Ανοικτο Πανεπιστημιο Κυπρου

Μεταπτυχιακό Πρόγραμμα Πληροφοριακών Συστημάτων
Φοιτητής: Νίκος Ισαάκ
Επιβλέπων Καθηγητής: Δρ. Λοΐζος Μιχαήλ

Διπλωματική εργασία με θέμα:

«Μια πρώτη προσπάθεια δημιουργίας διαδικτυακής μηχανής συμπερασμάτων κοινής λογικής»

Ιούλιος 2011

Περίληψη

Οι άνθρωποι έχουν την ικανότητα εξαγωγής συμπερασμάτων κοινής λογικής, τα οποία έχουν σχέση με το κρυμμένο νόημα προτάσεων ή και φράσεων που διαβάζουν την κάθε φορά, όπως π.χ. στα διάφορα έντυπα και ηλεκτρονικά μέσα.

Σε αυτή την έρευνα παρουσιάζουμε τον τρόπο δημιουργίας μιας νέας διαδικτυακής μηχανής που έχει την ικανότητα έξυπνης αλληλεπίδρασης με το χρήστη, την οποία ονομάζουμε Έξυπνη Μηχανή. Ο χρήστης κάνει την επερώτηση και η Έξυπνη Μηχανή ανταποκρίνεται εμφανίζοντας συμπεράσματα κοινής λογικής που έπονται της επερώτησης του χρήστη, σύμφωνα με την ανθρώπινη γνώση.

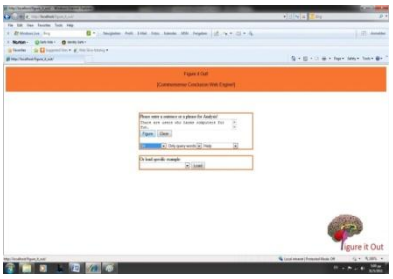
Ως πηγή μόρφωσης για την Έξυπνη Μηχανή χρησιμοποιούμε το διαδίκτυο, όπου υπάρχουν πολυάριθμες ιστοσελίδες που κωδικοποιούν εμμέσως την ανθρώπινη γνώση. Κυρίως μέσω μελέτης τεχνικών NLP (Natural Language Processing), μέσω μελέτης του τρόπου λειτουργίας υπαρχουσών μηχανών αναζήτησης και μέσω ανάπτυξης λογισμικών προχωρούμε στην αυτοματοποιημένη λήψη πληροφοριών από ιστοσελίδες του διαδικτύου. Ακολούθως, επεξεργαζόμαστε τις πληροφορίες για να τις φέρουμε σε κατάλληλη μορφή, μέσω της οποίας, σε συνδυασμό με συγκεκριμένα λογισμικά, μπορεί να λειτουργεί η Έξυπνη Μηχανή και να εξάγει συμπεράσματα κοινής λογικής σε επερωτήσεις χρηστών.

Περιεχόμενα

1 Εισαγωγή

Πώς ξεκινήσαμε και πού θέλουμε να καταλήξουμε;

Σε αυτό το κεφάλαιο παρουσιάζουμε το σκεπτικό, το οποίο μας οδήγησε στην ανάπτυξη της διαδικτυακής μηχανής συμπερασμάτων κοινής λογικής (Έξυπνης Μηχανής).



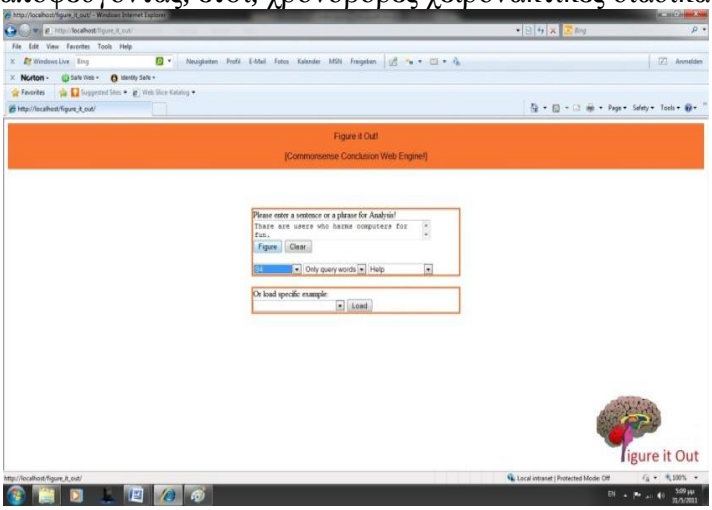
Ο κάθε άνθρωπος έχει την ικανότητα εξαγωγής συμπερασμάτων κοινής λογικής, μέσω συλλογισμών που κάνει την κάθε φορά και που έχουν σχέση με το κρυμμένο νόημα φράσεων ή προτάσεων που διαβάζει. Για παράδειγμα, αν έχουμε την πρόταση «Ο Γιάννης πήγε για πικνίκ με την οικογένειά του», μπορούμε να εξαγάγουμε συμπεράσματα κοινής λογικής όπως:

- *Ο καιρός ήταν ηλιόλουστος.*
- *Ο καιρός δεν ήταν βροχερός.*
- *Ο Γιάννης πέρασε ωραία με την οικογένειά του.*

Σχήμα 1.α | *Παράδειγμα μέσα από το οποίο εξάγουμε συμπεράσματα κοινής λογικής.*

Στόχος αυτής της ερευνητικής δουλειάς ήταν η δημιουργία μιας διαδικτυακής μηχανής, η οποία θα είχε την ικανότητα έξυπνης αλληλεπίδρασης με τους χρήστες. Οι χρήστες θα έκαναν επερωτήσεις και η Έξυπνη Μηχανή θα ανταποκρινόταν επιστρέφοντας συμπεράσματα κοινής λογικής που θα έπονταν αυτών των επερωτήσεων.

Για να μπορεί να επιστρέφει η Έξυπνη Μηχανή τα συμπεράσματα κοινής λογικής, θα έπρεπε να μορφωθεί με κάποιο τρόπο. Αποφασίσαμε η μόρφωση να γίνεται μέσω αυτοματοποιημένης διαδικασίας λήψης πληροφοριών από τις πολυάριθμες ιστοσελίδες του διαδικτύου που κωδικοποιούν εμμέσως την ανθρώπινη γνώση, αποφεύγοντας, έτσι, χρονοβόρες χειρονακτικές διαδικασίες μόρφωσης.



Σχήμα 1.β | *Περιβάλλον αλληλεπίδρασης Έξυπνης Μηχανής.*

Διεξαγάγαμε και έρευνα υπό μορφή ερωτηματολογίου (βλέπε Παράρτημα, Ερωτηματολόγιο, σελίδα 82) που τέθηκε σε μαθητές και καθηγητές σχολείων Μέσης Εκπαίδευσης, όπου προσπαθήσαμε να εντοπίσουμε τις εμπειρίες των χρηστών σε σχέση με τη χρήση υπαρχουσών μηχανών αναζήτησης, αλλά και τις απόψεις τους σε σχέση με τη νέα μηχανή που θέλαμε να δημιουργήσουμε.

Στο ερωτηματολόγιο τέθηκαν ερωτήσεις όπως:

- *Πόσο συχνά χρησιμοποιείτε τις μηχανές αναζήτησης;*
- *Για ποιο σκοπό χρησιμοποιείτε τις μηχανές αναζήτησης;*
- *Πόσο ευχαριστημένοι είστε από τις απαντήσεις που παίρνετε;*
- *Σε ποιο ποσοστό της συνολικής χρήσης μηχανών αναζήτησης που κάνετε, πιστεύετε ότι θα χρησιμοποιούσατε μια Έξυπνη Μηχανή εξαγωγής συμπερασμάτων κοινής λογικής;*
- *Τι είδους αλληλεπίδραση (ερώτησης και απάντησης) θα θέλατε να έχετε με μια Έξυπνη Μηχανή εξαγωγής συμπερασμάτων κοινής λογικής;*

Κατά την αξιολόγηση των αποτελεσμάτων από τους ερωτηθέντες καταλήξαμε σε συμπεράσματα όπως:

Στις υπάρχουσες μηχανές αναζήτησης:

- Δεν εξάγονται συμπεράσματα κοινής λογικής.
- Είναι ευχαριστημένοι οι χρήστες από τις απαντήσεις που παίρνουν, αλλά είναι χρονοβόρα η διαδικασία εντοπισμού πληροφοριών.
- Οι χρήστες τις χρησιμοποιούν κυρίως για την αναζήτηση γενικότερων πληροφοριών, αλλά και διευθύνσεων ιστοσελίδων.

