτωση σε μία πρόταση να εμφανίζεται ρήμα το οποίο να έχει την ίδια ρίζα με κάποιο object property της οντολογίας. Τότε σε αυτή τη περίπτωση, και αν φυσικά εμφανίζονται ασάφειες στη δομή που έχει ήδη δημιουργηθεί, συμβουλευόμαστε τις domain και range κλάσεις του συγκεκριμένου object property και ανάλογα με την ασάφεια τοποθετούμε τη κατάλληλη κλάση στη υπάρχουσα δομή, αφού είμαστε σίγουροι ότι υπάρχει συσχέτιση ανάμεσα στο ρήμα και το αντίστοιχο object property. Με αυτό τον τρόπο λοιπόν εκμεταλλευόμαστε τη σημασιολογία του ρήματος

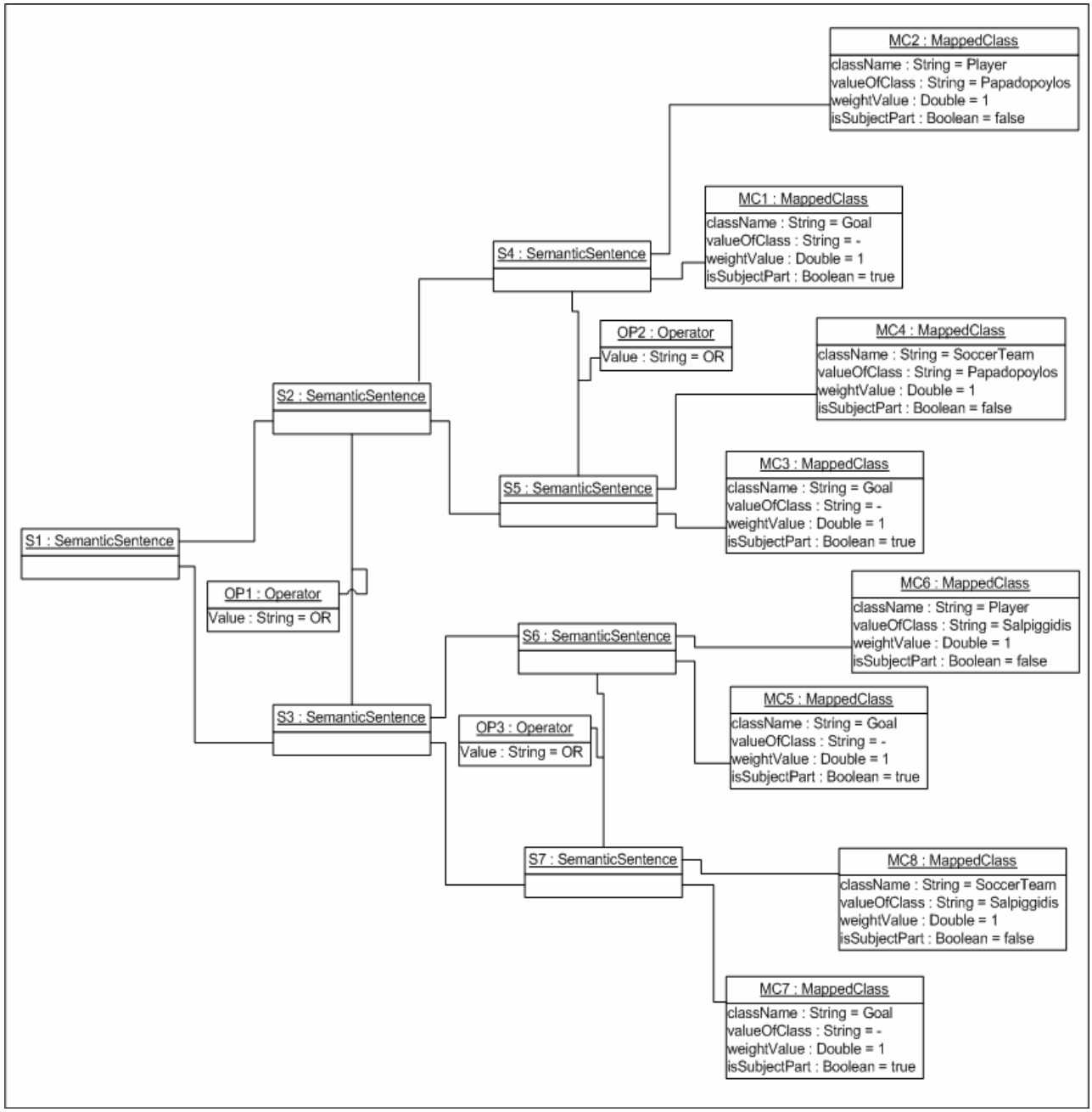
Για παράδειγμα, ας υποθέσουμε τη πρόταση ‘Give me goals scored by Papadopoylos or Salpiggidis’. Σε αυτή τη περίπτωση από το στάδιο **Search and match to ontological structures** (εικ.39) θα είχαμε τη δομή:



Εικόνα 51: Παράδειγμα SemanticSentence

στην οποία παρατηρούμε ότι έχουμε ασάφειες αφού υπάρχουν δομές MappedClass οι οποίες έχουν className κενό. Όμως πριν περάσουμε στο στάδιο που συμβουλευόμαστε το αρχείο με τους βαθμούς συσχέτισης, μπορούμε να κάνουμε τον εξής έλεγχο, αφού πρώτα υποθέσουμε ότι η κλάση Goal της οντολογίας έχει ένα object property με το όνομα scoredBy το οποίο το συνδέει με τις κλάσεις Player και SoccerTeam (όπως συμβαίνει στην οντολογία ποδοσφαίρου στην οποία έχουν γίνει πειράματα).

Παρατηρούμε ότι το ρήμα της πρότασης scored έχει την ίδια ρίζα με το object property της κλάσης Goal scoredBy. Έτσι λοιπόν αντί να συμβουλευτούμε το αρχείο μπορούμε να προτείνουμε τις κλάσεις που η κλάση Goal συνδέεται μέσω αυτού του object property και είναι οι κλάσεις Player και SoccerTeam. Έτσι λοιπόν η δομή τελικά θα γίνει:



Εικόνα 52: Παράδειγμα SemanticSentence χωρίς ασάφειες

Έτσι λοιπόν, με τη παραπάνω μεθοδολογία, εκμεταλλευόμαστε – αξιοποιούμε και το ρήμα της πρότασης όταν αυτό πληρεί ορισμένες προϋποθέσεις. Το συγκεκριμένο στάδιο ελέγχου εφαρμόζεται μόνο όταν έχει γίνει η αντιστοίχηση των λέξεων σε όρους της οντολογίας (**Search and match to ontological structures** (εικ.39)) και υπάρχουν ακόμη ασάφειες. Με το τρόπο αυτό, προτείνουμε στο σύστημα κλάσεις οι οποίες σίγουρα συσχετίζονται άμεσα μεταξύ τους, αφού συνδέονται μέσω object properties. Φυσικά υπάρχουν περιπτώσεις που το ρήμα της πρότασης δεν αντιστοιχίζεται σε κάποιο object property είτε δεν υπάρχει καθόλου ρήμα. Σε αυτές τις περιπτώσεις οι ασάφειες αντιμετωπίζονται με το τρόπο που ήδη έχει αναφερθεί,

σύμφωνα δηλαδή με τα αποτελέσματα του αλγορίθμου **OntoNL Semantic Relatedness Measure Algorithm** [2,3,4].

Υπάρχουν πολλά παραδείγματα προτάσεων στα οποία η παραπάνω μεθοδολογία που εκμεταλλεύεται τη σημασιολογία του ρήματος μπορεί να φανεί ιδιαίτερα χρήσιμη. Ιδιαίτερα σε περιπτώσεις που στο αίτημα του ο χρήστης δεν αναφέρει κανέναν όρο – ουσιαστικό που μπορεί να αντιστοιχηθεί στην οντολογία, η αξιοποίηση της σημασιολογίας του ρήματος, όπου αυτό βέβαια είναι εφικτό, μπορεί να βοηθήσει στην αποσαφήνιση του αιτήματος ενώ η μέχρι τώρα προτεινόμενη μεθοδολογία με τους βαθμούς συσχέτισης δε θα μπορούσε να εκμεταλλευτεί αφού δεν εμφανίζονται concepts στο αίτημα του χρήστη. Για παράδειγμα:

- Who *scored* for Barcelona?
- Did anyone *score* for Barcelona?
- Does Ronaldo *play* for Milan?
- Does Raul *belong* to Arsenal?

ή ακόμα και όταν εμφανίζονται όροι οντολογίας στη πρόταση

- Which players *scored* for Barcelona?
- Give me goals *scored* by Arsenal
- Give me fouls *commited* by Zidane

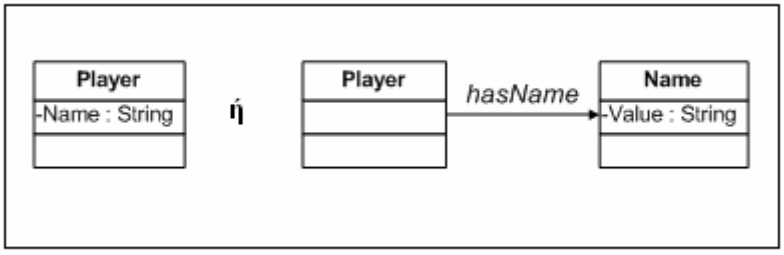
Σε όλα τα παραπάνω παραδείγματα, αν στην οντολογία που εξετάζεται υπάρχουν object properties που να αντιπροσωπεύουν τις αντίστοιχες ενέργειες που υποδηλώνονται από τα ρήματα (ScoredByRelation, PlayRelation, BelongRelation, CommitedByRelation κτλ.) τότε μπορεί να εφαρμοστεί η παραπάνω μεθοδολογία και να εκμεταλλευτεί η σημασιολογία των ρημάτων, δηλαδή να χρησιμοποιηθούν οι range και οι domain κλάσεις του συγκεκριμένου property για να επιλυθούν οι ασάφειες.

Αξίζει να σημειωθεί επίσης ότι στις περιπτώσεις που εμφανίζονται και όροι της οντολογίας στο αίτημα του χρήστη αλλά και ρήμα που να αντιστοιχίζεται με κάποιο object property, οι δύο μεθοδολογίες αποσαφήνισης (βάρη συσχέτισης, σημασιολογία ρήματος) λειτουργούν παράλληλα έτσι ώστε να εξασφαλιστεί το μέγιστο της αποσαφήνισης.

```
Program Void solveTheAmbiguitiesExtension (SemanticSentence
                                          semSent, String verb)
Begin
  For all terms i of semSent{
    MappedClass mc=semSent[i];
```

```
    If(chekIfAmbiguityExists(mc)==true){
         If(verbLooksLikeObjectProperty(verb)==true){
          String objectProperty=getObjectProperty(verb);
          List rangeClasses=getRangeClasses(objectProperty);
          List domainClasses=getDomainClasses(objectProperty);
          If(ambiguityInSubjectPart()==true){
             For all terms j of domainClasses{
                String className=domainClasses[j];
                 mc.setClassName(className);
                 mc.setWeightValue(1);
                 semSent.add(mc);
               }
             }
          elseIf(ambiguityInObjectPart==true){
             For all terms j of rangeClasses{
                String className=domainClasses[j];
                 mc.setClassName(className);
                 mc.setWeightValue(1);
                 semSent.add(mc);
               }
             }
           }
   }
End
```

Τέλος πρέπει να διευκρινιστεί ο όρος value κλάσης (έχει αναφερθεί πολλές φορές παραπάνω). Όταν αναφερόμαστε σε αυτό τον όρο εννοούμε πως το value αυτό θα είναι τιμή στα data properties της συγκεκριμένης κλάσης ή στα data properties των κλάσεων που είναι γείτονες αυτής της κλάσης. Και αυτό γιατί αν σκεφτούμε τη περίπτωση Player Ronaldo που το Ronaldo είναι value της κλάσης Player θα μπορούσαν να ισχύουν τα εξής: Η κλάση Player να έχει ένα data property name επομένως το Ronaldo θα απευθυνόταν εκεί ή η κλάση Player να συνδέεται μέσω ενός object property με μια κλάση Name και επομένως το Ronaldo να απευθύνεται σε ένα data property της κλάσης Name.



Επομένως στα sparql queries που δημιουργούνται ισχύει η παραπάνω θεώρηση όσον αφορά τα values των κλάσεων.

5.5 Υλοποίηση του συντάκτη ερωτήσεων (OntoNL Query Formulator)

Αυτό είναι και το τελικό στάδιο του **OntoNL Component** (εικ.11). Σε αυτό το στάδιο αξιοποιείται η δομή που προέκυψε από την εννοιολογική αποσαφήνιση του σταδίου **Semantic Disambiguator** και γίνεται η μετάφραση - μετατροπή της σε SPARQL. Πριν παρουσιαστεί η μεθοδολογία που ακολουθείται κατά τη μετατροπή των δομών αυτών σε SPARQL, παρουσιάζονται σε ένα πίνακα [1] όλοι οι πιθανοί τύποι ερωτήσεων που μπορεί να έχουν εμφανιστεί και τροποποιηθεί από τα προηγούμενα στάδια επεξεργασίας (Linguistic Analyzer, Semantic Disambiguator), έτσι ώστε να γίνει πιο κατανοητή και η διαδικασία που ακολουθείται κατά τη μετάβαση σε SPARQL.

	SUBJECT PART	OPER.	VERB	OBJECT PART	OPER.
1	Class or Datatype Property				
2	Class or Datatype Property	AND/ OR			
3	Class or Object - Datatype Property		1	Class or Object - Datatype Property Value	
4a	Class or Object - Datatype Property	AND/ OR	1	Class or Object - Datatype Property Value	
4b	Class or Object - Datatype Property		1	Class or Object - Datatype Property Value	AND/ OR
4c	Class or Object - Datatype Property	AND/ OR	1	Class or Object - Datatype Property Value	AND/ OR
5	Value				
6	Value		AND/ OR		
7	Value		1	Value [1, N]	
8a	Value	AND/ OR	1	Value	
8b	Value		1	Value	AND/ OR
8c	Value	AND/ OR	1	Value	AND/ OR

Εικόνα 53:Τύποι ερωτήσεων χρήστη μετά απο τη δομική αποσαφήνιση από τις Οντολογίες (Types of user queries after the structural disambiguation from Ontologies[1])

Έτσι λοιπόν, έχοντας δεδομένη μια δομή που η μορφή της υπακούει στο παραπάνω πίνακα, η μεθοδολογία που ακολουθείται κατά τη μετάβαση σε SPARQL είναι η εξής. Στο SELECT κομμάτι που περιέχεται στο query τοποθετούνται οι

μεταβλητές εκείνες για τις οποίες θέλουμε να δούμε αποτελέσματα μετά το τέλος εκτέλεσης του. Στο WHERE κομμάτι, γίνεται η μετάβαση σε μορφή τριάδων από μία κλάση σε μία άλλη μέσω object properties ή από μία κλάση σε ένα value μέσω data type properties με σκοπό να οδηγηθούν οι μεταβλητές μέσω κατάλληλων μονοπατιών στο σωστό αποτέλεσμα.

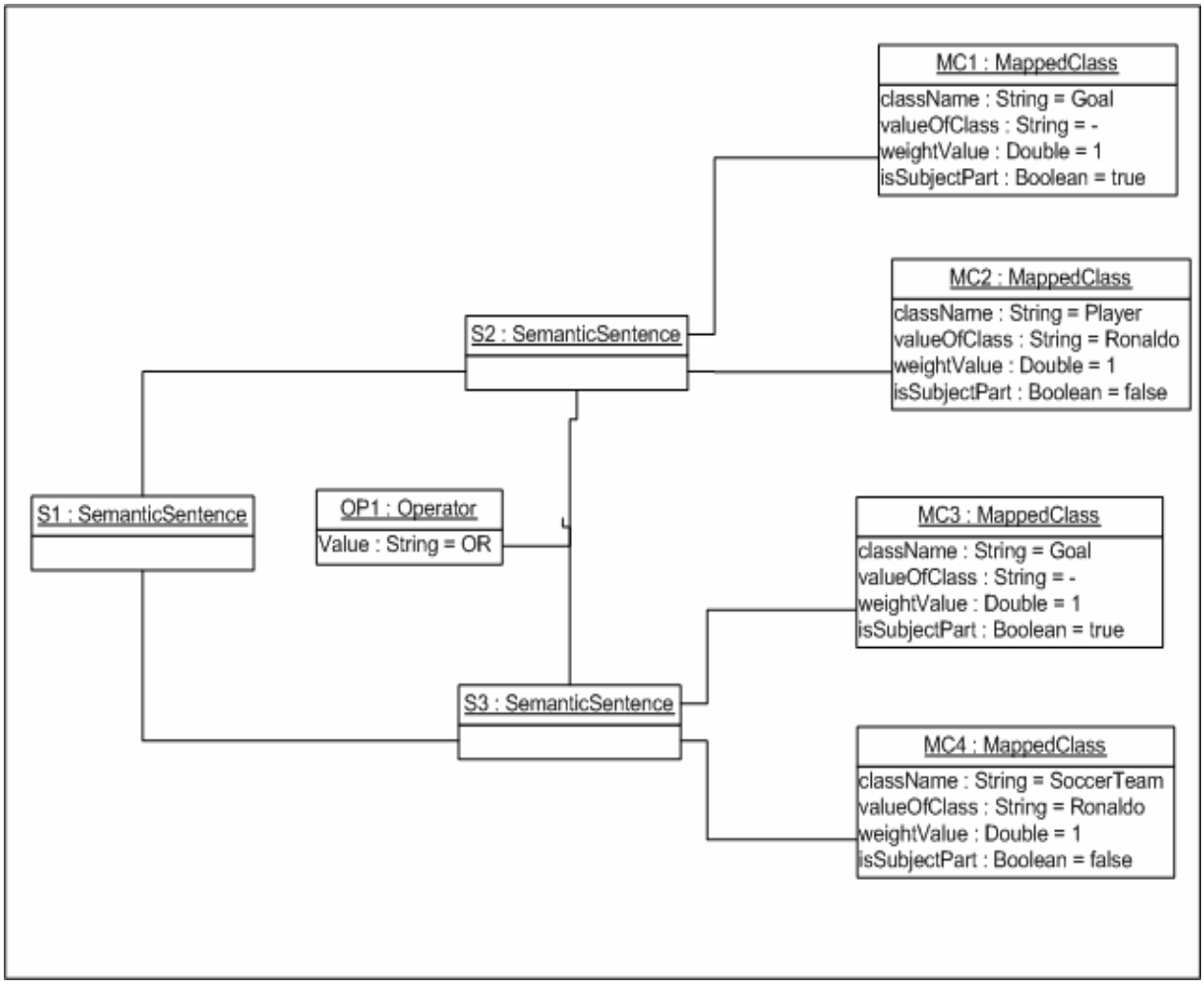
Έτσι λοιπόν, η μεθοδολογία που ακολουθείται είναι για κάθε δομή MappedClass που υπάρχει από το στάδιο της εννοιολογικής αποσαφήνισης να συνδέεται με το επόμενο του μέσω object properties και αν το συγκεκριμένο MappedClass έχει και value τότε να συνδέονται μαζί μέσω data type properties.

```
Program String sparqlBuilder (List listOfSemSent)

Begin
  String sparqlQuery;
  String prefixes=getPrefixesFromOntology();
  sparqlQuery=sparqlQuery + prefixes;
  sparqlQuery=sparqlQuery + "SELECT"
  String className= getSelectVariable(semSent);
  sparqlQuery=sparqlQuery+ "?" + className
  String whereClause;
  whereClause=whereClause+"WHERE";

  For all terms k of listOfSemSent{
      SemanticSentence semSent=listOfSemSent[k];
      if(k>1){
              If(semSent.getOperand().equals("OR"))
                 whereClause=whereClause+"UNION";
              Else if(semSent.getOperand().equals("AND"))
                 whereClause=whereClause+".";
             }
  For all terms i of semSent{
    If(i<semSent.size){
      MappedClass mc1=semSent[i];
      MappedClass mc2=semSent[i+1];
      String objectProperties=
                getObjectPropertiesConnectingClasses(mc1,mc2);
      whereClause=whereClause+ "?"+mc1.getClassName()+
                 objectProperties+ "?"+mc2.getClassName()+".";
      If(mc1.getValueOfClass().length>0){
        String className=mc1.getClassName();
        List dataProperties=getDataTypeProperties(className);
           For all terms j of dataProperties{
              String dp=dataProperties[j];
              whereClause=whereClause+"{ ?"+mc1.getClassName()
                         + dp + mc1.getValueOfClass(); +"}";
              If(j<dataProperties.size())
```

```
                whereClause=whereClause+ "UNION";
            }
          }
        }
    sparqlQuery=sparqlQuery+whereClause;
    return sparqlQuery;
End
```

Η παραπάνω μεθοδολογία γίνεται κατανοητή με το εξής παράδειγμα:

Έστω ότι το σύστημα επεξεργάζεται τη πρόταση 'Give me goals scored by Ronaldo'. Η τελική δομή που προκύπτει από το Semantic Disambiguator είναι η εξής:



Εικόνα 54: Παράδειγμα SemanticSentence για τη πρόταση goals scored by Ronaldo

Έχοντας λοιπόν δεδομένη τη συγκεκριμένη δομή, ξεκινάει η μετατροπή σε SPARQL. Αυτό που θα υπάρχει στο SELECT κομμάτι του query θα είναι η κλάση για την οποία ζητούνται πληροφορίες. Εδώ λοιπόν είναι η κλάση Goal. Στη συνεχεία ακολουθεί το WHERE κομμάτι του query. Σε αυτό το κομμάτι ουσιαστικά γίνεται η σύνδεση των MappedClass μεταξύ τους. Επίσης, στη παραπάνω δομή εμφανίζονται

δυο SemanticSentence που ενώνονται μεταξύ τους με OR επομένως στο WHERE κομμάτι θα υπάρχουν δυο υποενότητες , μια στην οποία θα ζητούνται τα Goal του SoccerTeam Ronaldo και μια στην οποία θα ζητούνται τα Goal του Player Ronaldo, και οι οποίες θα ενώνονται με OR.

Για κάθε λοιπόν υποενότητα θα γίνει το εξής. Θα εξετάζεται κάθε MappedClass που ανήκει στο συγκεκριμένο SemanticSentence και θα γίνεται η ένωση μέσω object properties της κλάσης που προσδιορίζει το συγκεκριμένο MappedClass με τη κλάση του επόμενου MappedClass. Τα object properties είναι γνωστά από το αρχείο που έχει δημιουργηθεί από τον **OntoNL Ontology Processor** και εμφανίζει τη βέλτιστη διαδρομή μέσω object properties που συνδέουν 2 κλάσεις . Εφόσον υποθετικά για να γίνει μετάβαση από τη κλάση A στη κλάση B απαιτούνται και ενδιάμεσοι σταθμοί , αυτοί θα φαίνονται στο SPARQL query. Στο συγκεκριμένο παράδειγμα η κλάση Goal συνδέεται άμεσα με τις κλάσεις SoccerTeam και Player με το object property scoredBy επομένως αυτές θα είναι και οι μόνες κλάσεις οι οποίες θα συμμετέχουν στο query.

Εκτός από αυτή τη σύνδεση μέσω κλάσεων, γίνεται επίσης το εξής. Αν κάποια κλάση έχει και value (γίνεται αντιληπτό από το MappedClass) τότε πρέπει να συσχετιστεί με κάποιο τρόπο η κλάση αυτή με το value. Έτσι λοιπόν γίνεται σύνδεση της κλάσης αυτής μέσω όλων των data properties που έχει με το συγκεκριμένο value (μεταξύ τους συνδέονται με OR) και επίσης γίνεται σύνδεση της συγκεκριμένης κλάσης με όλους τους γείτονες -κλάσεις και όμοια γίνεται σύνδεση του value με τα data properties των κλάσεων αυτών (επίσης μεταξύ τους συνδέονται μέσω OR).

```
?Goal rdf:type prefix7:Goal.
?Goal prefix7:ScoredByRelation ?PlayerObject.
{
        {?PlayerObject prefix2:ShirtNumber "Ronaldo"}UNION
        {?PlayerObject prefix0:id "Ronaldo"}UNION
        {?PlayerObject prefix0:mediaTimeUnit "Ronaldo"}UNION
        {?PlayerObject prefix0:timeUnit "Ronaldo"}UNION
        {?PlayerObject prefix0:timeBase "Ronaldo"}UNION
        {?PlayerObject prefix0:mediaTimeBase "Ronaldo"}
}
```

Εικόνα 55:Παράδειγμα υποενότητας ενός Sparql Query

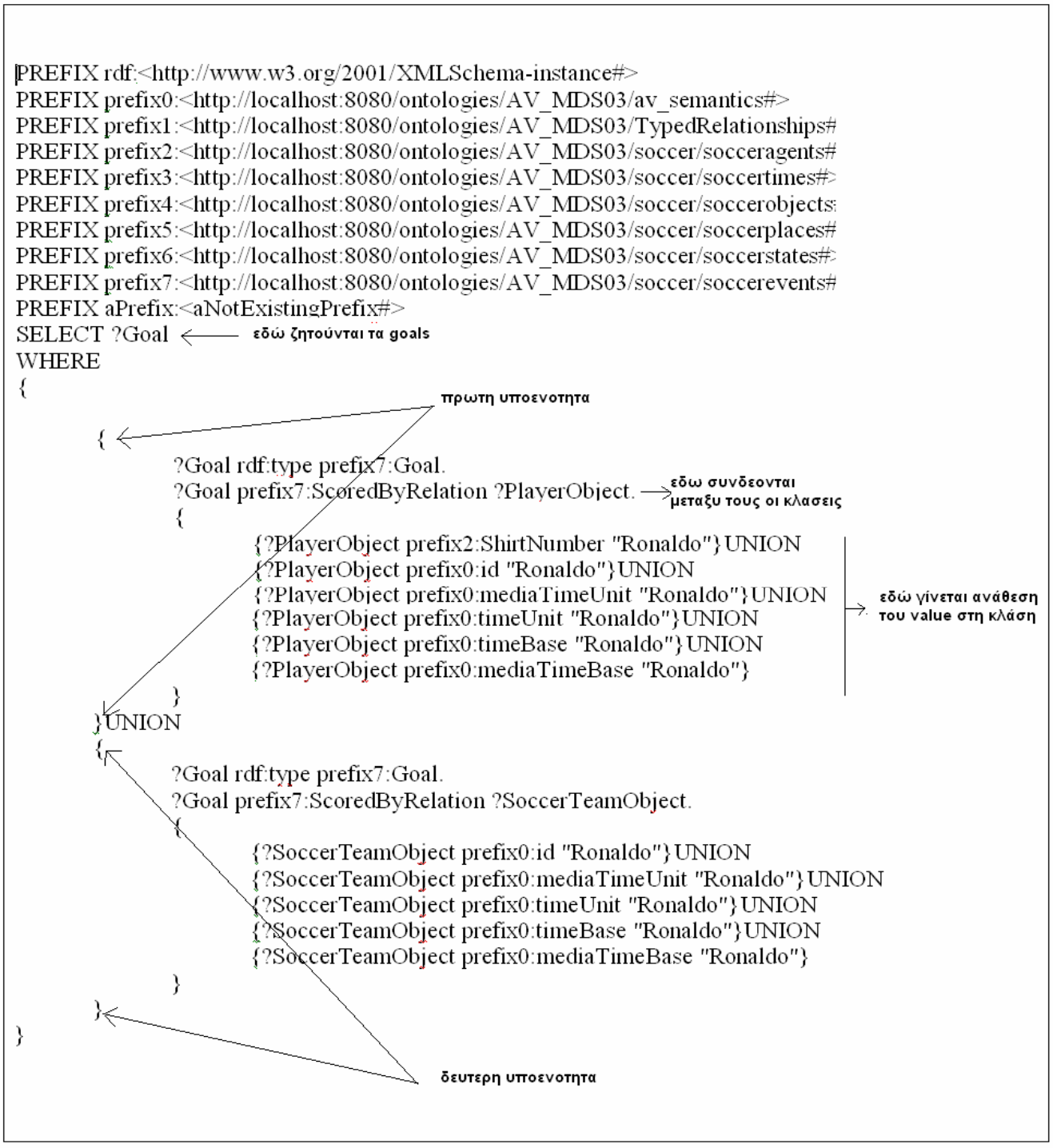
Επίσης κάποια πράγματα βασικά για τη SPARQL έτσι ώστε να γίνει καλύτερα κατανοητό το query που ακολουθεί είναι τα εξής:

Στη SPARQL ορίζονται με ερωτηματικό **(?)** οι μεταβλητές, με τέλεια **(.)** ενώνονται οι τριάδες με 'AND' και με **UNION** συνδέονται οι τριάδες με 'OR'.
Τριάδες είναι το βασικό στοιχείο της SPARQL (triples), τριάδες που το πρώτο μέρος είναι το subject , το τρίτο είναι το object και το δεύτερο είναι η σχέση η οποία ενώνει το πρώτο με το τρίτο μέρος. Στα SPARQL queries που δημιουργούνται από το **OntoNL** υπάρχει πάντα ένα SELECT κομμάτι και ένα WHERE κομμάτι. Εκτός από αυτά υπάρχει και το κομμάτι το οποίο αποτελείται από PREFIX, συντομογραφίες δηλαδή URIs που κάνουν την δομή του SPARQL πιο "μαζεμένη" και καλύτερα δομημένη.

Στο SELECT κομμάτι τοποθετούνται οι μεταβλητές εκείνες για τις οποίες θέλουμε να δούμε τις τιμές τους μετά το τέλος του query. Δηλαδή ουσιαστικά τοποθετούνται αυτά που επιθυμεί ο χρήστης να δει ως αποτελέσματα από το query. Στο WHERE κομμάτι τοποθετούνται όλες οι απαραίτητες σχέσεις (σε μορφή τριάδων) που οδηγούν τις μεταβλητές που ζητούνται στο SELECT κομμάτι στα σωστά αποτελέσματα , αφού ουσιαστικά είναι το κομμάτι που γίνονται οι αναθέσεις στις μεταβλητές μέσω των κατάλληλων σχέσεων.