Σε σχέση με την Έξυπνη Μηχανή:

- Οι χρήστες των υπαρχουσών μηχανών αναζήτησης θα ήθελαν τη δημιουργία μιας Έξυπνης Μηχανής, η οποία θα επέστρεφε συμπεράσματα κοινής λογικής σε επερωτήσεις.
- 75% δήλωσαν ότι θα χρησιμοποιούσαν την Έξυπνη Μηχανή σε ποσοστό 81%-100% του συνολικού τους χρόνου που αφιέρωναν στις υπάρχουσες μηχανές αναζήτησης.
- Ως μορφή απάντησης που θα ήθελαν να παίρνουν προτιμούσαν αυτήν της πρότασης-φράσης.
- Θα προτιμούσαν να έπαιρναν την απάντηση μέσα στο χρονικό διάστημα των δέκα δευτερολέπτων.
- Θα τη χρησιμοποιούσαν για ψάξιμο συγκεκριμένων πληροφοριών.

Ακολούθως, προχωρήσαμε σε ανασκόπηση της σχετικής βιβλιογραφίας για εντοπισμό παρόμοιων ή σχετικών συστημάτων που αναπτύχθηκαν στο παρελθόν. Στόχος ήταν η εύρεση χρήσιμων πληροφοριών, αλλά και λογισμικών που θα μας βοηθούσαν στη σχεδίαση και ανάπτυξη της Έξυπνης Μηχανής.

2

Ανασκόπηση Σχετικής Βιβλιογραφίας

Ποια συστήματα που αναπτύχθηκαν και γενικά, ποιες προσεγγίσεις που ακολουθήθηκαν μπορούν να μας βοηθήσουν στη σχεδίαση και ανάπτυξη της Έξυπνης Μηχανής;

Σε αυτό το κεφάλαιο παρουσιάζονται πληροφορίες για έξυπνα συστήματα που αναπτύχθηκαν στο παρελθόν (Cyc, OpenMind, Wolfram Alpha), αλλά και προσεγγίσεις που ακολουθήθηκαν και ήταν σε θέση να εγγυηθούν τον τρόπο δημιουργίας έξυπνων συστημάτων εξαγωγής συμπερασμάτων κοινής λογικής (Knowledge Infusion). Επίσης, μέσω ανάλυσης του τρόπου λειτουργίας υπαρχουσών μηχανών αναζήτησης, προσπαθούμε να εντοπίσουμε τον τρόπο ανάκτησης ιστοσελίδων από την κάθε μηχανή.

«Ονομάζουμε το είδος μας homo sapiens — άνθρωπος ο σοφός — επειδή οι νοητικές μας ικανότητες είναι πολύ σημαντικές για μας και επειδή, για χιλιάδες χρόνια, προσπαθούμε να κατανοήσουμε το πώς μια χούφτα ύλης μπορεί να αντιλαμβάνεται, να κατανοεί, να προβλέπει και να χειρίζεται έναν κόσμο πολύ μεγαλύτερο και πολύ πιο περίπλοκο από τον εαυτό της» [15].

Μία από τις πιο σημαντικές προκλήσεις της επιστήμης Η.Υ. και ειδικότερα της Τεχνητής Νοημοσύνης είναι να καταλάβει τον τρόπο με τον οποίο θα μπορέσει να δημιουργήσει συστήματα κοινής λογικής, τα οποία θα μπορούν να συλλογίζονται όπως ο κάθε άνθρωπος [21]. Συστήματα τα οποία θα μπορούσαν κάποτε στο μέλλον να αντικαταστήσουν ή και να υποβοηθήσουν βασικές ανθρώπινες λειτουργίες. Ως σύστημα κοινής λογικής ονομάζουμε το σύστημα, το οποίο αυτόματα παράγει ένα σύνολο από ενέργειες, την κάθε στιγμή αναλόγως με το τι θα του πουν και αναλόγως με το τι ξέρει [11].

Ο μηχανικός συλλογισμός ήταν ένας στόχος που τέθηκε από τα μέσα της δεκαετίας του ’50 για δημιουργία συστημάτων, τα οποία θα συλλογίζονταν σε συγκεκριμένους τομείς [11]. Τελική επιθυμία ήταν η δημιουργία συστημάτων, τα οποία θα δέχονταν μεγάλο αριθμό πληροφοριών για μάθηση, πάνω στα οποία θα μπορούσαν πλέον να γίνονται συλλογισμοί για την εξαγωγή χρήσιμων συμπερασμάτων [12]. Οι περισσότερες προσπάθειες που έγιναν είχαν ως κύριο στόχο όχι μόνο τη λύση των προβλημάτων, αλλά και τη σύγκριση της ακολουθίας συλλογιστικών βημάτων με την αντίστοιχη ακολουθία συλλογιστικών βημάτων των ανθρώπων που έλυναν το ίδιο πρόβλημα [15]. Η ανάκτηση όμως και η χρήση των πληροφοριών είναι σε μορφή γεγονότων με την εξαγωγή κάποιων απλών γενικεύσεων, χωρίς να εξάγονται συμπεράσματα κοινής λογικής [2]. Αρκετά συστήματα τα οποία δημιουργήθηκαν, προϋποθέτουν πολλή χειρονακτική προσπάθεια στον τομέα της μόρφωσης και αυτή δεν είναι σε κατάλληλη μορφή για συλλογισμό [18] ή, αν είναι σε μια κοντινή μηχανική μορφή [7], τα πάντα θεωρούνται αληθή [2].

2.1 Παραδείγματα έξυπνων συστημάτων

2.1.1 Cyc[1]

Το Cyc [7] ασχολείται με την αναπαράσταση και με τα συμπεράσματα σε σχέση με τη γνώση. Ασχολείται με μια μορφή προτάσεων κατηγορηματικού λογισμού και οι κανόνες καταχωρούνται με το χέρι (θεωρείται πως ό,τι καταχωρείται είναι αληθές). Το όλο έργο ξεκίνησε το 1994 και κατά καιρούς κάποιοι το θεωρούσαν ως ένα από τα μοναδικά συστήματα που εξήγαγε γεγονότα που περιείχαν λογική.

> «People have silly reasons why computers don't really think. The answer is we haven't programmed them right; they just don't have much common sense. There's been only one large project to do something about that, that's the famous Cyc project».
> ~ Marvin Minsky, MIT, May 2001

Το Cyc έχει διάφορες μορφές αλληλεπίδρασης με το χρήστη, αλλά η πιο χαρακτηριστική είναι η εμφάνιση μιας πυραμίδας, μέσω της οποίας μπορούμε να εντοπίζουμε γεγονότα για διάφορα θέματα μέσω του ποντικιού του Η.Υ.

[1] http://www.cyc.com [Μάιος 2011].

Σχήμα 2.1.1 | *Πυραμίδα αλληλεπίδρασης Cyc.*

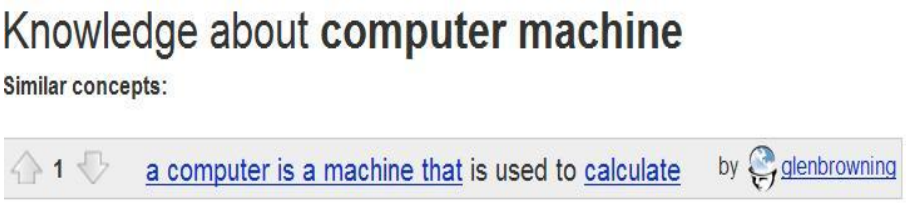
Το σύστημα κωδικοποιεί πληροφορία, όπως για παράδειγμα, το να μοιράζεις μια κιμωλία στη μέση παράγει δύο κιμωλίες.

2.1.2 OpenMind[2]

Το OpenMind [18] χρησιμοποιεί φυσική γλώσσα για αναπαράσταση της γνώσης και αυτό βοηθάει τους χρήστες στην καταχώρηση πληροφοριών μέσω του διαδικτύου για εμπλουτισμό της γνώσης του.

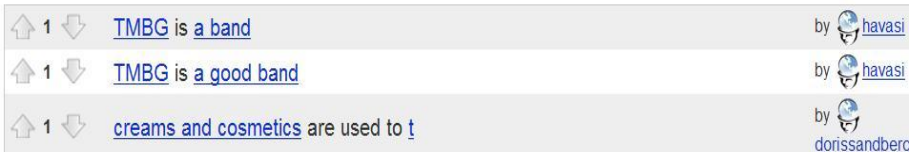
Κατά τη διεξαγωγή πειραμάτων με το OpenMind, εντοπίσαμε τα εξής:

- Αφού εισάγουμε τη λέξη ή φράση που θέλουμε, η μηχανή επιστρέφει μόνο προτάσεις που περιέχουν αυτή τη λέξη ή φράση ή και συνδυασμό των λέξεων. Π.χ. Αν εισάγουμε τη φράση *computer machine*, μας επιστρέφονται τα εξής:

  Σχήμα 2.1.2.α | *Παράδειγμα εξόδου της μηχανής OpenMind.*

  Δηλαδή, αν στη βάση γνώσης δεν υπάρχουν προτάσεις που περιέχουν τη λέξη ή φράση κλειδί που εισάγουμε, δεν επιστρέφεται τίποτα.
- Αν εισάγουμε ένα γράμμα όπως *t* ή κάτι άλλο, η μηχανή ανταποκρίνεται θετικά, επιστρέφοντάς μας και προτάσεις με λέξεις που περιέχουν ή ξεκινούν από αυτό το γράμμα.