Λόγω του μεγάλου όγκου πληροφορίας που δημιουργείται στο query, στο παρακάτω παράδειγμα που ακολουθεί και για λόγους ευανάγνωσης, τα values των κλάσεων αντιστοιχίζονται μόνο στα data properties της ίδιας κλάσης και όχι και στων γειτονικών κλάσεων.

Έτσι το SPARQL query που δημιουργείται είναι το εξής :



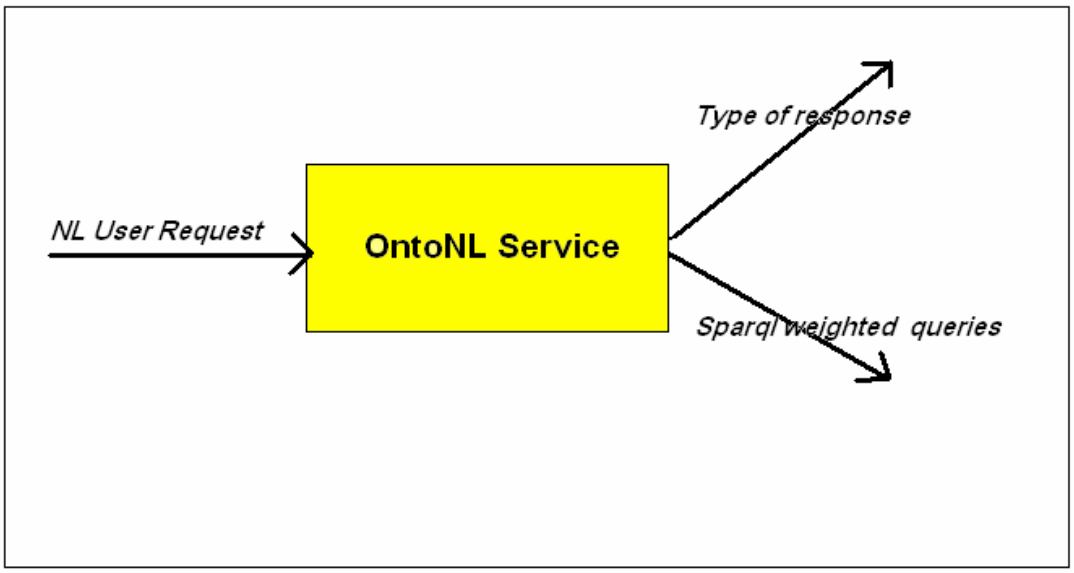
Εικόνα 56: Παράδειγμα δημιουργίας Sparql query

Παρατηρείται λοιπόν ποια είναι η δομή του query που δημιουργήθηκε στο συγκεκριμένο παράδειγμα. Σε αυτή γίνεται φανερό τι ζητείται (goals) κοιτώντας στο SELECT κομμάτι του query, επίσης παρατηρείται ότι υπάρχουν δύο κύριες υποενότητες (goals of player Ronaldo, goals of soccerTeam Ronaldo) που ενώνονται μεταξύ τους με UNION (OR), πως γίνονται οι μεταβάσεις από κλάση σε κλάση και

από κλάση σε value σε κάθε μία από αυτές τις υποενότητες και πως αυτές συνδέονται μεταξύ τους. Αλλού χρειάζεται να ενωθούν με το λογικό ΚΑΙ **(.)** και αλλού με το λογικό Ή **(UNION)**, ανάλογα την περίπτωση που προκύπτει από την εκάστοτε δομή.

5.6 Το ONTONL Service

Στα πλαίσια της παρούσας διπλωματικής εργασίας, δημιουργήθηκε η ανάγκη χρησιμοποίησης της λειτουργικότητας που παρέχει η υλοποίηση του OntoNL πλαισίου από διάφορες εφαρμογές οι οποίες θα παρουσιαστούν στο επόμενο κεφάλαιο. Απόρροια αυτής της ανάγκης ήταν η δημιουργία μιας συνάρτησης η οποία παρέχεται στους χρήστες που επιθυμούν να τη χρησιμοποιήσουν, μέσω της τεχνολογίας των web services, και πιο συγκεκριμένα με τη τεχνολογία Axis Web Services. Έτσι λοιπόν υλοποιήθηκε το **OntoNL Service,** το οποίο είναι μια συνάρτηση που ως είσοδο έχει το αίτημα του χρήστη σε φυσικό λόγο και ως έξοδο τα sparql queries που δημιουργούνται καθώς και το τύπο απάντησης που παρέχεται (ενότητα 4.7 NL Query API).



Εικόνα 57:Το OntoNL Service

5.7 Ανακεφαλαίωση

Συνοψίζοντας λοιπόν, στο συγκεκριμένο κεφάλαιο αναλύσαμε τα πλαίσια – στάδια υλοποίησης **OntoNL**, ποιες δομές χρησιμοποιούνται και πως επικοινωνούνε μεταξύ τους. Στο επόμενο κεφάλαιο θα εξετάσουμε τη χρησιμότητα – εφαρμογή του **OntoNL** σε ένα ολοκληρωμένο σενάριο στο οποίο γίνεται χρήση του MPEG-7 Semantic Repository για το domain του ποδοσφαίρου. Θα εξετάσουμε την

αρχιτεκτονική του, τα στάδια υλοποίησης καθώς και διάφορες ιδιαιτερότητες που αντιμετωπίσαμε

ΚΕΦΑΛΑΙΟ 6

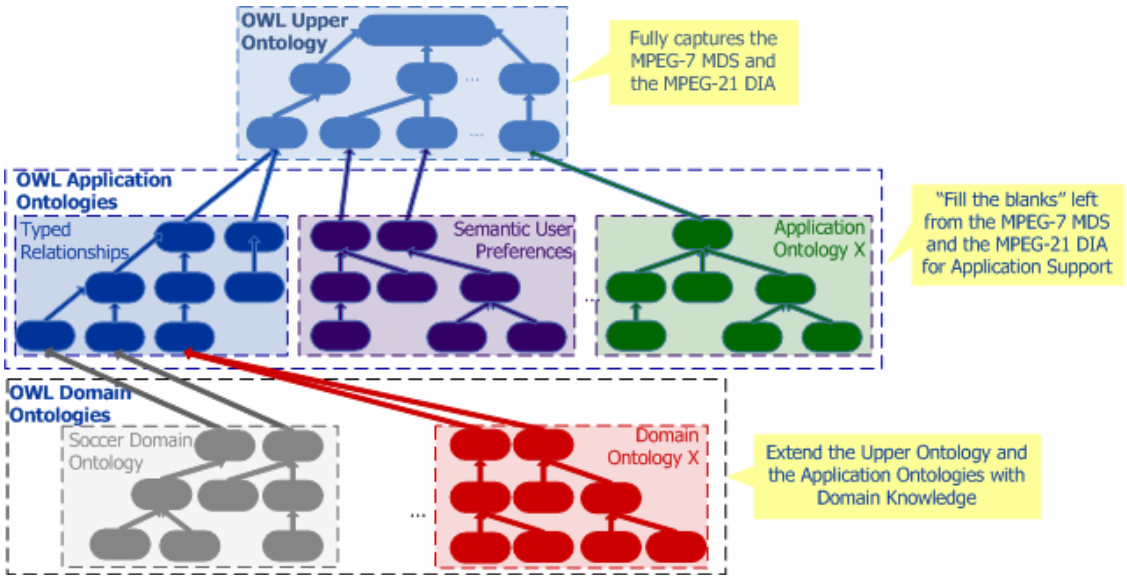
ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ ONTONL FRAMEWORK ΣΕ MPEG-7 SEMANTIC REPOSITORY ΓΙΑ ΤΟ DOMAIN ΤΟΥ ΠΟΔΟΣΦΑΙΡΟΥ

6.1 Εισαγωγή

Στο προηγούμενο κεφάλαιο εξετάστηκε αναλυτικά η υλοποίηση του **OntoNL** framework, τα στάδια επεξεργασίας που υλοποιούνται, ποιες δομές αυτό δημιουργεί και τελικά το SPARQL query που εξάγεται. Στο συγκεκριμένο κεφάλαιο θα εξεταστεί ένα ολοκληρωμένο σενάριο υλοποίησης της εφαρμογής του **OntoNL** σε ένα Semantic Repository για το domain του ποδοσφαίρου. Θα εξεταστούν οι ιδιαιτερότητες που αντιμετωπίστηκαν στη συγκεκριμένη οντολογία, η αρχιτεκτονική του συστήματος **Sparql2Mp7ql Translator**, ένα σύστημα που μετατρέπει τη SPARQL σε MP7QL, καθώς και το ολοκληρωμένο σενάριο ανάκτησης αποτελεσμάτων από το Repository

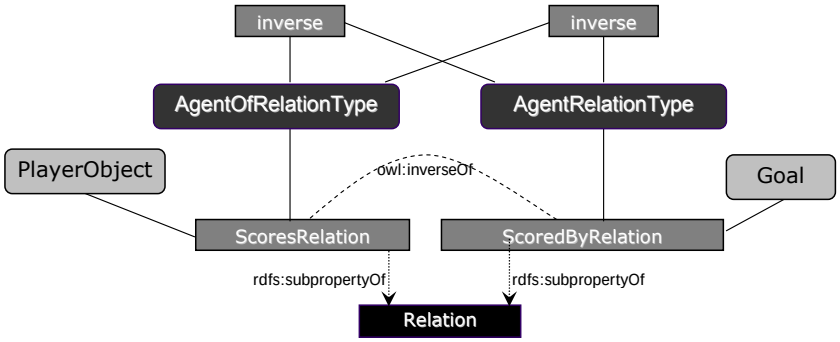
6.2 Η οντολογία ποδοσφαίρου και οι ιδιαιτερότητες της

Η OWL οντολογία η οποία χρησιμοποιήθηκε στην εφαρμογή, είναι μια domain οντολογία η οποία υπακούει στο DS-MIRF ontological infrastructure [10] . Συγκεκριμένα η ιεραρχία φαίνεται στη παρακάτω εικόνα:



Εικόνα 58: Η ιεραρχία στο DS-Mirf ontological infrastracture

Είναι φανερό ότι η soccer ontology είναι μια domain οντολογία, που σημαίνει ότι κληρονομεί όλα τα χαρακτηριστικά των upper οντολογιών και εμπλουτίζεται με χαρακτηριστικά του τομέα τον οποίο περιγράφει. Η ιδιαιτερότητα που παρουσιάστηκε με τη παραπάνω οντολογία ήταν η εξής. Οι κλάσεις της οντολογίας όπως όλες οι κλάσεις των OWL οντολογιών έχουν κάποια object properties τα οποία απευθύνονται σε κάποιες άλλες κλάσεις (range). Η ιδιαιτερότητα είναι ότι στη συγκεκριμένη οντολογία για να εντοπιστεί σε ποια κλάση απευθύνεται το εκάστοτε object property πρέπει να επιλεχθεί το inverse property του συγκεκριμένου object property και να βρεθεί η domain κλάση στην οποία αυτό απευθύνεται.

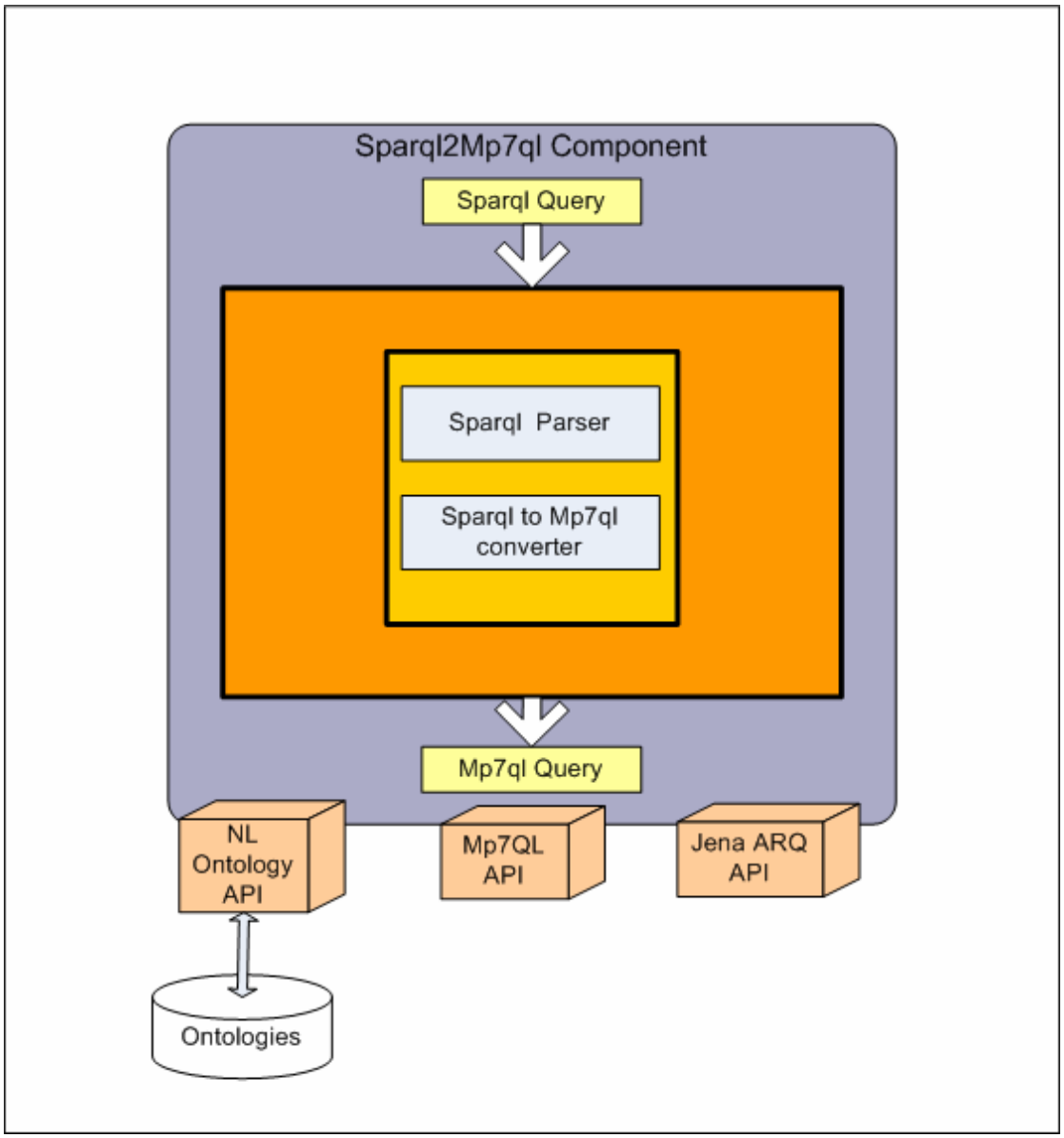


Εικόνα 59: Παράδειγμα συσχέτισης κλάσεων στην οντολογία του ποδοσφαίρου

Παρατηρείται δηλαδή ότι ενώ ουσιαστικά ένα 'PlayerObject Scores Goal' και ένα 'Goal ScoredBy PlayerObject' αυτή η μετάβασή δεν είναι δυνατή παρά μόνο αν βρεθεί η domain κλάση του inverse property του αντίστοιχου object property που μας ενδιαφέρει. Αποτέλεσμα αυτής της ιδιαιτερότητας είναι στη συγκεκριμένη εφαρμογή στο τομέα του ποδοσφαίρου, όταν χρειάζεται να βρούμε πως συνδέονται οι κλάσεις μεταξύ τους μέσω object properties να ακολουθείται η παραπάνω μεθοδολογία.

6.3 Η αρχιτεκτονική του Sparql2Mp7Ql Translator

Όπως έχει ήδη αναφερθεί, στην υλοποίηση του OntoNL σε Mpeg7 Semantic Repository για το τομέα του ποδοσφαίρου χρειάζεται να «μεταφραστεί» το SPARQL query που έχει προκύψει από το **OntoNL Component** σε Mp7Ql, έτσι ώστε να απευθυνθεί το νέο query στο repository για να ανακτηθούν αποτελέσματα. Το **Sparql2Mp7Ql Translator Component** αναλαμβάνει τη διεκπεραίωση αυτής της μετατροπής και η αρχιτεκτονική του συγκεκριμένου συστήματος φαίνεται παρακάτω:



Εικόνα 60: Η αρχιτεκτονική του Sparql2Mp7ql Translator

Όπως φαίνεται στο παραπάνω σχήμα, το **Sparql2Mpeg7ql component** αποτελείται κυρίως από 2 υποενότητες , το **Sparql Parser** και το **Sparql to Mp7ql converter**. Επίσης χρησιμοποιούνται κάποια Java APIs τα οποία είναι απαραίτητα στην υλοποίηση της παραπάνω αρχιτεκτονικής και είναι τα εξής:

- Ένα **ontology API** το οποίο επικοινωνεί με την εκάστοτε domain οντολογία και χρησιμοποιείται γιατί το σύστημα χρειάζεται πληροφορία που σχετίζεται με την οντολογία και δε μπορεί να ανακτηθεί από το Sparql query. Λεπτομέρειες θα αναλυθούν παρακάτω
- Το **Mp7QL API** το οποίο είναι ένα API που βοηθά να δημιουργηθεί το Mp7QL query, αφού προσφέρει συναρτήσεις οι οποίες παρέχουν

συγκεκριμένες λειτουργίες και αφορούν την αυτόματη δημιουργία των queries.

- το **Jena ARQ API** το οποίο βοηθά να επεξεργαστεί από το σύστημα το Sparql query που υπάρχει στην είσοδο του, να γίνει parse και να μπορεί να αναλυθεί η δομή του

Έτσι λοιπόν, υπάρχει ένα query εκφρασμένο σε SPARQL και τελικά δημιουργείται το αντίστοιχο query εκφρασμένο σε Mp7ql. Πιο συγκεκριμένα αυτό το Sparql query περνάει από 2 στάδια επεξεργασίας .

Αρχικά, στην υποενότητα **Sparql Parser** (εικ.60), γίνεται parse το Sparql query που βρίσκεται στην είσοδο του συστήματος. Με αυτό το τρόπο ελέγχετε αν το συγκεκριμένο query είναι έγκυρο και επίσης δημιουργείτε ένα query Object το οποίο είναι μέσω συναρτήσεων που παρέχονται από το Jena ARQ API άμεσα προσπελάσιμο. Έτσι λοιπόν σε αυτό το στάδιο ελέγχετε η εγκυρότητα του query έτσι ώστε στο επόμενο στάδιο να αρχίσει η μετατροπή του σε Mp7ql, αφού είναι εφικτό μέσω συναρτήσεων να διαβαστεί και να αναλυθεί η δομή του.

Όσον αφορά τη μετατροπή σε MP7QL, ακολουθεί μια σύντομη εισαγωγή σε βασικές έννοιες που καθιστούν τη κατανόηση της διαδικασίας ευκολότερη. Τα **Σημασιολογικά Εργαλεία** του MPEG-7 επιτρέπουν την περιγραφή της σημασίας του οπτικοακουστικού περιεχομένου. Έτσι, υποστηρίζεται ο ορισμός αφηρημένων (abstract) και απτών (concrete) σημασιολογικών οντοτήτων που μπορεί να είναι Γεγονότα (Events), Αντικείμενα, Έννοιες, Τόποι (Places), Χρονικές Έννοιες και Παράγοντες (Agents) που περιλαμβάνουν Πρόσωπα (Persons), Ομάδες Προσώπων και Οργανισμούς (Organizations). Τα σχήματα περιγραφής για την αναπαράσταση των εννοιών αυτών προκύπτουν από το σχήμα περιγραφής **SemanticBase DS**. Το επίπεδο αφαίρεσης (abstraction level) μιας σημασιολογικής οντότητας καθορίζεται από το στοιχείο **AbstractionLevel**, που ορίζεται στο σχήμα περιγραφής SemanticBase DS και διαθέτει ένα γνώρισμα, με Dimension, με τύπο μη αρνητικό ακέραιο. Όταν το AbstractionLevel απουσιάζει από μια σημασιολογική περιγραφή, η περιγραφή αναφέρεται σε συγκεκριμένο οπτικοακουστικό υλικό. Όταν AbstractionLevel.Dime-nsion=0, υπάρχει η περιγραφή μιας συγκεκριμένης σημασιολογικής οντότητας (π.χ. ο ποδοσφαιριστής Zidane) που σχετίζεται με κάθε οπτικοακουστικό τμήμα όπου εμφανίζεται η οντότητα. Όταν το AbstractionLevel.Dimension έχει μη μηδενική τιμή, η σημασιολογική οντότητα είναι

αφηρημένη και αναπαριστά μια κλάση (π.χ. η κλάση "PlayerObject" που αναπαριστά τους ποδοσφαιριστές). Οι σημασιολογικές οντότητες μπορεί να συσχετίζονται μέσω σχέσεων. Το MPEG-7 MDS διαθέτει πάνω από 100 πρότυπους τύπους σχέσεων που οργανώνονται σε 5 σχήματα κατηγοριοποίησης (Classification Schemes): **(α)** Το σχήμα κατηγοριοποίησης **BaseRelation CS**, που περιέχει βασικούς τύπους σχέσεων (equals, inside, refines κ.α.) **(β)** Το σχήμα κατηγοριοποίησης **GraphRelation CS** που περιέχει τύπους σχέσεων που υπάρχουν μεταξύ κόμβων γράφων (identity, equivalent κ.α.) **(γ)** Το σχήμα κατηγοριοποίησης **SpatialRelation CS**, που περιέχει τύπους χωρικών σχέσεων (over, below, north κ.α.) **(δ)** Το σχήμα κατηγοριοποίησης **TemporalRelation CS** , που περιέχει τύπους χρονικών σχέσεων (precedes, overlaps, contains κ.α.) **(ε)** Το σχήμα κατηγοριοποίησης **SemanticRelation CS**, που περιέχει τύπους σημασιολογικών σχέσεων (shows, agent, causer κ.α.).

Στην υποενότητα **Sparql to Mp7ql converter** (εικ.60), όπως φαίνεται και από τη συγκεκριμένη ονοματολογία, έχει αναπτυχθεί μεθοδολογία η οποία μετατρέπει το Sparql query στο αντίστοιχο Mp7ql. Η διαδικασία που ακολουθείται σε γενικές γραμμές είναι να διαβαστούν σταδιακά οι υποενότητες που αποτελούν το Sparql query και να δημιουργηθούν οι αντίστοιχες σε Mp7ql.

Συγκεκριμένα για κάθε υποενότητα στο Sparql query δημιουργείτε ένα SemanticPreference στο Mp7ql query, στο οποίο θα αντιστοιχηθεί η ανάλογη πληροφορία που περιέχεται στην συγκεκριμένη υποενότητα, αφού σε αυτό το element (SemanticPreference) αναφέρονται οι σημασιολογικές προτιμήσεις του χρήστη . Ανάλογα με το τρόπο που συνδέονται οι υποενότητες μεταξύ τους (OR ή AND) το ίδιο ισχύει και στο αντίστοιχο κομμάτι του Mp7ql στο οποίο ορίζεται το attribute booleanOperator να είναι OR ή AND έτσι ώστε τα αντίστοιχα Semantic Preferences να συνδέονται με τον ίδιο τρόπο.

Στη συνέχεια, για τη κάθε υποενότητα που έχει διαβαστεί από το Sparql query, αρχίζει ένα στάδιο ανάλυσης τριάδα προς τριάδα και για κάθε τριάδα δημιουργείται αντίστοιχα ένα SemanticBase στο οποίο αναπαρίσταται η σημασιολογική βάση της πληροφορίας, δηλαδή ορίζονται οι διάφορες οντότητες και η σχέση μεταξύ τους, και το οποίο τοποθετείται στο εκάστοτε SemanticPreference. Για κάθε ένα από τα elements (SemanticBase) που δημιουργούνται, ανάλογα με την περίπτωση, αναθέτονται και διαφορετικές τιμές στα attributes που του αντιστοιχούνε. Επίσης σε

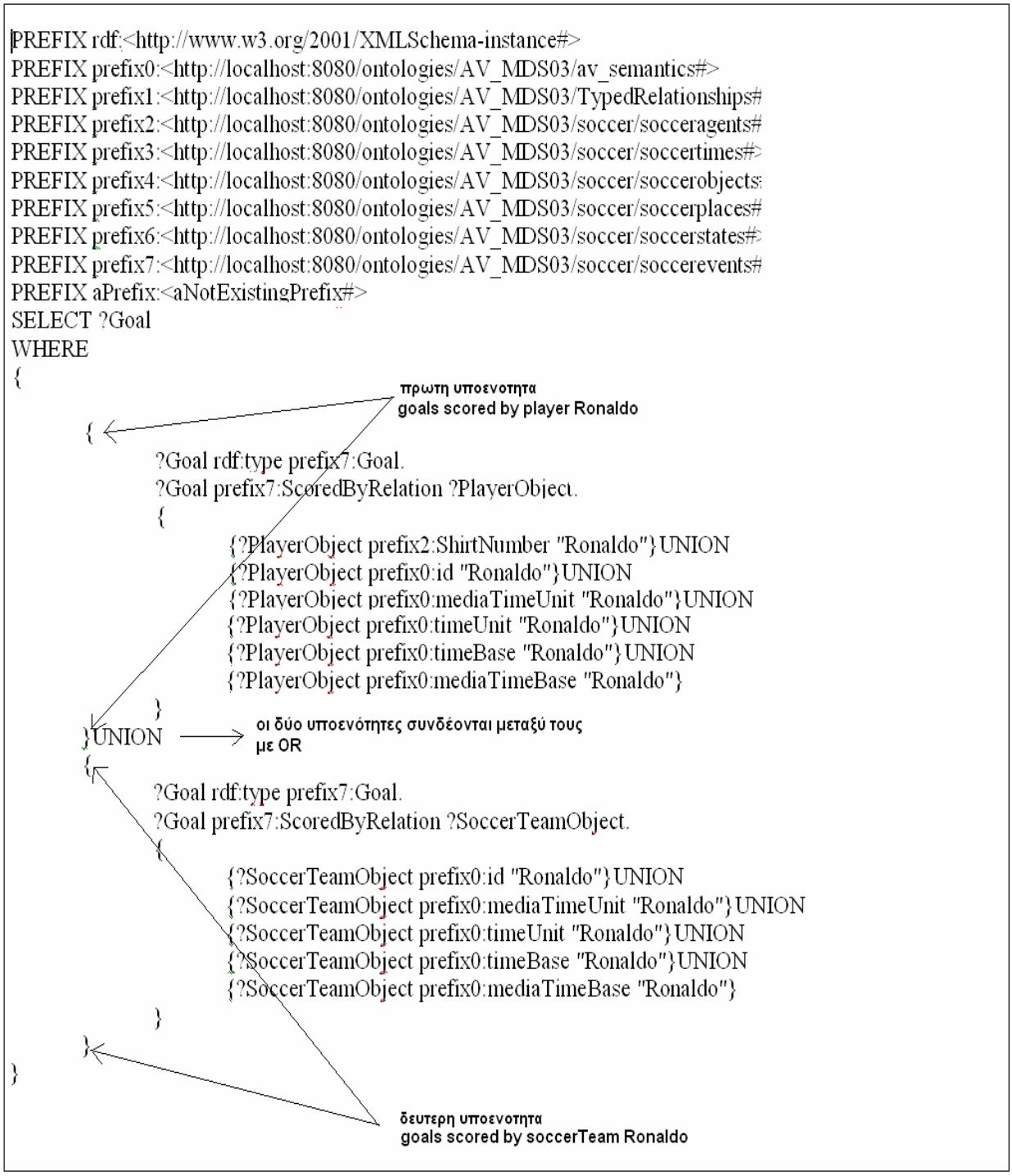
κάθε SemanticBase μπορεί να εμφανίζονται ένα ή περισσότερα Relation ή Label ανάλογα με τη περίπτωση.

```
Program String mp7qlBuilder (String sparqlQuery)
Mp7qlQueryObject mp7qlObject;
Begin
   QueryObject spObj=parse(sparqlQuery);

   List elements=spObj.getElements();
   For all terms i of elements{
      QueryElement element=elements[i];
      SemanticPreference sp;

      If(element.getOperand().equals("OR"))
         sp.setBooleanOperator("OR");
      Else if(element.getOperand().equals("AND"))
         sp.setBooleanOperator("AND");

      List triples=element.lisTriples();
      For all terms j of triples{
         Triple triple=triples[j];
         SemanticBase sbSub;
         String subject=triple.getSubject();
         Relation relSub;
         relSub.setType(getType(triple));
         relSub.setTarget(subject);
         relSub.setID("?"+subject);
         sbSub.add(relSub);

         String object=triple.getObject();
         Relation relObj;
         SemanticBase sbObj;
         relObj.setType(getType(triple));
         relObj.setTarget(object);
         relObj.setID("?"+object);
         sbObj.add(relObj);
         If(triple.getObject().isValue()){
              sbObj.addLabel().addName(
                             triple.getObject().getValue());
         }

         SemanticBase sbPred;
         String predicate=triple.Predicate();
         Relation relPred;
         relPred.setType(getType(triple));
         relPred.setTarget(subject);
         relPred.setSource(object);
```

```
          sbPred.add(relPred);

          sp.add(sbSub);
          sp.add(sbPred);
          sp.add(sbObj);
        }
       mp7qlObject.add(sp);
     }
      return mp7qlObject.toString();
End
```

Γενικότερα αυτή είναι η μεθοδολογία που ακολουθήσαμε, δηλαδή να αντιστοιχίζονται οι δομές του Sparql σε αντίστοιχες Mp7ql ξεκινώντας από τις πιο γενικές και φτάνοντας στις πιο ειδικές, δηλαδή από το πιο ευρύ scope στο πιο συγκεκριμένο. Η παραπάνω διαδικασία θα γίνει πιο κατανοητή με το παράδειγμα που ακολουθεί, στο οποίο σχολιάζεται η διαδικασία που ακολουθήθηκε για να μετατραπεί ένα Sparql query σε Mp7ql.

Για τη πρόταση ‘Give me goals scored by Ronaldo’ όπως αναλύθηκε στην ενότητα 5.5 παίρνουμε το παρακάτω Sparql query:



Εικόνα 61: Παράδειγμα Sparql Query

Μετά την επεξεργασία από το Sparql2Mp7ql Component παίρνουμε το παρακάτω Mpeg7ql query:

Εικόνα 62:Παράδειγμα Mp7ql query

Στο παραπάνω query υπάρχει ένα root element με το όνομα Mpeg7Query. Εμπεριέχεται το element QuerySpecification στο οποίο θα οριστούν τα κριτήρια του query.