  Σχήμα 2.1.2.β | *Παράδειγμα εξόδου της μηχανής OpenMind.*

  Γενικά, μπορούμε να πούμε πως η συγκεκριμένη μηχανή επιστρέφει μόνο προτάσεις που περιέχουν λέξεις της επερώτησής μας.

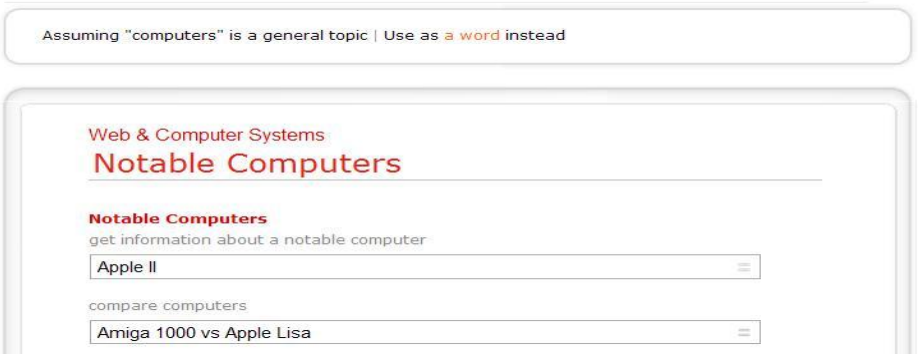
[2] http://openmind.media.mit.edu/en/concept/ [Μάιος 2011].

2.1.3 Wolfram Alpha[3]

Το Wolfram Alpha είναι ένα σύστημα, το οποίο προσεγγίζει τη γνώση από υπολογιστικής πλευράς και απαντάει ερωτήματα, αντλώντας πληροφορία από πολλά *domains*. Όπως και στο Cyc, οι πληροφορίες μαζεύονται και οργανώνονται δια χειρός, και οι διαδικασίες αναπτύσσονται για να δουλεύουν με τα υπάρχοντα δεδομένα.

Μέσα από πειράματά μας με το Wolfram Alpha εντοπίσαμε τα εξής:

- Όταν εισάγονται λέξεις που αναφέρονται σε πολύ γνωστές έννοιες, τα αποτελέσματα είναι πολύ καλά. Κυρίως όμως, η μηχανή ανταποκρίνεται εμφανίζοντας πληροφορίες για τα συνώνυμα, παρουσιάζοντας, συνάμα, και προτάσεις που έχουν σχέση με τη λέξη-φράση κλειδί.
- Ακόμη, μέσα σε κάθε φράση που εισάγει ο χρήστης, η μηχανή εντοπίζει τη λέξη που θεωρεί πως είναι η πιο σημαντική και προσπαθεί να εμφανίσει τα αποτελέσματα για αυτή τη λέξη.
  Για παράδειγμα, κατά την εισαγωγή της φράσης *spyware infects computers*, παίρνουμε τα εξής αποτελέσματα:

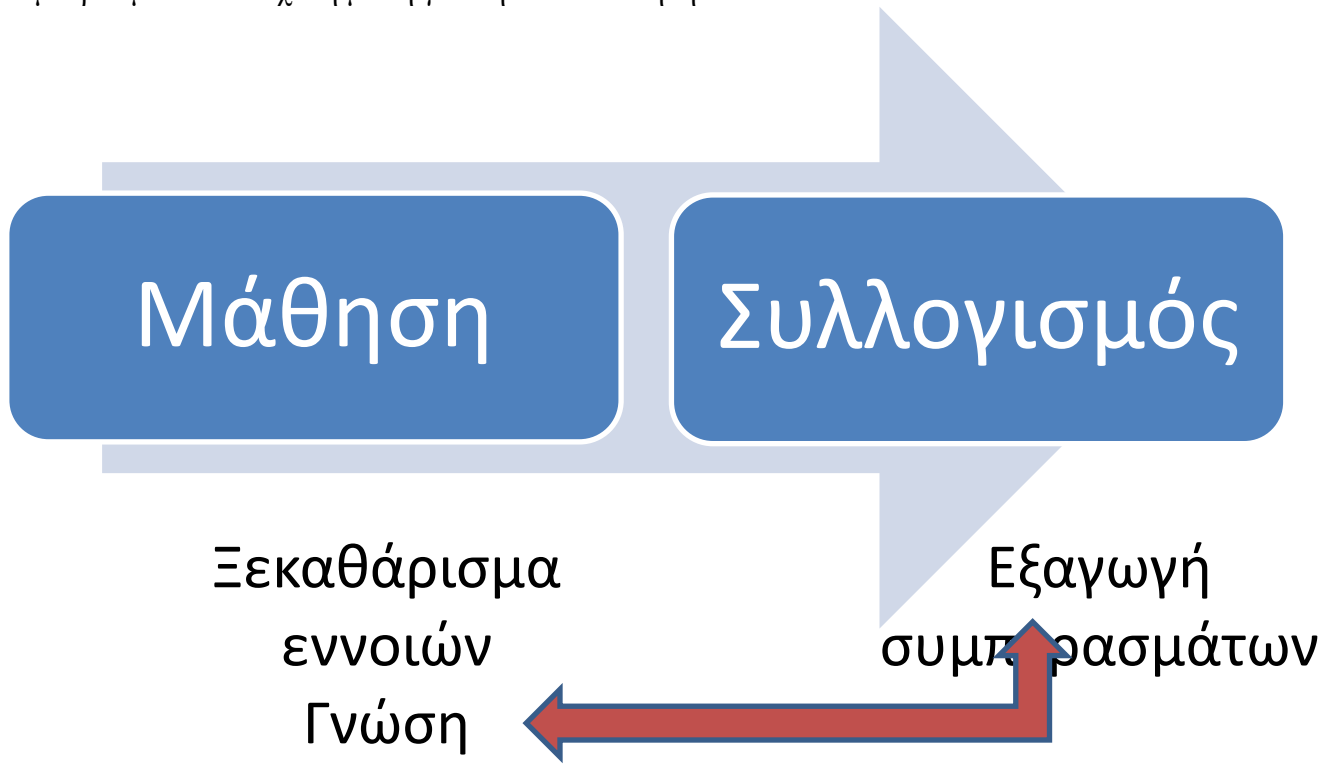


Σχήμα 2.1.3.α | Παράδειγμα εξόδου της μηχανής Wolfram Alpha.

[3] http://www.wolframalpha.com [Μάιος 2011].

2.2 Knowledge Infusion

Υπήρξαν συγκεκριμένες θεωρητικές προσεγγίσεις και μελέτες, οι οποίες ήταν σε θέση να εγγυηθούν τη δημιουργία έξυπνων συστημάτων [21]. Πρόκειται για συστήματα, τα οποία στηρίζονται σε δύο βασικούς άξονες, αυτόν της μάθησης και αυτόν του συλλογισμού. Η μάθηση γίνεται για το ξεκαθάρισμα εννοιών και γεγονότων (γνώση), ενώ η διαδικασία του συλλογισμού χρειάζεται για εξαγωγή συμπερασμάτων σε σχέση με τη γνώση που αποκτήθηκε.