Στο συγκεκριμένο QuerySpecification υπάρχει ένα attribute με το όνομα booleanOperator στο οποίο γίνεται ανάθεση της τιμής OR που σημαίνει ότι τα SemanticPreference που αυτό περιέχει ενώνονται μεταξύ τους με OR, κάτι που είναι λογικό γιατί παρατηρείται ότι στο Sparql query υπάρχουν δύο υποενότητες που ενώνονται με UNION (OR), έτσι λοιπόν και στο Mpeg7ql query τα δύο SemanticPreference που αντιπροσωπεύουν αυτές τις υποενότητες πρέπει να συνδέονται με τον ίδιο τρόπο.

Σε κάθε SemanticPreference τοποθετούνται τόσα SemanticBase όσα είναι ικανά να περιγράψουν αυτά που αναφέρουν οι αντίστοιχες τριάδες του Sparql query. Επίσης στο SemanticPreference γίνεται ανάθεση στο attribute booleanOperator της τιμής AND αφού στόχος είναι τα SemanticBase που αυτό περιέχει να συνδέονται μεταξύ τους με τον operator AND αφού κάτι ανάλογο συμβαίνει και στο Sparql query.

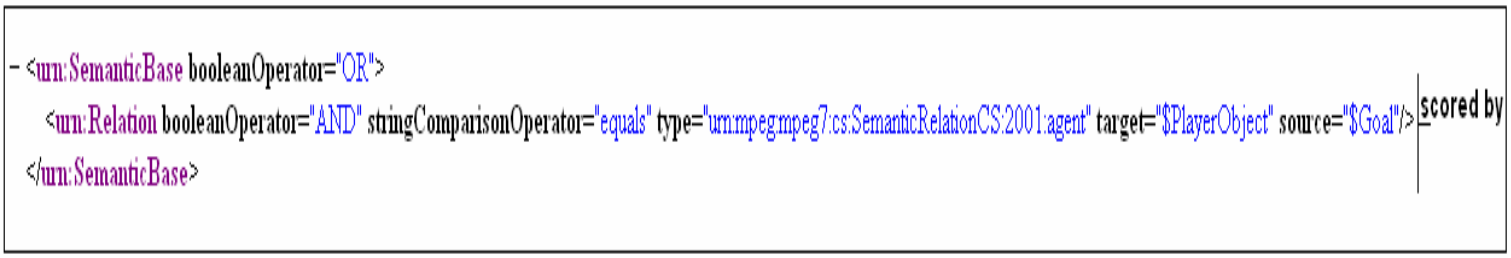
Εικόνα 63:Παράδειγμα SemanticPreference

Σε κάθε SemanticBase ορίζονται διάφορες οντότητες ή συνδέονται μεταξύ τους ανάλογα με τη περίπτωση. Ιδιαίτερη χρηστικότητα σε αυτή τη δομή έχει το attribute id που ουσιαστικά προσδιορίζει το συγκεκριμένο element και το καθιστά επαναχρησιμοποιήσιμο απλά κάνοντας αναφορά στο συγκεκριμένο id. Αν πρέπει σε κάποιο SemanticBase να γίνει αναφορά σε μία κλάση, δημιουργείται ένα Relation στο οποίο ορίζονται τα attributes type και target, στα οποία αναθέτονται τιμές που προέρχονται από την οντολογία και προέκυψαν από συγκεκριμένη μεθοδολογία έτσι

ώστε να γίνεται κατανοητό ποια είναι η διαδρομή που πρέπει να ακολουθηθεί για να γίνει αναφορά στη συγκεκριμένη κλάση.

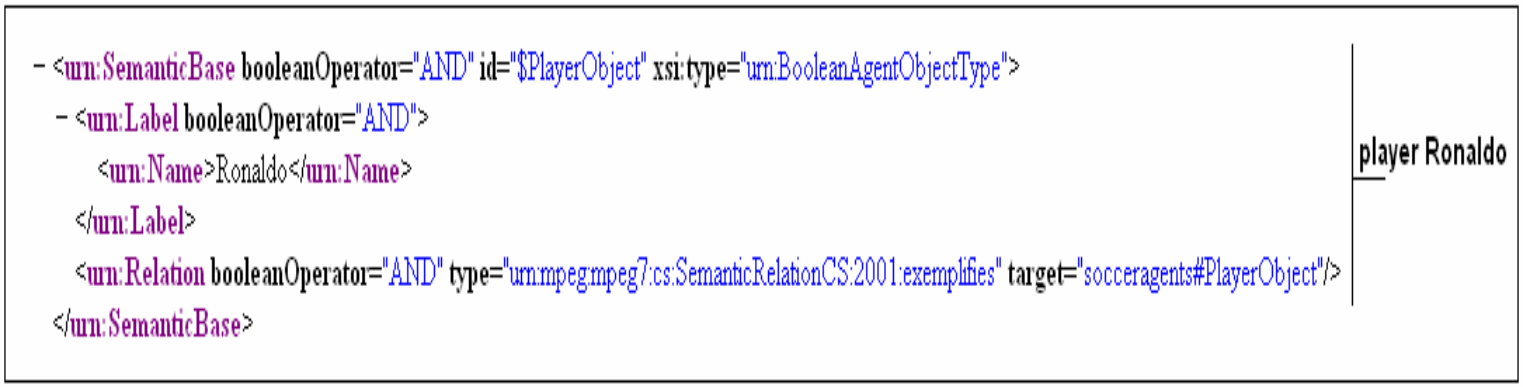
Εικόνα 64:Παράδειγμα SemanticBase με ανάθεση κλάσης

Επίσης αν πρέπει σε ένα SemanticBase να ενωθούν δύο κλάσεις μέσω κάποιου object property τότε ορίζεται στο συγκεκριμένο element ένα Relation στο οποίο φυσικά αναθέτονται τα attributes target και source να δείχνουν σε αυτές τις κλάσεις μέσω των id όπως ήδη έχει αναφερθεί και επίσης ορίζεται το type που υποδηλώνει τι τύπου είναι το object property στο οποίο γίνεται η αναφορά.



Εικόνα 65:Παράδειγμα SemanticBase με σύνδεση κλάσεων

Τέλος, αν γίνεται ανάθεση μίας τιμής σε κάποια κλάση, δημιουργείται ένα SemanticBase στο οποίο υπάρχει ένα Label με Name τη τιμή που θα ανατεθεί και επίσης υπάρχει ένα Relation στο οποίο γίνεται η αναφορά στην κλάση στην οποία γίνεται η ανάθεση.



Εικόνα 66:Παράδειγμα Semantic Base με ανάθεση κλάσης και τιμής

Έτσι λοιπόν με τη παραπάνω μεθοδολογία επιτυγχάνετε η μετατροπή από Sparql σε Mp7ql αφού για κάθε δομή που υπήρχε στη Sparql δημιουργήθηκε η αντίστοιχη σε Mp7ql και επίσης έγινε ανάθεση των κατάλληλων τιμών που έπρεπε να οριστούνε όπου αυτό ήταν απαραίτητο.

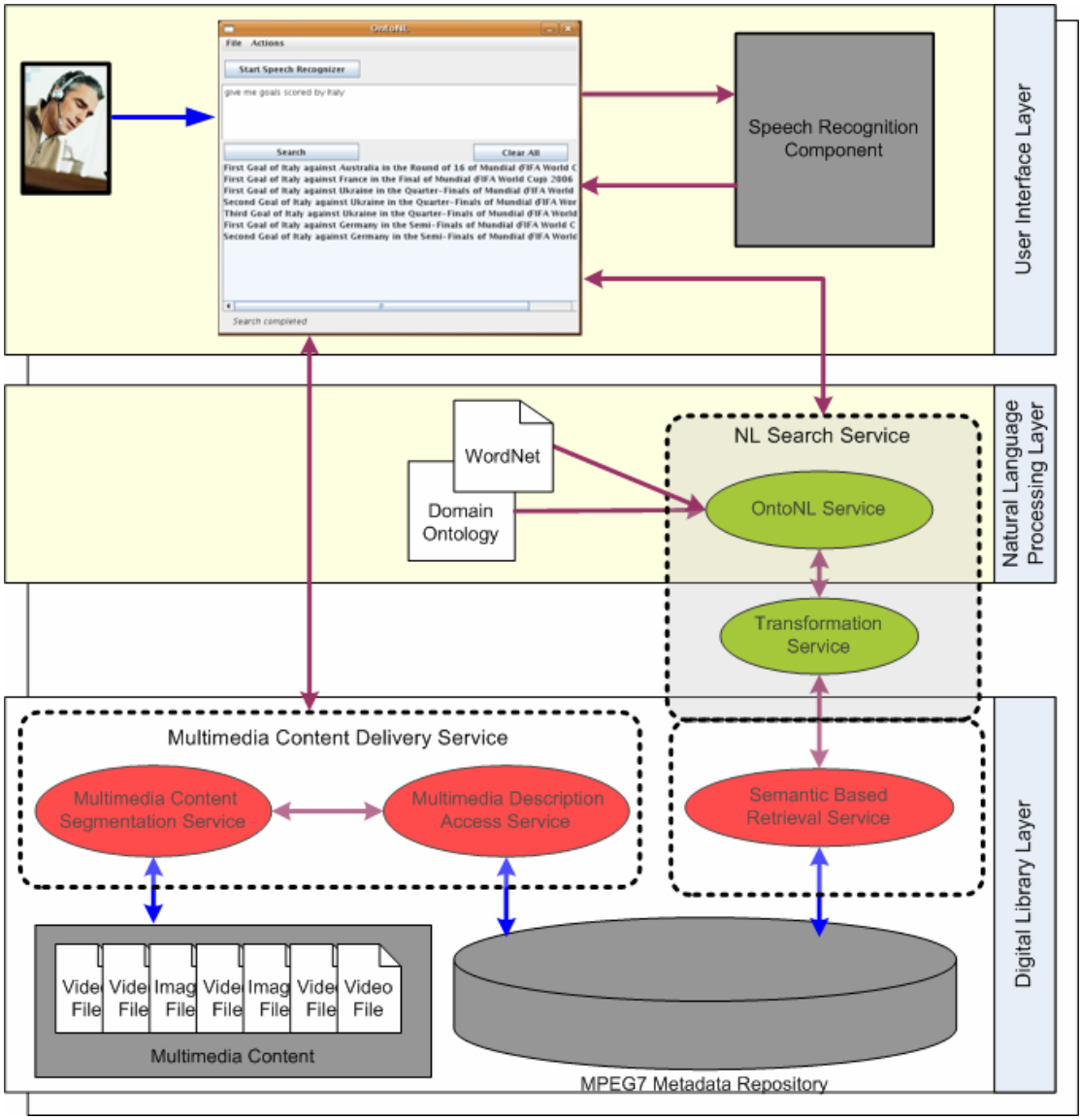
6.4 Το ολοκληρωμένο σενάριο NL2DSMIRF

Στο σενάριο που θα ακολουθήσει γίνεται χρήση των **OntoNL Service** και **Transformation Service** που υλοποιήθηκαν κατά την εκπόνηση της συγκεκριμένης διπλωματικής εργασίας σύμφωνα με την τεχνολογία Axis Web Service που αναλύθηκε στο κεφάλαιο 3.

Πιο συγκεκριμένα το **OntoNL Service** επιτελεί όλη τη λειτουργία που αναλύθηκε στο κεφάλαιο 4 για το **OntoNL Component**. Δηλαδή είναι μία συνάρτηση η οποία έχει ως είσοδο το αίτημα του χρήστη στο φυσικό λόγο και ως έξοδο έχει το Sparql query που δημιουργείται με ότι αυτό συνεπάγεται.

Το **Transformation Service** αντίστοιχα επιτελεί όλη τη λειτουργία που αναλύθηκε στο παρών κεφάλαιο και αφορά το **Sparql2Mp7Ql Translator** (εικ. 60) και είναι μία συνάρτηση η οποία έχει ως είσοδο ένα Sparql query και ως έξοδο το αντίστοιχο query σε Mp7ql.

Έτσι λοιπόν, αυτά τα 2 services, συνεργαζόμενα μεταξύ τους χρησιμοποιήθηκαν στο σενάριο που ακολουθεί και που η αρχιτεκτονική του φαίνεται στο παρακάτω σχήμα:



Εικόνα 67: Αρχιτεκτονική ολοκληρωμένου σεναρίου NL2DSMIRF

Σε γκρι πλαίσιο και με διακεκομμένες γραμμές (NL Search Service) βρίσκονται τα συγκεκριμένα services που έχουν υλοποιηθεί κατά τη πραγματοποίηση της συγκεκριμένης διπλωματικής εργασίας και τα οποία λαμβάνουν μέρος στο στάδιο του Natural Language Processing Layer (εικ. 67). Σε γενικές γραμμές, στο σενάριο πραγματοποιούνται οι εξής ενέργειες:

Ο χρήστης, μέσω του GUI , μιλάει στο σύστημα, το Speech Recognition Component κάνει την αναγνώριση και επιστρέφει στο GUI το αίτημα του χρήστη σε φυσικό λόγο.

Στη συνέχεια ο χρήστης υποβάλει το αναγνωρισμένο μήνυμα και έτσι με αυτό το τρόπο καλείται το **NL Search Service**. Σταδιακά λοιπόν, πρώτα καλείται το **OntoNL Service** που παίρνει το αίτημα του χρήστη σε φυσικό λόγο, το επεξεργάζεται

συμβουλευόμενο και την domain οντολογία (συγκεκριμένα εδώ του ποδοσφαίρου) καθώς και το Wordnet λεξικό και τελικά επιστρέφει ένα Sparql query.

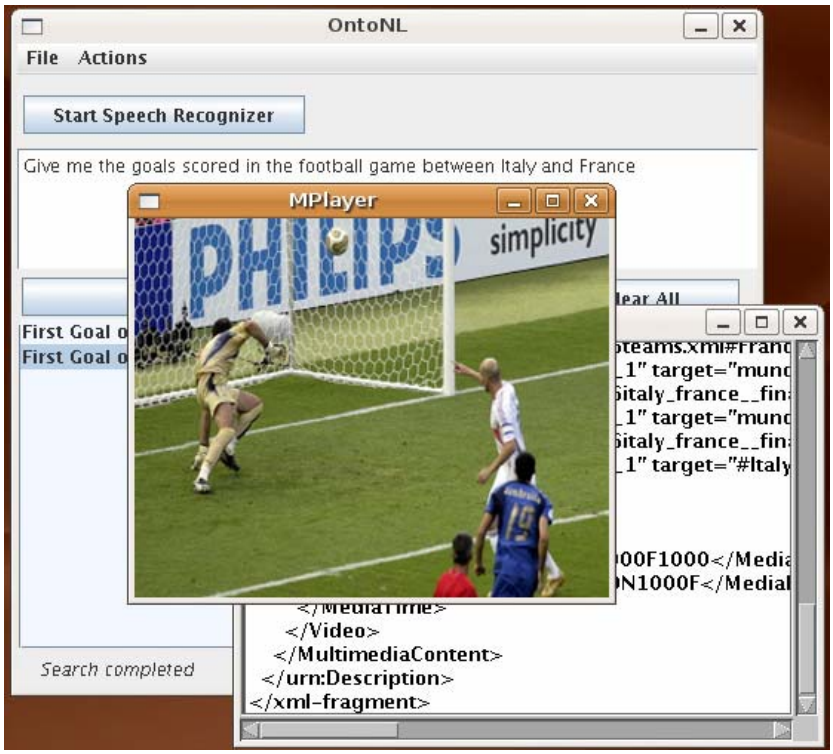
Στο επόμενο στάδιο καλείται το **Transformation Service** το οποίο παίρνει αυτό το Sparql query , το επεξεργάζεται και τελικά επιστρέφει ένα Mp7ql query. Τέλος , εκτελείται το Mp7ql query μέσω του Semantic Based Retrieval Service το οποίο επικοινωνεί με το MPEG7 Metadata Repository και επιστρέφει τα id των αναγνωρισμένων multimedia objects που πληρούν τα κριτήρια.

Στο τελικό στάδιο του σεναρίου, καλείται το Multimedia Content Delivery Service, που σε γενικές γραμμές εντοπίζει ποια multimedia content αντιστοιχούν στα id που ήδη έχουν βρεθεί και επιστρέφει ολόκληρη την MPEG7 περιγραφή, συμπεριλαμβανομένου και ενός media URI. Έτσι λοιπόν ο χρήστης βλέπει να αναπαράγεται το video segment που αντιστοιχεί στο αίτημα του.

Το παραπάνω σενάριο λοιπόν αποτέλεσε μία εφαρμογή στην οποία χρησιμοποιήθηκαν τα **OntoNL Service** και **Transformation Service**. Φυσικά η χρησιμότητα τους δε περιορίζεται μόνο σε αυτή την εφαρμογή, απλά γίνεται αναφορά ενός παραδείγματος στο οποίο εφαρμόστηκαν.

Ακολουθούνε κάποια χαρακτηριστικά screenshots της εφαρμογής:

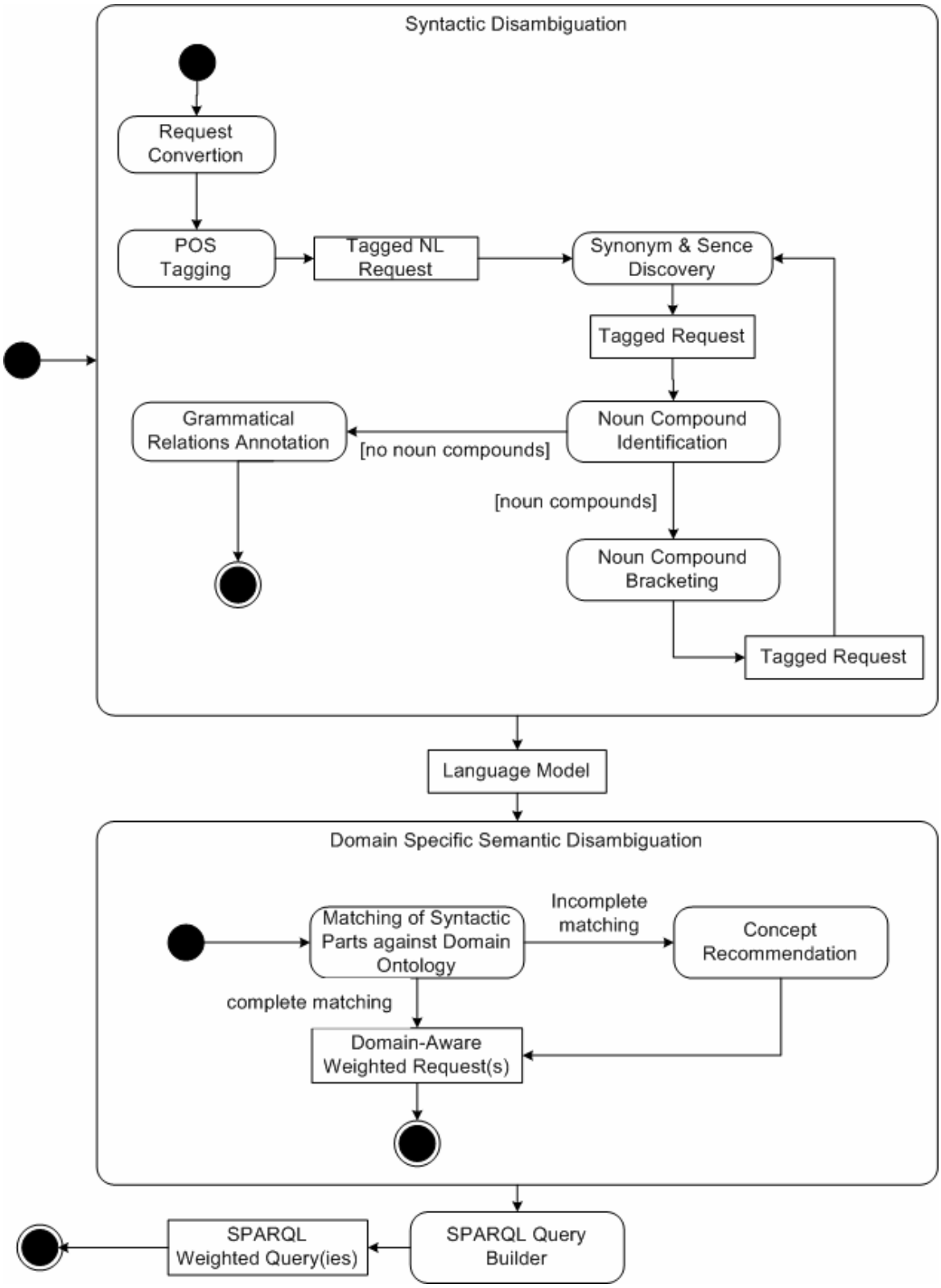
Εικόνα 68: Screenshot από την εφαρμογή του OntoNL framework με χρήση speech recognizer για την είσοδο του αιτήματος σε φυσική γλώσσα



Εικόνα 69:Το multimedia object έχει ανακτηθεί από το σύστημα και παρουσιάζεται στο χρήστη

6.5 System Flow

Στη συγκεκριμένη ενότητα θα δειχθεί η ροή πληροφορίας κατά την επεξεργασία μιας πρότασης από τη φάση που ο χρήστης έχει υποβάλει το αίτημα του ως τη φάση που δημιουργούνται τα Sparql Queries. Υπενθυμίζονται στον αναγνώστη με τη μορφή activity diagram [1] οι ενέργειες που λαμβάνουν μέρος κατά τα διάφορα στάδια επεξεργασίας



Εικόνα 70:OntoNL System Flow

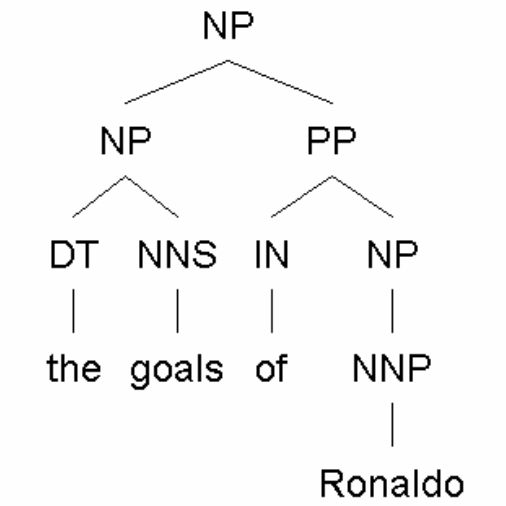
Έτσι λοιπόν ενδεικτικά για τη πρόταση 'Give me goals of Ronaldo' στα διάφορα στάδια επεξεργασίας συμβαίνουν τα εξής

Sentence :'Give me the goals of Ronaldo'

Syntactic Disambiguation

Request Convertion: the goals of Ronaldo

POS Tagging:



Tagged NL Request: (NP (NP (DT the) (NNS goals)) (PP (IN of) (NP (NNP Ronaldo))))

Synonyms and Sense Discovery:

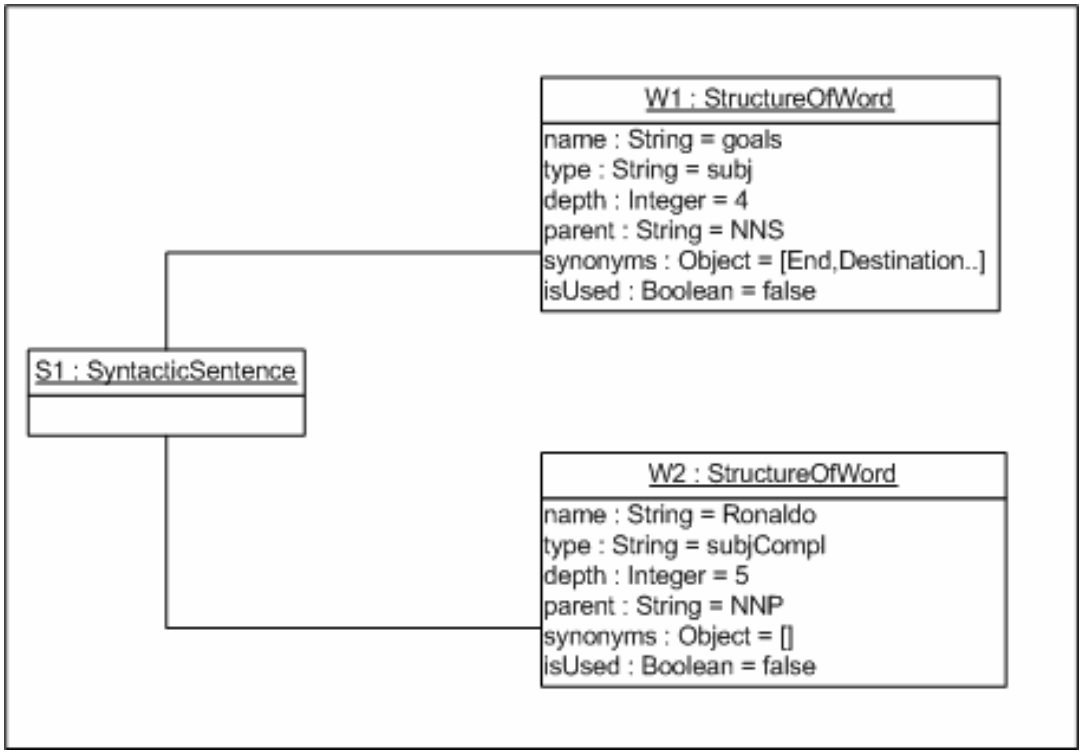
Goals:

End
Destination
Aim
Object
Objective
Target
Bourn
Intent
Basket
Hoop
Net
terminus

Ronaldo: Not Specified

Noun Compound Identification: No Noun Compounds

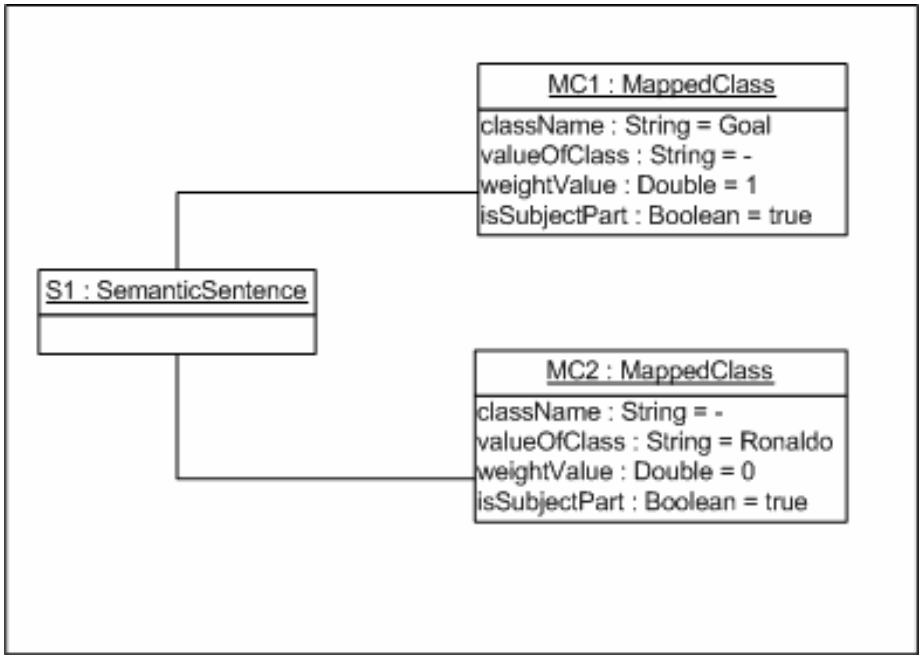
Grammatical Relation Annotation:



Export Language Model

Semantic Disambiguation

Search and match to Ontological Structures:



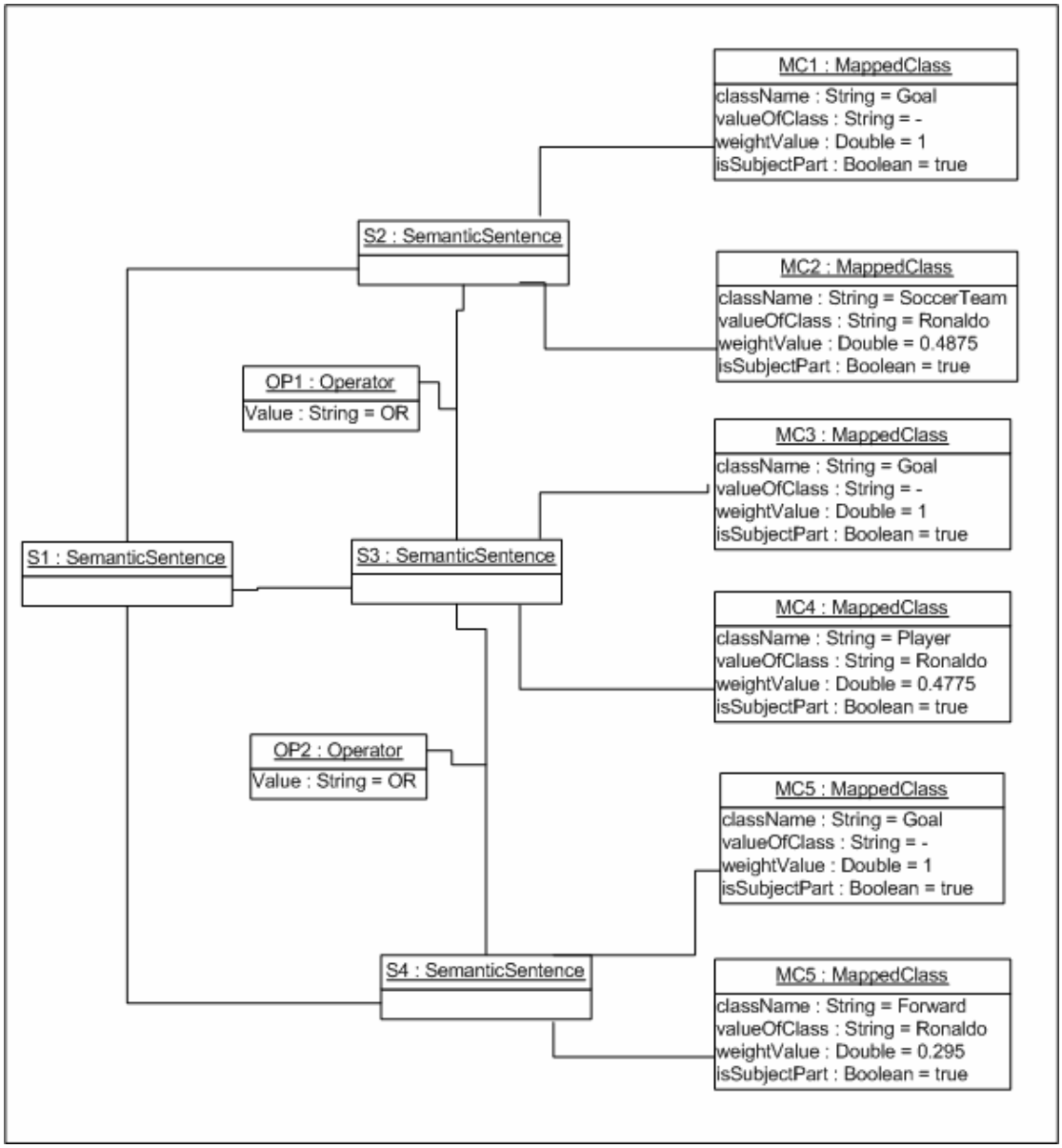
Not Complete Disambiguation of the subject part