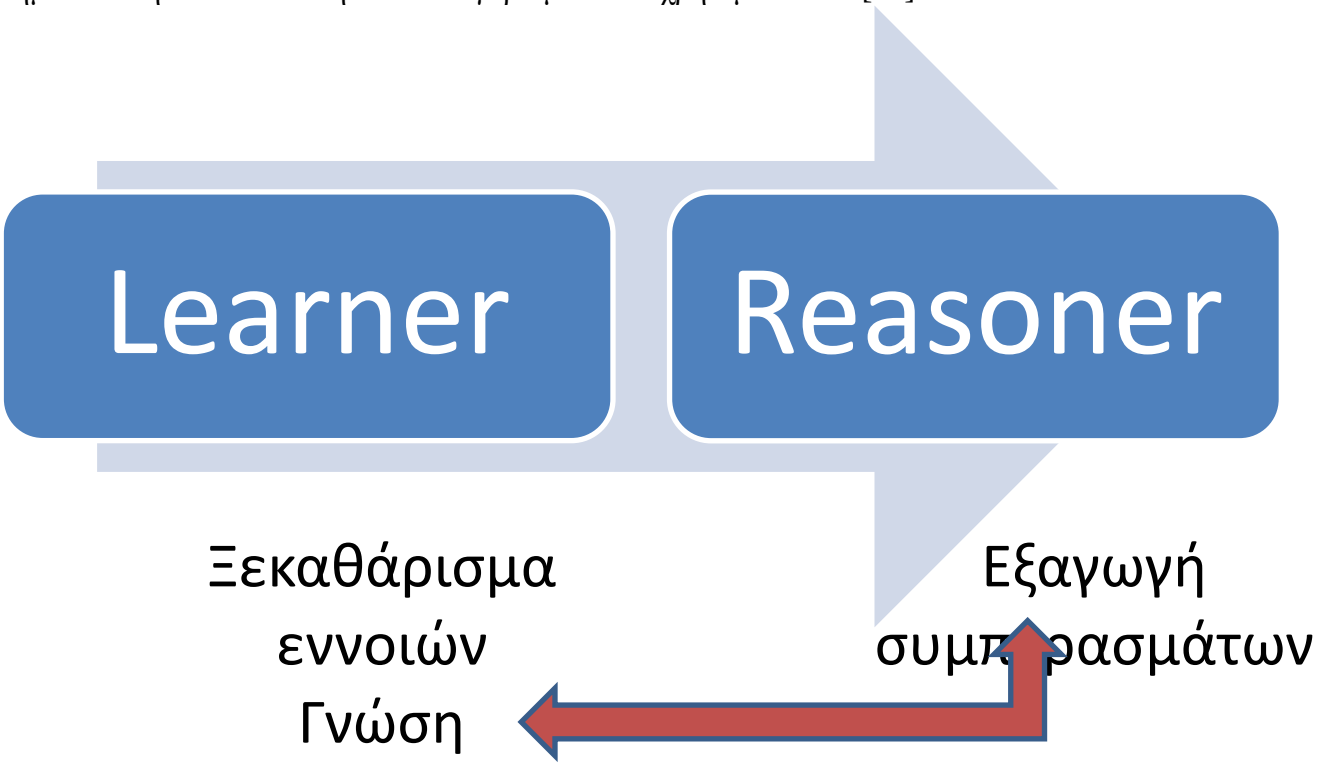


Σχήμα 2.2.α | *Συστήματα που στηρίζονται στη μάθηση και το συλλογισμό.*

Τα πιο πάνω έχουν άμεση σχέση με το robust logic [20], το οποίο προσφέρει τον τρόπο δημιουργίας τέτοιων συστημάτων, τα οποία μπορούν να μαθαίνουν και να συλλογίζονται με γρήγορους ρυθμούς.
Αφορούν τον τρόπο υλοποίησης ενός αλγόριθμου συμπερασμάτων κοινής λογικής, ο οποίος, με τη σειρά του, στηρίζεται στον αλγόριθμο Winnow [8] που σχεδιάστηκε για να χωρίζει αποτελεσματικά σχετικά από άσχετα χαρακτηριστικά, σε ένα περιβάλλον μάθησης. Ο Winnow είναι αλγόριθμος απλός και βαρύς (simple and massive) αλλά και συνάμα πολύ γρήγορος [8]. Απλός, γιατί δέχεται ένα σύνολο από κόμβους εισαγωγής και τους τοποθετεί σε δύο ξεχωριστά μέρη, ζυγίζοντας και ελέγχοντας κάποια κριτήρια. Βαρύς, γιατί σχεδιάστηκε για να δουλεύει με πάρα πολλά δεδομένα, όπου οι κόμβοι εισαγωγής μπορεί να είναι από μερικούς μέχρι μερικές δεκάδες χιλιάδες. «Γρήγορος, γιατί μπορεί να εξάγει συλλογισμούς μέσα σε πολύ σύντομα χρονικά διαστήματα μερικών δευτερολέπτων, σε σχέση με άλλες τεχνικές, οι οποίες ήταν πάρα πολύ αργές» [21].

Αργότερα, μέσω ανάπτυξης συγκεκριμένων λογισμικών, υλοποιήθηκαν και επιβεβαιώθηκαν τα πιο πάνω [12]. Θα μπορούσαμε να πούμε μεταφορικά, πως έχει υλοποιηθεί ένα σύστημα, το οποίο μοιάζει με παιδί που διαθέτει όλα τα απαραίτητα όργανα για μάθηση και συλλογισμό και το οποίο πρέπει να μορφωθεί με κάποιον τρόπο. Η μάθηση υλοποιείται με το λογισμικό Learner και η διαδικασία των

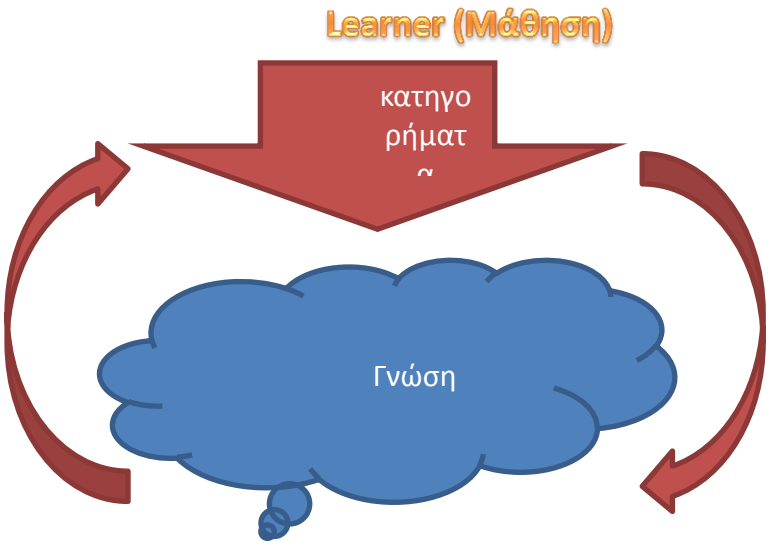
συλλογισμών με το λογισμικό Reasoner σε πολύ γρήγορους ρυθμούς, αντιθέτως με άλλα συστήματα που αναπτύχθηκαν στο παρελθόν, τα οποία δε δίνουν μεγάλη σημασία στην αποδοτικότητα των αλγορίθμων που χρησιμοποιούν [14].



Σχήμα 2.2.β | *Μάθηση με το λογισμικό Learner και συλλογισμοί με το λογισμικό Reasoner.*

2.2.1 Μάθηση

Η διαδικασία της μάθησης ξεκινά με την καταχώρηση κατηγορημάτων στο λογισμικό Learner που αντιστοιχούν σε φυσικές προτάσεις και ολοκληρώνεται με τη δημιουργία της γνώσης.



Σχήμα 2.2.1 | *Διαδικασία μάθησης με το Learner.*

Για παράδειγμα, η φυσική πρόταση «Honest John is playing football», μπορεί να αντιστοιχεί στα εξής κατηγορήματα:

o___OP_WRD_OP__John_o([token:1]) is true.
o___OP_WRD_OP__Honest_o([token:1]) is true.
o___OP_WRD_OP__football_o([token:2]) is true.
o___OP_REL_OP__playing_o([token:1, token:2]) is true.

Όπου ισχύουν τα εξής:

Τα μοναδιαία κατηγορήματα:
o___OP_WRD_OP__John_o([token:1]) is true.
o___OP_WRD_OP__Honest_o([token:1]) is true.

- Αφορούν χαρακτηριστικά της οντότητας *token:1*, όπου η οντότητα *token:1* έχει το χαρακτηριστικό «John» και το χαρακτηριστικό «Honest».

John

Hon
est

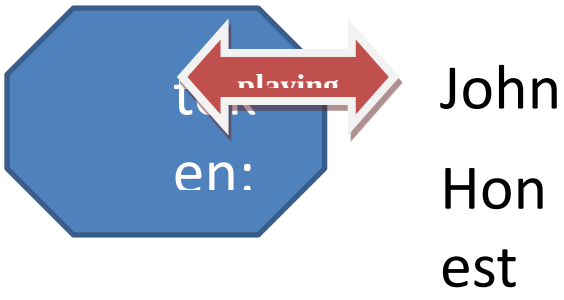
Το μοναδιαίο κατηγόρημα:
o___OP_WRD_OP__football_o([token:2]) is true.

- Αφορά χαρακτηριστικό της οντότητας *token:2*, όπου η οντότητα *token:2* έχει το χαρακτηριστικό «football».

foot
ball

Το δυαδικό κατηγόρημα:
o___OP_REL_OP__playing_o([token:1, token:2]) is true.

- Υποδηλώνει σχέση μεταξύ των οντοτήτων *token:1* και *token:2*, η οποία μας δείχνει ότι ο «Τίμιος Γιάννης παίζει ποδόσφαιρο» (Honest John playing football). Δηλαδή, είναι μια σχέση του υποκειμένου John με το αντικείμενο football, μέσω του ρήματος playing.

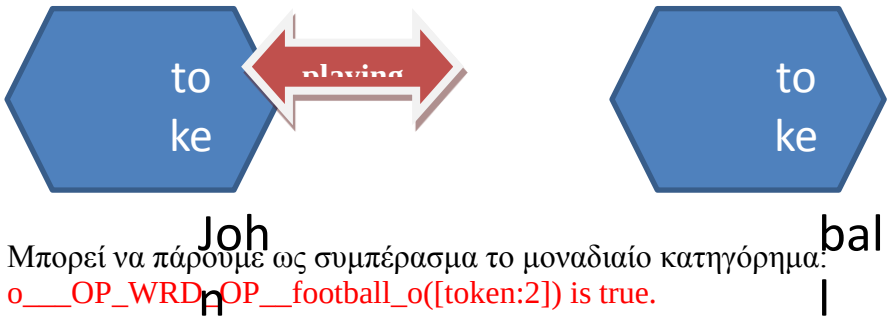


2.2.2 Συλλογισμοί

Ακολούθως, αφού ολοκληρωθεί η διαδικασία της μάθησης, μ
Reasoner μπορούμε να εξαγάγουμε μέσω συλλογισμών σ
λογικής υπό μορφή κατηγορημάτων.