Search for the most related concepts to the subject matched to ontological structure

(most related classes to Goal)

Forward	0.295
Player	0.4775
SoccerTeam	0.4875

Assign the semantic relatedness measurement value



Έτσι λοιπόν η αναζήτηση αφορά τα εξής:

S2: Goals scored by SoccerTeam Ronaldo
S3: Goals scored by Player Ronaldo
S4: Goals scored by Forward Ronaldo

Enhance the language model

Create sparql query: (για τη περίπτωση PlayerObject)

```
PREFIX rdf:<http://www.w3.org/2001/XMLSchema-instance#>
PREFIX prefix0:<http://localhost:8080/ontologies/AV_MDS03/av_semantics#>
PREFIX prefix1:<http://localhost:8080/ontologies/AV_MDS03/TypedRelationships#>
PREFIX prefix2:<http://localhost:8080/ontologies/AV_MDS03/soccer/socceragents#>
PREFIX prefix3:<http://localhost:8080/ontologies/AV_MDS03/soccer/soccertimes#>
PREFIX prefix4:<http://localhost:8080/ontologies/AV_MDS03/soccer/soccerobjects#>
PREFIX prefix5:<http://localhost:8080/ontologies/AV_MDS03/soccer/soccerplaces#>
PREFIX prefix6:<http://localhost:8080/ontologies/AV_MDS03/soccer/soccerstates#>
```

```
PREFIX prefix7:<http://localhost:8080/ontologies/AV_MDS03/soccer/soccerevents#>
SELECT ?Goal
WHERE
{
    {
        ?Goal rdf:type prefix7:Goal.
        {
            {?Goal prefix7:ScoredByRelation ?PlayerObject}UNION
            {?Goal prefix7:CauserRelation ?PlayerObject}UNION
            {?Goal prefix7:PerformerRelation ?PlayerObject}
        }.
        {
            {?PlayerObject prefix2:ShirtNumber "Ronaldo"}UNION
            {?PlayerObject prefix0:timeUnit "Ronaldo"}UNION
            {?PlayerObject prefix0:id "Ronaldo"}UNION
            {?PlayerObject prefix0:mediaTimeBase "Ronaldo"}UNION
            {?PlayerObject prefix0:mediaTimeUnit "Ronaldo"}UNION
            {?PlayerObject prefix0:timeBase "Ronaldo"}

        }
    }
}
```

6.6 Ανακεφαλαίωση

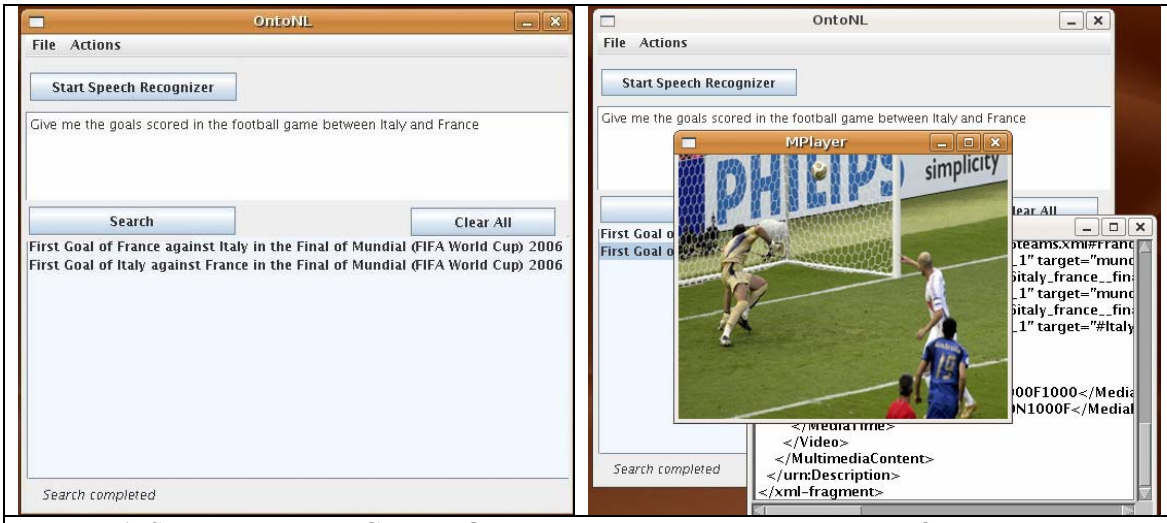
Στο συγκεκριμένο κεφάλαιο λοιπόν περιγράφηκε η υλοποίηση της εφαρμογής OntoNL Framework σε MPEG-7 Semantic Repository για το domain του ποδοσφαίρου, αναλύθηκε η αρχιτεκτονική του Sparql2Mp7Ql Translator και αναφέρθηκε ένα σενάριο στο οποίο εφαρμόζονται όλα όσα έχουν αναφερθεί. Στο επόμενο κεφάλαιο θα γίνει λόγος για τη σταθερότητα και την ευχρηστία του συστήματος.

ΚΕΦΑΛΑΙΟ 7

ΑΞΙΟΛΟΓΗΣΗ ΤΗΣ ΔΙΕΠΑΦΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΤΟΥ ONTONL ΠΛΑΙΣΙΟΥ

7.1 Αξιολόγηση ορθότητας και καταλληλότητας

Η αξιολόγηση της ορθότητας και καταλληλότητας απευθύνεται ουσιαστικά στην αξιολόγηση χρηστικότητας της διεπαφής σύμφωνα πάντα με κάποιους όρους αξιολόγησης. Το graphical user interface (GUI στο εξής) που αξιολογήθηκε είναι η εφαρμογή του **OntoNL Framework** στο τομέα του ποδοσφαίρου όπως αυτό περιγράφθηκε στο κεφάλαιο 6. Αυτή η αξιολόγηση πραγματοποιείται σύμφωνα με τα 10 μέτρα αξιολόγησης του Nielsen [50]. Τα σχόλια αξιολόγησης βασίζονται στα σχόλια μιας ομάδας αξιολόγησης τριών HCI ειδικών, των καθηγητών Tiziana Catarci, Dr. Yael Dubinsky και Dr. Stephen Kimani, από το Department of Computer and Systems Science of the University of Rome "La Sapienza"



Εικόνα 71: Screenshot από το GUI της OntoNL εφαρμογής για το τομέα του ποδοσφαίρου.

Τα χρήσιμα αυτά σχόλια που προέκυψαν από αυτό το στάδιο αξιολόγησης λήφθηκαν υπόψιν για περαιτέρω βελτίωση του GUI. .

1. Visibility of System Status

Περιγραφή: Το σύστημα πρέπει πάντα να κρατά τους χρήστες ενήμερους σχετικά με το τι συμβαίνει, μέσα σε λογικό χρονικό διάστημα.

Σχόλια:

• Στο συγκεκριμένο σύστημα, οι χρήστες πρέπει μόνο να πατήσουν ένα κουμπί έτσι ώστε να γίνει επεξεργασία του αιτήματος τους και να εμφανιστούν τα αποτελέσματα της αναζήτησης. Οι χρήστες δεν ενδιαφέρονται για τα διάφορα στάδια επεξεργασίας (parsing tree, συντακτική ανάλυση, εννοιολογική αποσαφήνιση κτλ) και έτσι αυτή η πληροφορία δεν παρουσιάζεται.

•Το σύστημα αναμένεται να παρέχει πληροφορία για τη πρόοδο της επεξεργασίας μέχρι την τελική απάντηση (progress information).

2. Match between the System and the Real World

Περιγραφή: Το σύστημα πρέπει να "μιλάει" τη γλώσσα του χρήστη, με λέξεις, φράσεις και έννοιες οικίες σε αυτόν, και όχι με τεχνικούς όρος του συστήματος. Ακολουθώντας τους real world κανόνες συμπεριφοράς, η πληροφορία παρουσιάζεται σε φυσική και λογική σειρά.

Σχόλια:

• Γενικά, οι λέξεις που χρησιμοποιούνται είναι γνωστές στο χρήστη. Οι χρήστες περιμένουν να τους παρουσιαστούν οι απαντήσεις στο δικό τους φυσικό λόγο και όχι σε κάποια XML data αναπαράσταση, έτσι το XML fragment δε θα πρέπει να εμφανίζεται.

• Οι χρήστες μπορούν να εισάγουν τα αιτήματα τους με φράσεις όπως "I would like to know…" και με πιο άμεσους τρόπους όπως how, what, when questions που είναι πολύ καλύτερη προσέγγιση όταν το σύστημα χρησιμοποιείται σε μεγάλο βαθμό.

3. User control and freedom

Περιγραφή: Οι χρήστες συχνά επιλέγουν λειτουργίες κατά λάθος και χρειάζονται ξεκάθαρα ένα "emergency exit" έτσι ώστε να φύγουν από αυτή την ανεπιθύμητη κατάσταση. Υποστήριξη undo και redo.

Σχόλια:

• Επιλογές delete, move up/down πρέπει να προστεθούν έτσι ώστε οι χρήστες να μπορούν να διαχειριστούν προηγούμενες ερωτήσεις.

• Επιλογές undo and redo πρέπει να προστεθούν έτσι ώστε οι χρήστες να μπορούν να διαχειριστούν τις λίστες με τα αποτελέσματα

4. Consistency and Standards

Περιγραφή: Οι χρήστες δε πρέπει να αναρωτιούνται αν διαφορετικές λέξεις, καταστάσεις και ενέργειες σημαίνουν το ίδιο πράγμα.

Όχι ιδιαίτερα σχόλια

5. Error prevention

Περιγραφή: Ακόμη καλύτερο από ένα μήνυμα λάθους είναι ένας προσεκτικός σχεδιασμός που αποτρέπει το λάθος από την αρχή. Μπορεί είτε να περιοριστούν οι συνθήκες που μπορεί να συμβεί κάποιο λάθος είτε να γίνεται έλεγχος και να παρουσιάζεται στο χρήστη η επιλογή επιβεβαίωσης (confirmation) έτσι ώστε να καταχωρηθεί η επιλεγμένη ενέργεια.

Σχόλια:

• Τα λάθη πρέπει να προληφθούν όταν υπάρχει μεγάλη πιθανότητα το αίτημα του χρήστη να μη μπορεί να γίνει σωστά parse.

6. Recognition rather than recall

Περιγραφή: Να ελαχιστοποιηθεί η ανάγκη ο χρήστης να χρησιμοποιεί τη μνήμη του με το να γίνονται αντικείμενα, ενέργειες και επιλογές συνέχεια ορατά. Ο χρήστης δε πρέπει να θυμάται πράγματα από τη μια κατάσταση του διαλόγου στην άλλη. Οι εντολές – επιλογές για τη χρήση του συστήματος πρέπει να είναι πάντα ορατές ή εύκολα προσπελάσιμες.

Σχόλια:

• Οι χρήστες μπορούν να διαχειριστούν το ήδη χρησιμοποιημένο question set. Με αυτό το τρόπο δε πρέπει να το θυμούνται και είναι εύκολο να το επαναχρησιμοποιήσουν.

7. Flexibility and efficiency of use

Περιγραφή: Οι accelerators – άγνωστο για τους απλούς χρήστες – συχνά επιταχύνουν την αλληλεπίδραση για τους expert χρήστες.

Σχόλια:

• Παρέχονται shortcuts στους έμπειρους χρήστες.

8. Aesthetic and minimalist design

Περιγραφή: Οι διάλογοι αλληλεπίδρασης δε πρέπει να περιέχουν πληροφορία που να μην είναι σχετική ή να χρησιμοποιείται σπάνια.

Σχόλια:

• Όλες οι περιοχές στο συγκεκριμένο GUI είναι χρήσιμες και συχνά χρησιμοποιήσιμες από το χρήστη.

9. Help users recognize, diagnose and recover from errors

Περιγραφή: Τα μηνύματα λάθους πρέπει να παρουσιάζονται σε απλή γλώσσα (no codes), να ορίζουν επακριβώς το πρόβλημα και να προτείνουν λύσεις.

Σχόλια:

• Περισσότερα μηνύματα λάθους και διάλογοι επιβεβαίωσης πρέπει να προστεθούν.

10. Help and documentation

Περιγραφή: Αν και είναι καλύτερο ένα σύστημα να μπορεί να χρησιμοποιηθεί χωρίς documentation, ίσως είναι αναγκαίου να παρέχεται ένα είδος υποστήριξης στο χρήστη. Τέτοιου είδους πληροφορία πρέπει να είναι εύκολο να προσεγγιστεί, να είναι συγκεκριμένη στο πρόβλημα του χρήστη, να περιέχει διάφορα βήματα ώστε να είναι πιο εύκολα κατανοητή και να μην είναι πολύ μεγάλη.

Σχόλια:

• On-line βοήθεια πρέπει να παρέχεται στους χρήστες έτσι ώστε οι χρήστες να γνωρίζουν τι τύπου ερωτήσεις επιτρέπονται και πως να διαχειριστούν μια σειρά από ερωτήσεις / απαντήσεις .

ΚΕΦΑΛΑΙΟ 8

ΑΝΑΚΕΦΑΛΑΙΩΣΗ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

8.1 Ανακεφαλαίωση διπλωματικής εργασίας

Σε αυτή την εργασία παρουσιάστηκε η υλοποίηση του OntoNL Framework, μιας γεννήτριας διεπαφών φυσικής γλώσσας, και επίσης ένα σύστημα φυσικής γλώσσας για αλληλεπιδράσεις με βάσεις οπτικοακουστικού υλικού στη περιοχή του ποδοσφαίρου που δημιουργήθηκε βασισμένο στο OntoNL Framework.

Είναι ευρέως γνωστό ότι ένα πρόβλημα με τις διεπαφές φυσικής γλώσσας σε αποθήκες πληροφορίας (information repositories) είναι οι ασάφειες που παρουσιάζονται στα διάφορα αιτήματα (requests) και που μπορούν να οδηγήσουν σε μεγάλους διάλογους αποσαφήνισης.