Για παράδειγμα, αν εισάγουμε τη φυσική πρόταση «John is playing ball» και η οποία αντιστοιχεί στα κατηγορήματα:

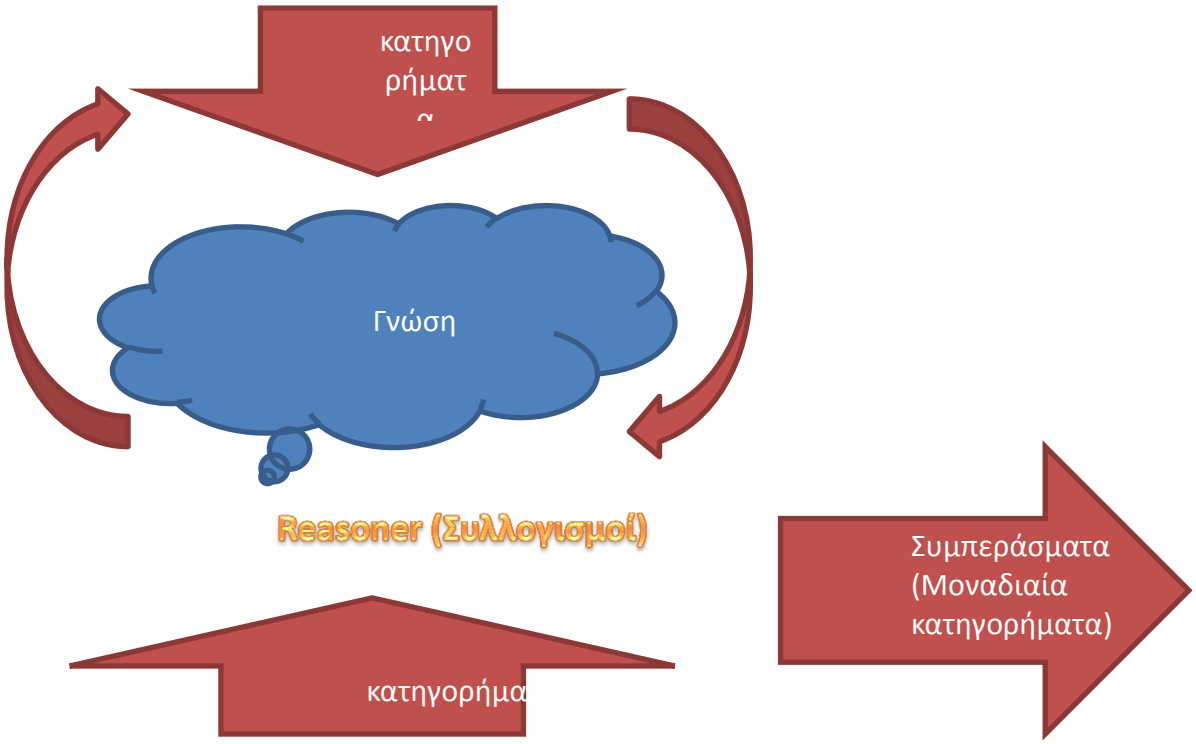
o___OP_WRD_OP__John_o([token:1]) is true.
o___OP_WRD_OP__ball_o([token:2]) is true.
o___OP_REL_OP__playing_o([token:1, token:2]) is true.



Μπορεί να πάρουμε ως συμπέρασμα το μοναδιαίο κατηγόρημα:
o___OP_WRD_OP__football_o([token:2]) is true.

Όπου πληροφορούμαστε ότι η οντότητα *token:2* έχει και το χαρακτηριστικό «football».

Δηλαδή, έχουμε το ακόλουθο σχήμα:



Σχήμα 2.2.2 | *Εξαγωγή συμπερασμάτων κοινής λογικής με το Reasoner.*

2.3 Μελέτη υπαρχουσών μηχανών αναζήτησης

Επειδή τελικός στόχος ήταν η δημιουργία μιας διαδικτυακής μηχανής συμπερασμάτων κοινής λογικής, η οποία θα συλλογιζόταν μέσα σε συγκεκριμένα περιθώρια χρόνου, έπρεπε να μελετήσουμε και τον τρόπο λειτουργίας υπαρχουσών μηχανών αναζήτησης, για να πληροφορηθούμε σχετικά με τον τρόπο λήψης ιστοσελίδων από το διαδίκτυο.

Μέσα από τη μελέτη του τρόπου λειτουργίας των μηχανών αναζήτησης Google, Lycos, AltaVista και Bing, φάνηκε ξεκάθαρα ότι προσκόμιζαν ιστοσελίδες από το διαδίκτυο, χρησιμοποιώντας *Web Crawlers* και τις αποθήκευαν τοπικά στο δίσκο. Αυτό γινόταν, γιατί ήθελαν να τις έχουν διαθέσιμες την ώρα που θα τις ζητούσε ο χρήστης, με την υποβολή κάποιου ερωτήματος, με κυριότερο στόχο τη μείωση του χρόνου αναζήτησης. Δηλαδή, κατά την υποβολή ερωτημάτων προς τις μηχανές, φιλτράρονται πρώτα οι ιστοσελίδες που βρίσκονται τοπικά στο δίσκο και αφού βρεθεί ποιες ιστοσελίδες ταιριάζουν, προτείνονται στη συνέχεια στους χρήστες.

Η μηχανή αναζήτησης ***Google***[4]***:***

1. Εντοπίζει και κατεβάζει ιστοσελίδες από το διαδίκτυο, με *crawler* που ονομάζεται *Googlebot*. Για να ξεκινήσει το ψάξιμο ιστοσελίδων, αρχικά, δίνεται στο *Googlebot* μια λίστα με διευθύνσεις ιστοσελίδων και ακολουθώντας τους υπερσύνδεσμους που περιέχει η κάθε μια, ξεκινάει τον έλεγχο και το κατέβασμα τους.
2. Κατά τον εντοπισμό και το κατέβασμα των ιστοσελίδων δημιουργεί ανάλογο κατάλογο (*indexing*), όπου παίζουν ρόλο και τα *title tags* και τα *alt attributes*, μέσω των οποίων ξεχωρίζει σε ποιο θέμα αναφέρεται η κάθε ιστοσελίδα.
3. Τέλος, κατά την επερώτηση του χρήστη, προτείνεται από τη μηχανή αναζήτησης συγκεκριμένη λίστα απαντήσεων, ξεκινώντας από αυτήν που έχει

[4] http://www.google.com/support/webmasters/bin/answer.py?answer=182072 [Μάιος 2011].

το μεγαλύτερο *pagerank* (αυτήν που δέχεται τους περισσότερους υπερσύνδεσμους από άλλες ιστοσελίδες).

Η μηχανή αναζήτησης ***Bing***[5]*:*

1. Εντοπίζει και κατεβάζει ιστοσελίδες από το διαδίκτυο, χρησιμοποιώντας τον *crawler* της που ονομάζεται *Msnbot* (σύντομα πρόκειται να αλλάξει όνομα).
2. Δημιουργεί κατάλογο και δίνει στην κάθε ιστοσελίδα ανάλογο *bing rank*. Η κάθε ιστοσελίδα παίρνει μεγαλύτερο *rank*, αναλόγως με το περιεχόμενο της (*meta-names*, *title tags* κτλ.), αλλά και με βάση τους υπερσύνδεσμους που οδηγούν σε αυτήν από τις άλλες ιστοσελίδες.
3. Κατά τη διάρκεια προσκόμισης του ερωτήματος του χρήστη, του εμφανίζει την ιστοσελίδα με το μεγαλύτερο *bing rank*.

Η μηχανή αναζήτησης ***Lycos***[6]*:*

1. Μέσω του *crawler* της δίνει ιδιαίτερο βάρος στο *url* της κάθε ιστοσελίδας, στο *meta-title* και γενικά στη συχνότητα εμφάνισης συγκεκριμένων λέξεων.
2. Επίσης, δίνει ιδιαίτερο βάρος στην ποσότητα του κειμένου που υπάρχει σε κάθε ιστοσελίδα και, συγκεκριμένα, ιστοσελίδες που περιέχουν αριθμό λέξεων μικρότερο του εβδομήντα πέντε δεν καταχωρούνται στον κατάλογο.
3. Τέλος, αγνοεί κείμενο που περιέχει ειδικούς χαρακτήρες όπως «$», «=», κτλ.