Η γνώση του πλαισίου στο οποίο μια ασάφεια εμφανίζεται είναι σημαντικό και κρίσιμο για την επίλυση του. Αυτή η παρατήρηση μας οδηγεί στη προσπάθεια εκμετάλλευσης των οντολογιών περιοχής (domain ontologies) που περιγράφουν την περιοχή χρήσης της διεπαφής φυσικής γλώσσας. Η μεθοδολογία που έχουμε υλοποιήσει είναι επαναχρησιμοποιήσιμη , ανεξάρτητη περιοχής, και δουλεύει με μόνη είσοδο την OWL οντολογία σε μια γνωστική περιοχή που μπορεί να χρησιμοποιηθεί ως σχήμα αναφοράς (reference schema) για την κατασκευή μιας αποθήκης γνώσεων (knowledge repository). Η μεθοδολογία της σημασιολογικής αποσαφήνισης εξαρτάται από ένα σημασιολογικό μέτρο συσχέτισης που έχει αναπτυχθεί για τις οντολογίες περιοχών και που έχει ως αποτέλεσμα μια σημασιολογική ταξινόμηση. Αυτή η διαδικασία καταλήγει στην αναπαράσταση της φυσικής γλώσσας για την ανάκτηση πληροφοριών χρησιμοποιώντας μια γλώσσα διατύπωσης ερωτήσεων οντολογίας, τη SPARQL. Οι ερωτήσεις SPARQL ταξινομούνται βασισμένοι στη σημασιολογική αξία του μέτρου συσχέτισης που χρησιμοποιείται επίσης για την αυτόματη κατασκευή ερωτήσεων.

Αυτή η μεθοδολογία είναι ενσωματωμένη στο **OntoNL Framework**, ένα γενικευμένο αρχιτεκτονικό πλαίσιο για τη κατασκευή και χρησιμοποίηση των

διεπαφών φυσικής γλώσσας για τις αλληλεπιδράσεις χρηστών με τις αποθήκες γνώσεων (knowledge repositories). Η εφαρμογή του πλαισίου του OntoNL εξετάζει μία σημασιολογική βάση πολυμέσων με το ψηφιακό οπτικοακουστικό περιεχόμενο των γεγονότων ποδοσφαίρου γενικά, που έχουν αναπτυχθεί και καταδειχθεί στο δεύτερο και τρίτο Annual Review of the DELOS II EU Network of Excellence (IST 507618) (http://www.delos.info/).

Το **OntoNL Framework** υλοποιεί μία πλατφόρμα λογισμικού που αυτοματοποιεί σε ένα μεγάλο βαθμό τη κατασκευή των διεπαφών φυσικής γλώσσας για τις βάσεις γνώσεων. Για να επιτευχθεί η δυνατότητα εφαρμογής και επαναχρησιμοποίησης του πλαισίου OntoNL σε πολλές διαφορετικές εφαρμογές και περιοχές, το λογισμικό είναι ανεξάρτητο από τις οντολογίες περιοχών (domain ontologies).

Τα τμήματα λογισμικού του **OntoNL Framework** εξετάζουν ομοιόμορφα μια σειρά προβλημάτων στην ανάλυση της πρότασης τα οποία απαιτούν χωριστούς μηχανισμούς. Μια ενιαία αρχιτεκτονική χειρίζεται τη συντακτική και σημασιολογική ανάλυση, αλλά και τις ασάφειες. Παράλληλα, το πλαίσιο έχει κατασκευαστεί με τέτοιο τρόπο έτσι ώστε να αποφεύγονται οι εξαρτήσεις με τις βάσεις πληροφοριών (information repositories) με αποτέλεσμα να είναι επαναχρησιμοποιήσιμο σε διαφορετικές εφαρμογές με διαφορετική σημασιολογία περιοχών (domain semantics).

Αυτή η έρευνα που διεξήχθη σε αυτήν τη διατριβή υποστηρίχθηκε από το DELOS II, Network of Excellence on Digital Libraries NoE-G038-507618 / IST-507618

8.2 Μελλοντικές επεκτάσεις

Όσον αφορά το γραφικό περιβάλλον εργασίας του OntoNL και λαμβάνοντας υπόψιν τα διάφορα σχόλια που προέκυψαν από την αξιολόγηση του συστήματος θα μπορούσαν να προταθούν ως μελλοντική επέκταση τα εξής:

- Παροχή συστήματος βοήθειας (help system). Παρόλο που το γραφικό περιβάλλον εργασίας του OntoNL είναι ιδιαίτερα φιλικό προς το χρήστη, έτσι ώστε ο χρήστης να είναι σε θέση να προσαρμόζεται γρήγορα και εύκολα στο περιβάλλον εργασίας του, είναι επιθυμητή η ύπαρξη ενός συστήματος βοήθειας το οποίο θα παρέχει πληροφορίες για το εργαλείο και τις λειτουργίες του με παραδείγματα και οδηγούς για την εκτέλεση των λειτουργιών

- Πολύ χρήσιμο και ενδιαφέρον θα ήταν επίσης να γίνει σχεδιασμός και υλοποίηση ενός component για αυτόματη μετατροπή της Sparql στις πιο χρησιμοποιήσιμες – διάσημες query languages όπως η SQL και η XQuery.
- Επίσης, η υλοποίηση του GUI σε handheld devices θα μπορούσε να αποτελέσει μια μελλοντική επέκταση.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. A.Karanastasi 2007. OntoNL: An Ontology- Based Natural Language Interface Generator for Knowledge Repositories, Master Thesis, Technical University of Crete, Department of Electronic and Computer Engineering
2. Karanastasi, S. Christodoulakis: *"Semantic Processing of Natural Language Queries in the OntoNL Framework"*, in the Proceedings of the IEEE International Conference on Semantic Computing (IEEE ICSC), 17-19 September 2007, Irvine, CA
3. A. Karanastasi, S. Christodoulakis: *"Ontology-Driven Semantic Ranking for Natural Language Disambiguation in the OntoNL Framework"*, in the Proceedings of the 4th European Semantic Web Conference (ESWC), 3-7 June 2007, Innsbruck, Austria
4. A. Karanastasi, S. Christodoulakis: *"The OntoNL Semantic Relatedness Measure for OWL Ontologies"*, in the Proceedings of the Second IEEE International Conference on Digital Information Management (IEEE ICDIM), 28-31 October 2007, Lyon, France
5. A. Karanastasi, A. Zotos, S. Christodoulakis: *"The OntoNL Framework for Natural Language Interface Generation and a Domain-Specific Application"*, in the Proceedings of the DELOS Conference on Digital Libraries 2007, 13-14 February 2007, Tirrenia, Pisa, Italy
6. A. Karanastasi, A. Zotos, S. Christodoulakis: *" User Interactions with Multimedia Repositories using Natural Language Interfaces - OntoNL: an Architectural Framework and its Implementation"*, in Journal of Digital Information Management - JDIM, Volume 4, Issue 4, December 2006
7. A. Karanastasi, A. Zotos, S. Christodoulakis: *" User Interactions with Multimedia Repositories using Natural Language Interfaces: an Architectural Framework and its Implementation"*, in Proceedings of the Fourth Special Workshop on Multimedia Semantics - WMS06, June 19-21, 2006
8. S. Christodoulakis, A. Karanastasi, J. Koehler, K. Biatov, T. Catarci, S. Kimani: *" Natural Language and Speech Interfaces to Knowledge Repositories"*,

Poster on the 9th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2005), September 2005, Vienna, Austria

9. A. Karanastasi, S. Christodoulakis: *"OntoNL: An Ontology-based Natural Language Interface Generator for Multimedia Repositories"*, In Proceedings of the AVIVDiLib Seventh International Workshop of the EU Network of Excellence DELOS on AUDIO-VISUAL CONTENT AND INFORMATION VISUALIZATION IN DIGITAL LIBRARIES (AVIVDiLib'05), May 2005
10. TSINARAKI, C., CHRISTODOULAKIS, S. 2006. A Multimedia User Preference Model that supports Semantics and its application to MPEG 7/21, In Proceedings of MMM 2006, Beijing, China, 4-6 January 2006
11. Vargas-Vera, M., Motta, E., *AQUA – Ontology-based Question Answering System.* Third International Mexican Conference on Artificial Intelligence (MICAI-2004), Lecture Notes in Computer Science 2972 Springer Verlag, 2004
12. ISO/IEC JTC1/SC29/WG11. N8219 – MPEG-7 Query Format Requirements version 1.1, Klagenfurt, Austria, July 2006.
13. Chrisa Tsinaraki, Stavros Christodoulakis: "*An MPEG-7 Query Language and a User Preference Model that allow Semantic Retrieval and Filtering of Multimedia Content*", ACM-Springer Multimedia Systems Journal, in Special Issue on Semantic Multimedia Adaptation and Personalization, 2007
14. Polydoros P., Tsinaraki C., Christodoulakis S.: *"GraphOnto: OWL-Based Ontology Management and Multimedia Annotation in the DS-MIRF Framework",* In the proceedings of the Workshop on Multimedia Semantics 2006 (WMS 2006), pp. 14-24, 19-21 June 2006, Chania, Crete
15. MPEG-7 Overview, version 8, July 2002 http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm [Accessed August 22, 2003]
16. Java™ 2 Platform – *Standard Edition, v. 1.4.1 API Specification,* http://java.sun.com/j2se/1.4.1/docs/api/

17. Jurafsky, D., Martin, J.H. (2000). *Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* New Jersey: Prentice Hall

18. OWL Web Ontology Language Overview, W3C Recommendation 10 February 2004, Deborah L. McGuinness (Knowledge Systems Laboratory, Stanford University), Frank van Harmelen (Vrije Universiteit, Amsterdam)

19. RDF: Resource Description Framework (RDF): Concepts and Abstract Syntax, Recommendation, World Wide Web Consortium, 2004-02-10; http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

20. THE PENN TREEBANK PROJECT, www.cis.upenn.edu/~treebank/

21. SPARQL Query Language for RDF : http://www.w3.org/TR/2007/CR-rdf-sparql-query-20070614/

22. Extensible Markup Language (XML) 1.0 (Third Edition), 2004, http://www.w3.org/XML

23. Wordnet: http://wordnet.princeton.edu/

24. JWordnet: http://jwn.sourceforge.net/

25. XML Beans: http://xmlbeans.apache.org/

26. Graph pattern: http://www.w3.org/TR/rdf-sparql-query/#GraphPattern

27. Jena ARQ API: http://jena.sourceforge.net/ARQ/app_api.html

28. :http://www.ibm.com/developerworks/library/w-ovr/

29. WOODS, W. A., KAPLAN, R. M., WEBBER, B. N. 1972 The Lunar Sciences Natural Language Information System: Final Report. BBN Report 2378, Bolt Beranek and Newman Inc., Cambridge, Massachusetts

30. WOODS, W.A. 1968 Procedural Semantics for a Question-Answering Machine. In Proceedings of the Fall Joint Computer Conference, pages 457–471, New York, NY, 1968. AFIPS

31. CODD, E.F. 1974. Seven Steps to RENDEZVOUS with the Casual User. In J. Kimbie and K. Koffeman, editors, Data Base Management. North-Holland Publishers, 1974

32. HENDRIX, G., SACERDOTI, E., SAGALOWICZ, D., SLOCUM, J. 1978 Developing a Natural Language Interface to Complex Data. ACM Transactions on Database Systems, 3(2):105–147, 1978

33. WALTZ, D.L. 1978 An English Language Question Answering System for a Large Relational Database. Communications of the ACM, 21(7):526–539, July 1978

34. SCHA, R.J.H. 1977 Philips Question Answering System PHILIQA1. In SIGART Newsletter, no.61. ACM, New York, February 1977

35. WARREN, D, PEREIRA, F. 1982 An Efficient Easily Adaptable System for Interpreting Natural Language Queries. Computational Linguistics, 8(3-4):110–122, July-December 1982

36. GROSZ. B. J. 1983 TEAM: A Transportable Natural-Language Interface System. In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California, pages 39–45

37. THOMPSON, B. H., THOMPSON, F. B. 1983 Introducing ASK, A Simple Knowledgeable System. In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California, pages 17–24, 1983

38. RESNIK, P. 1989. Access to Multiple Underlying Systems in JANUS. BBN report 7142, Bolt Beranek and Newman Inc., Cambridge, Massachusetts, September 1989

39. JOHNSON, T.1985. Natural Language Computing: The Commercial Applications. Ovum Ltd., London

40. ALSHAWI,, H., CARTER, D., CROUCH, R., PULMAN, S., RAYNER, M., SMITH, A. CLARE 1992 – A Contextual Reasoning and Cooperative Response Framework for the Core Language Engine. Final report, SRI International, December 1992

41. CODD, E.F. 1970 A Relational Model for Large Shared Data Banks. Communications of the ACM, 13(6):377–387, 1970

42. http://www.w3schools.com/sql/sql_intro.asp

43. MELTON, J. 2006 SQL, XQuery, and SPARQL: What's Wrong With This Picture? XTech 2006: "Building Web 2.0" — 16-19 May 2006, Amsterdam, The Netherlands

44. VARGAS-VERA, M., MOTTA, E. 2004. AQUA – Ontology-based Question Answering System. Third International Mexican Conference on Artificial Intelligence (MICAI-2004), Lecture Notes in Computer Science 2972 Springer Verlag, 2004

45. CIMIANO, P., HAASE, P., SURE, Y., VÖLKER, J., AND WANG, Y. 2006. Question answering on top of the BT digital library. In Proceedings of the 15th international Conference on World Wide Web (Edinburgh, Scotland, May 23 - 26, 2006). WWW '06. ACM Press, New York, NY, 861-862

46. http://www.w3.org/TR/xquery/

47. http://www.w3.org/TR/xpath

48. XDM: XQuery 1.0 and XPath 2.0 Data Model (XDM), Candidate Recommendation, World Wide Web Consortium, 2005-11-03; http://www.w3.org/TR/2005/CR-xpath-datamodel-20051103

49. http://www.uml.org/

50. NIELSEN, J. 1994. Heuristic Evaluation, In Nielsen, J. and Mack, R. L. (Eds.), Usability Inspection Methods. John Wiley and Sons, New York, pp. 25-62

51. http://xmlbeans.apache.org/

52. M. Kay (ed.). XSL Transformations (XSLT) Version 2.0. W3C Candidate Recommenda-tion, 3 Nov. 2005. (http://www.w3.org/TR/xslt20/)

53. ISO/IEC JTC1/SC29/WG11. N8219 – MPEG-7 Query Format Requirements version 1.1, Klagenfurt, Austria, July 2006

54. ISO/IEC: 15938-3:2001: Information Technology – Multimedia content description interface – Part 3 Visual (2001) Version 1

55. ISO/IEC: 15938-4:2001: Information Technology – Multimedia content description interface – Part 4 Audio (2001) Version 1

56. Frederick Jelinek, John D. Lafferty, and Robert L. Mercer. 1992. Basic methods of probabilistic context free grammars. In Laface and De Mori [Laface and De Mori1992], pages 345-360. Proceedings of the NATO Advanced Study Institute, Cetraro, Italy, July 1990.