Η μηχανή αναζήτησης ***AltaVista***[7]*:*

1. Μέσω του *crawler* της, που ονομάζεται *Scooter*, αξιολογεί την κάθε ιστοσελίδα λαμβάνοντας υπόψη λέξεις, οι οποίες είναι τοποθετημένες στο πάνω μέρος της ιστοσελίδας αλλά και λέξεις που είναι στο κυρίως μέρος της (*body*).
2. Δίνει πολλή αξία στον τίτλο της ιστοσελίδας, αλλά και σε λέξεις που εμφανίζονται με έντονα γράμματα, πλάγια ή υπερσύνδεσμους.
3. Τέλος, λαμβάνει υπόψη και τον αριθμό άλλων ιστοσελίδων που οδηγούν στην τρέχουσα ιστοσελίδα που αξιολογεί, όπως γίνεται με το *pagerank* στη μηχανή αναζήτησης Google.

Έχοντας υπόψη τα πιο πάνω, προχωρήσαμε ακολούθως στη σχεδίαση και ανάπτυξη της Έξυπνης Μηχανής.

---

[5] http://help.live.com/Help.aspx?market=enUS&project=WL_Webmasters&querytype=topic&query=WL_WEBMASTERS_CONC_GettingSiteIndexed.htm [Μάιος 2011].

[6] http://www.webinformers.com/search_lycos_main.cfm [Μάιος 2011].

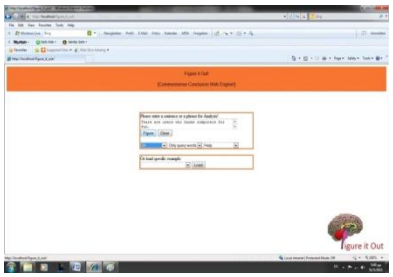
[7] http://websearch.about.com/od/enginesanddirectories/a/altavista.htm [Μάιος 2011].

3

Μεθοδολογία και Σχεδίαση

Πώς προχωρήσαμε στη σχεδίαση και ανάπτυξη της Έξυπνης Μηχανής;

Σε αυτό το κεφάλαιο παρουσιάζουμε τον τρόπο σχεδίασης της Έξυπνης Μηχανής, μέσω αλληλεπίδρασης με λογισμικά που αναπτύχθηκαν και αφορούσαν το Knowledge Infusion και ακολούθως παρουσιάζουμε τον τρόπο ανάπτυξης και λειτουργίας της.

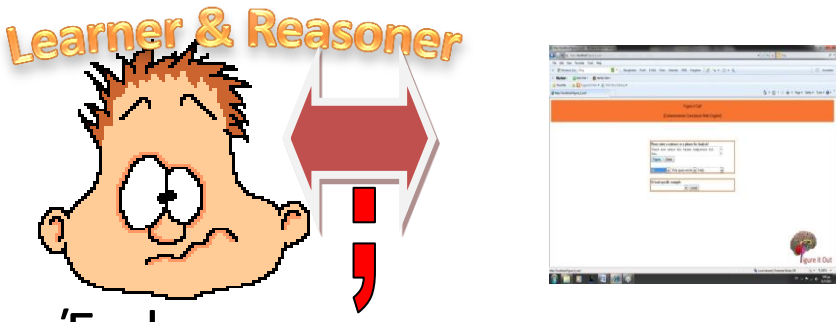


Έχοντας ολοκληρώσει την Ανασκόπηση Σχετικής Βιβλιογραφίας, καταλήξαμε στα εξής συμπεράσματα:

- Δεν εντοπίσαμε διαδικτυακή μηχανή συμπερασμάτων κοινής λογικής, η οποία να εξάγει το κρυμμένο νόημα μιας πρότασης ή φράσης.
- Η πλειοψηφία των συστημάτων που αναπτύχθηκαν απαιτούσαν, εκ των προτέρων, μεγάλη χειρονακτική δουλειά και, εκ των υστέρων, τα αποτελέσματα που εξήγαγαν δεν ήταν συμπεράσματα κοινής λογικής.
- Υπάρχει ανοικτό ερευνητικό πλαίσιο στο συγκεκριμένο χώρο, το οποίο σχετίζεται με έρευνα [12] όπου αναπτύχθηκαν λογισμικά μάθησης και συλλογισμού (Learner & Reasoner), πάνω στα οποία θα μπορούσαμε να στηρίξουμε την ανάπτυξη της Έξυπνης Μηχανής.

Αφού, με την ολοκλήρωση της διαδικασίας μάθησης μέσω του λογισμικού Learner, μπορούμε μέσω του λογισμικού Reasoner να εξάγουμε συμπεράσματα κοινής λογικής, θα μπορούσαμε να συνδυάσουμε την Έξυπνη Μηχανή με αυτά τα λογισμικά για να πετύχουμε την εξαγωγή συμπερασμάτων κοινής λογικής.

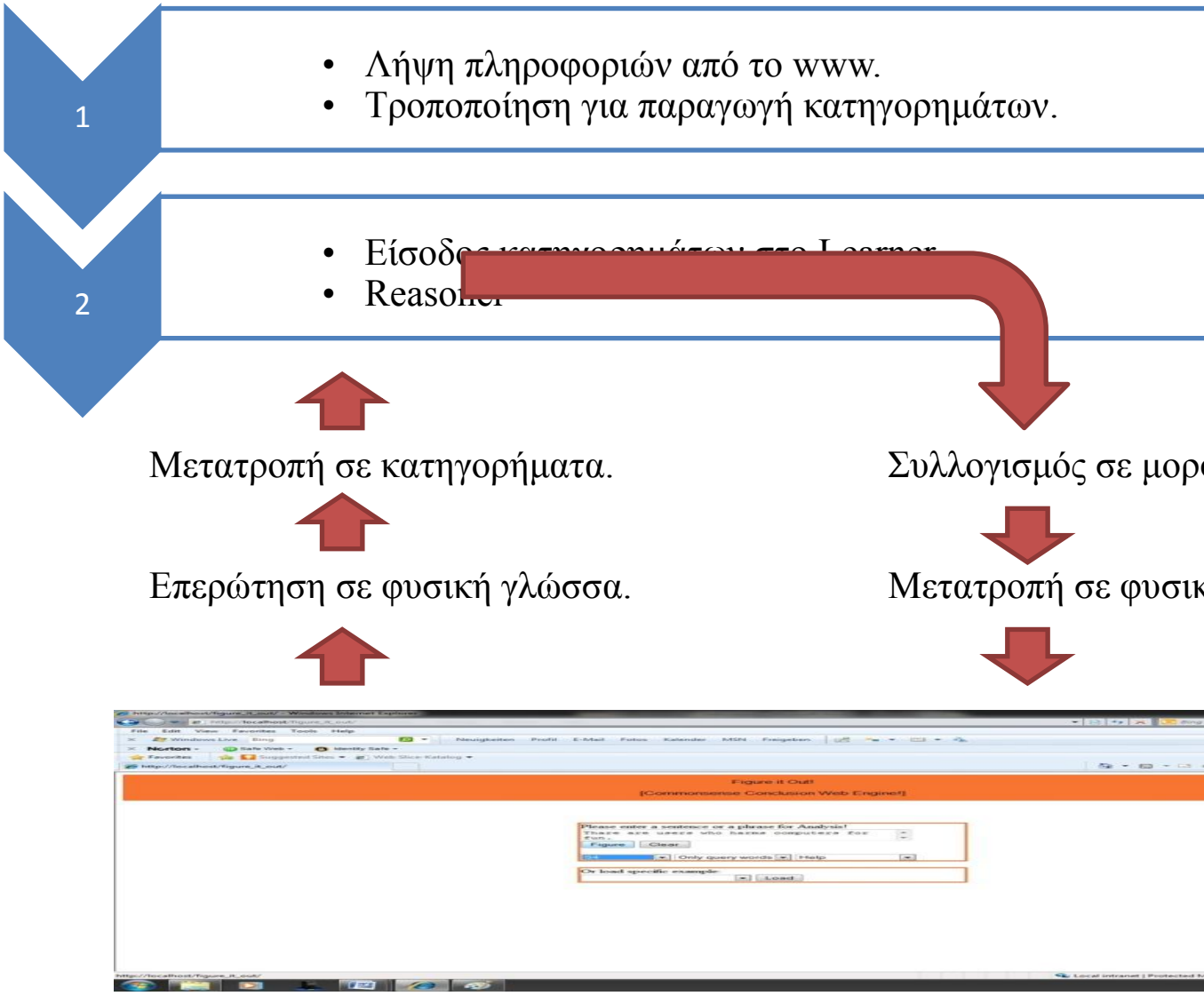
Όμως, πώς θα συνδυάζαμε την Έξυπνη Μηχανή με τα λογισμικά Learner και Reasoner;



Σχήμα 3.α | *Αναζητώντας τρόπο σύνδεσης.*

Ένας από τους στόχους ήταν και η αυτοματοποιημένη λήψη πληροφοριών από το διαδίκτυο. Σύμφωνα με τον τρόπο λειτουργίας αρκετών μηχανών αναζήτησης, αυτό θα έπρεπε να γίνει πριν τη διαδικασία επερώτησης του χρήστη, έτσι ώστε, κατά την ώρα της επερώτησης, να επιστρέφονται σε σύντομο χρονικό διάστημα τα συμπεράσματα κοινής λογικής. Έχοντας υπόψη τα πιο πάνω, αλλά και το ότι ο Learner και ο Reasoner δέχονται και επιστρέφουν κατηγορήματα, καταλήξαμε στη σχεδίαση του τρόπου λειτουργίας της Έξυπνης Μηχανής:

Λειτουργία Έξυπνης Μηχανής



Σχήμα 3.β | *Τρόπος λειτουργίας Έξυπνης Μηχανής.*

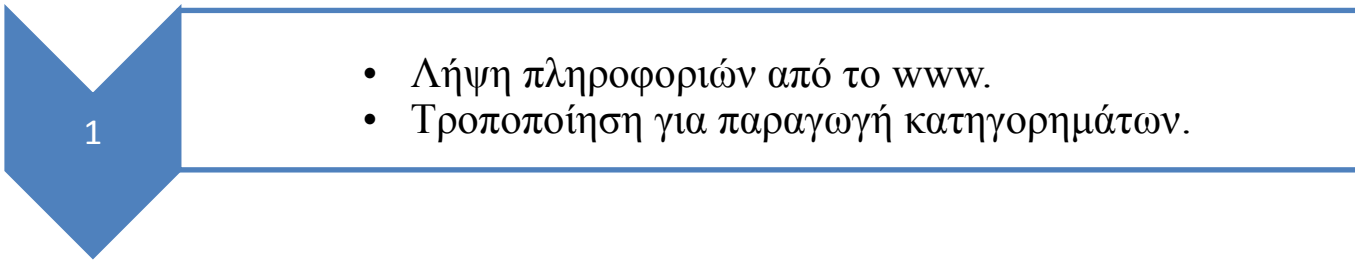
Σύντομη επεξήγηση του τρόπου λειτουργίας της Έξυπνης Μηχανής

- Αφού γίνει λήψη πληροφοριών από το *www*, τις μετατρέπουμε στη συνέχεια σε κατηγορήματα, γιατί τα λογισμικά Learner και Reasoner αυτού του είδους τις πληροφορίες δέχονται.
- Ακολούθως, δίνουμε τα κατηγορήματα στο Learner για το ξεκίνημα, μέχρι και την ολοκλήρωση της διαδικασίας της μάθησης (γνώση).
- Μέσω της Έξυπνης Μηχανής παίρνουμε την επερώτηση των χρηστών σε φυσική γλώσσα, τη μετατρέπουμε σε κατηγορήματα και τη δίνουμε στο Reasoner.
    - Με το Reasoner γίνονται οι απαραίτητοι συλλογισμοί και αν υπάρχουν συμπεράσματα κοινής λογικής επιστρέφονται σε μορφή κατηγορημάτων.

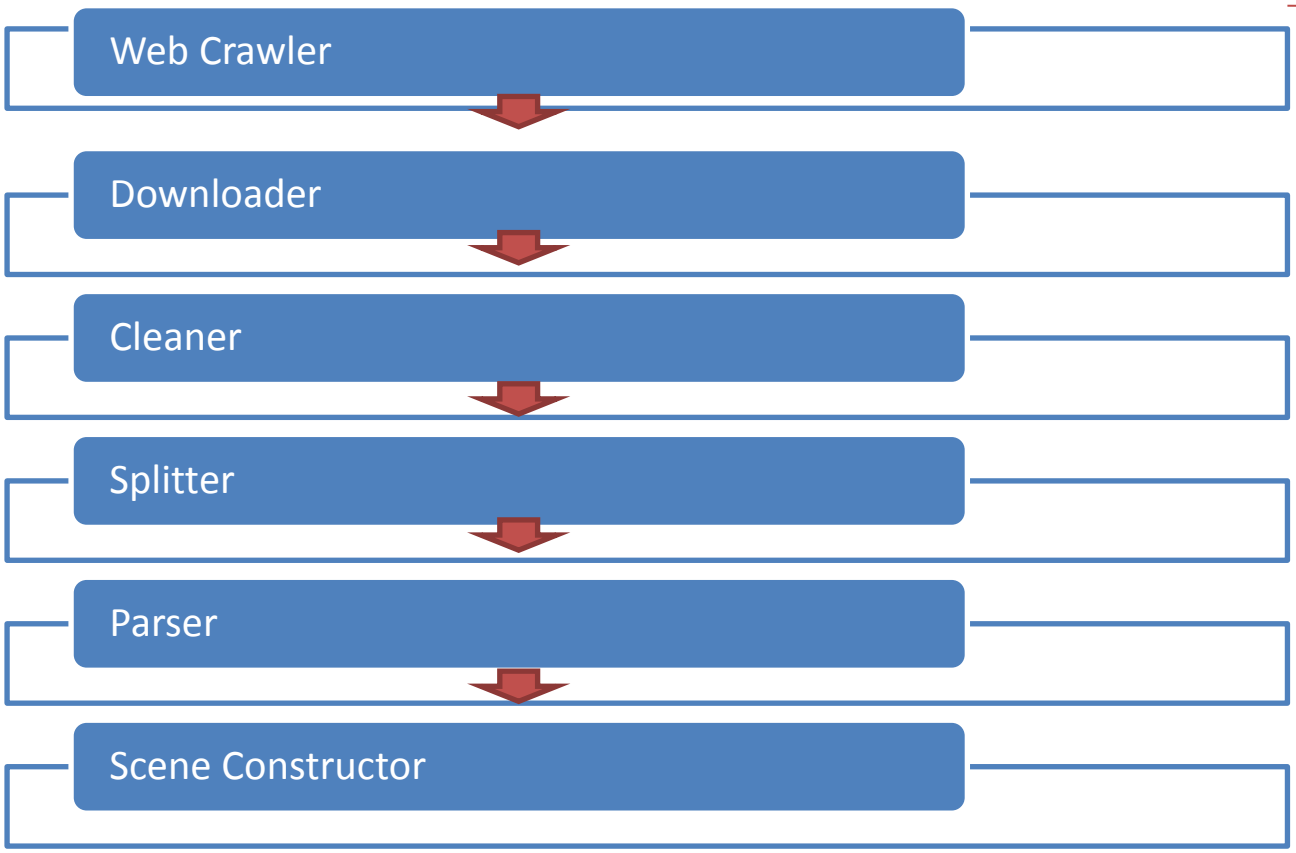
- Τέλος, γίνεται μετατροπή των κατηγορημάτων σε φυσική γλώσσα, για να επιστρέφονται πίσω στους χρήστες ως απάντηση προς την αρχική επερώτηση.

Αναλυτική επεξήγηση του τρόπου λειτουργίας της Έξυπνης Μηχανής

3.1 Πρώτο στάδιο [Από το κείμενο των ιστοσελίδων σε κατηγορήματα]



Αυτό το στάδιο αφορά τη λήψη πληροφοριών από ιστοσελίδες του *www* μέχρι και τη δημιουργία των κατηγορημάτων. Για το σκοπό αυτό έχουμε δημιουργήσει, αλλά και χρησιμοποιήσει λογισμικά, μέσω των οποίων λαμβάνουμε πληροφορίες από το *www* και αφού τις τροποποιήσουμε, παράγουμε δυαδικά και μοναδιαία κατηγορήματα.



Σχήμα 3.1.α | Από τις ιστοσελίδες στη δημιουργία κατηγορημάτων.

3.1.1 Web Crawler

Πρόκειται για λογισμικό το οποίο εντοπίζει ιστοσελίδες από το διαδίκτυο, ξεκινώντας από μία αρχική λίστα διευθύνσεων διαδικτύου. Επειδή βρισκόμαστε σε αρχικό ερευνητικό στάδιο, επιλέξαμε η μάθηση με το λογισμικό Learner να γίνεται γύρω από συγκεκριμένο θέμα (domain), μέσω του οποίου μπορούμε να ελέγχουμε την ορθότητα των συμπερασμάτων κοινής λογικής που επιστρέφει η Έξυπνη Μηχανή σε

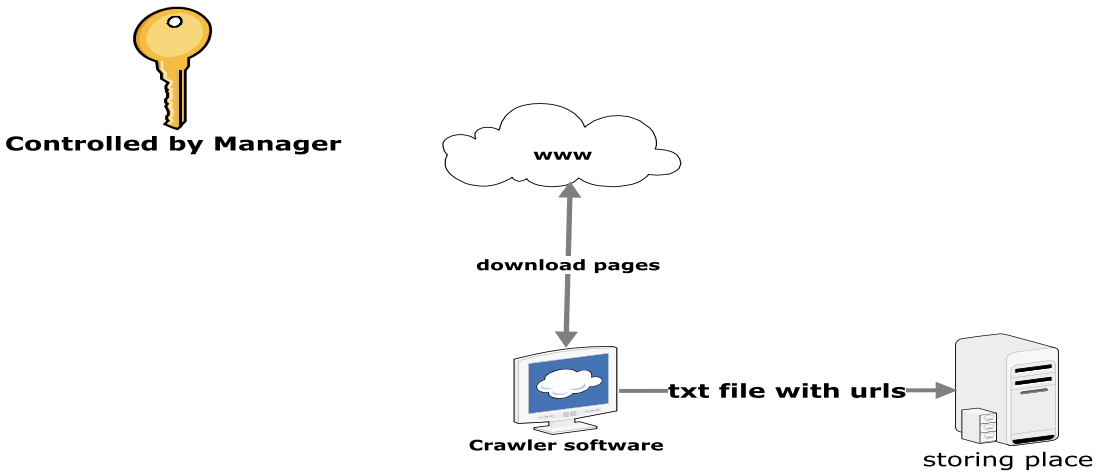
επερωτήσεις και για αυτό ο εντοπισμός ιστοσελίδων από το www σχετίζεται με συγκεκριμένο θέμα που καθορίζεται στο Web Crawler.

Αφού καθορίσουμε το θέμα με το οποίο θέλουμε να σχετίζονται οι ιστοσελίδες που θα εντοπιστούν, αλλά και τον επιθυμητό αριθμό διευθύνσεων ιστοσελίδων που θέλουμε, ο Web Crawler εντοπίζει, μέσω υπάρχουσας μηχανής αναζήτησης (π.χ. Google, Bing, Yahoo), έναν αριθμό διευθύνσεων (ο αριθμός καθορίζεται από το εμάς). Αυτό γίνεται για να μπορεί να ξεκινήσει η αναζήτηση ιστοσελίδων και, σαν εναλλακτική προσέγγιση, θα μπορούσαν αυτές οι διευθύνσεις να καταχωρηθούν δια χειρός. Ακολούθως, ο Web Crawler απενεργοποιεί τη χρήση της μηχανής αναζήτησης και μέσω αυτών των αρχικών διευθύνσεων, κάνοντας χρήση του αλγόριθμου A* [4], εντοπίζει διευθύνσεις που περιέχουν κείμενο, το οποίο σχετίζεται με το θέμα που εισάξαμε. Επίσης, μέσω της τεχνικής crc-32 [16], μέσω της οποίας ελέγχεται το περιεχόμενο της κάθε ιστοσελίδας, αποφεύγεται ο εντοπισμός ιστοσελίδων που περιέχουν πανομοιότυπο κείμενο. Τελειώνοντας, ο Web Crawler δημιουργεί μια λίστα με διευθύνσεις διαδικτύου, τις οποίες προτείνει για κατέβασμα και η οποία αποθηκεύεται σε ένα αρχείο κειμένου.

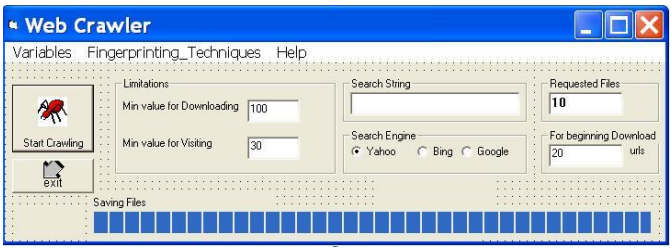
Λειτουργία μέσω του Αλγόριθμου A* [4]
Όταν προσκομίζεται μια ιστοσελίδα, τότε φιλτράρεται το περιεχόμενό της και εντοπίζουμε πόσες φορές εμφανίζεται το θέμα που μας ενδιαφέρει, με αποτέλεσμα την αξιολόγηση του περιεχομένου της ιστοσελίδας με μια συγκεκριμένη τιμή που φέρει το όνομα *value*. Για παράδειγμα, αν το θέμα μας είναι «Κύπρος», εντοπίζεται πόσες φορές εμφανίζεται η λέξη «Κύπρος» στην κάθε ιστοσελίδα, όπου ο αριθμός αυτός αντιστοιχεί στην τιμή *value* που δίνεται στην ιστοσελίδα. Ακολούθως, συγκρίνουμε την τιμή *value* με δύο προκαθορισμένες τιμές και συγκεκριμένα το *min_value_for_visiting*, που έχει τιμή τριάντα και το *min_value_for_downloading*, που έχει τιμή εκατό (οι τιμές μπορεί να αλλάζουν από εμάς). Κάθε ιστοσελίδα, η οποία αξιολογείται με τιμή τουλάχιστον ίση με τριάντα, θεωρείται σημαντική και για αυτό επισκεπτόμαστε και αξιολογούμε και τα παιδιά της (υπερσύνδεσμοι που υπάρχουν στην ιστοσελίδα). Κάθε ιστοσελίδα που αξιολογείται με τιμή τουλάχιστον εκατό, τοποθετείται σε λίστα για να προταθεί αργότερα για κατέβασμα. Η πιο πάνω διαδικασία επαναλαμβάνεται μέχρι και τον εντοπισμό *n* διευθύνσεων, με τιμή αξιολόγησης τουλάχιστον εκατό (όπου n ισούται με τον επιθυμητό αριθμό διευθύνσεων που αναζητούμε).

Λόγω του τρόπου λειτουργίας του αλγόριθμου A*, το λογισμικό του Web Crawler είναι το μόνο που τρέχει σειριακά και δεν τρέχει παράλληλα, μέσω πολλών στιγμιότυπων του ιδίου προγράμματος, όπως γίνεται και με τα άλλα λογισμικά που αναπτύχθηκαν και παρουσιάζονται στη συνέχεια. Για το σκοπό αυτό, με την ολοκλήρωση της διαδικασίας, η λίστα με τις διευθύνσεις τοποθετείται σε ένα αρχείο κειμένου, για να μπορέσει να ξεκινήσει η διαδικασία παράλληλου κατεβάσματος των ιστοσελίδων, αλλά και της γενικότερης επεξεργασίας των πληροφοριών μέχρι και τη δημιουργία των κατηγορημάτων.
Τεχνική Fingerprinting crc-32 [16]
Στο διαδίκτυο υπάρχουν αρκετές ιστοσελίδες με διαφορετική διεύθυνση, αλλά με το ίδιο περιεχόμενο. Αυτό μπορούσε να προκαλέσει προβλήματα, αφού ο Web Crawler, κάνοντας χρήση του A*, θα μπορούσε να κατέβαζε, σε αρκετές περιπτώσεις, σελίδες με διαφορετικό μεν *url*, αλλά με το ίδιο περιεχόμενο, πράγμα που σε τελικό στάδιο θα μας οδηγούσε στη δημιουργία πανομοιότυπων κατηγορημάτων. Ένας τρόπος για να

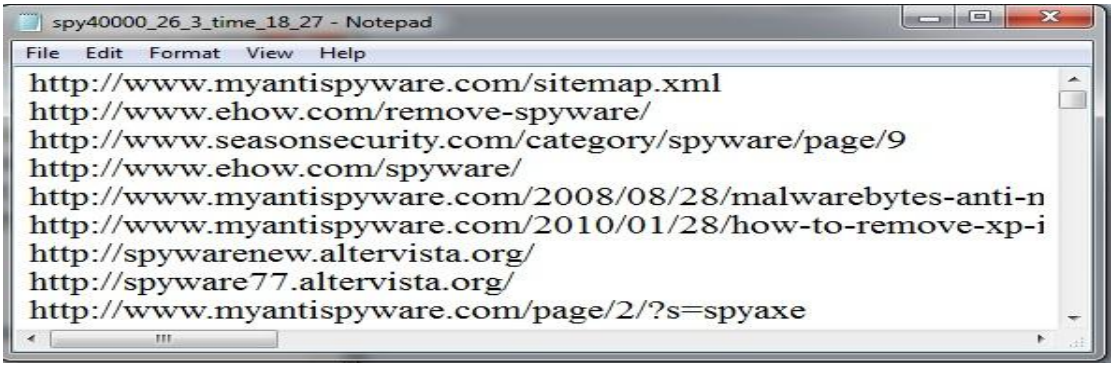
αντιμετωπιστεί αυτό, ήταν με τεχνικές *fingerprinting*. Αποφασίσαμε να χρησιμοποιήσουμε την τεχνική *cyclic redundancy check* (crc-32), η οποία είναι μια από τις πιο διαδεδομένες και η οποία αξιολογεί το περιεχόμενο κειμένου, παράγοντας και δίνοντας σε αυτή την ιστοσελίδα ένα δεκαεξαδικό κωδικό. Άρα, σε περίπτωση που εντοπίζεται μια νέα ιστοσελίδα με διαφορετικό *url* από τις υπάρχουσες που κατέβηκαν, τότε ενεργοποιείται ο crc-32, μέσω του οποίου παράγεται ένας δεκαεξαδικός κωδικός. Ακολούθως, ελέγχεται ένας πίνακας, ο οποίος περιέχει τους προηγούμενους δεκαεξαδικούς κωδικούς για τις υπόλοιπες ιστοσελίδες που ήδη αξιολογήθηκαν και αν εντοπιστεί ο ίδιος κωδικός, τότε αγνοούμε τη συγκεκριμένη ιστοσελίδα και προχωρούμε στον έλεγχο των υπολοίπων.



Σχήμα 3.1.1.α | *Εντοπισμός διευθύνσεων ιστοσελίδων από το www.*



Σχήμα 3.1.1.β | *Κύρια οθόνη Web Crawler.*

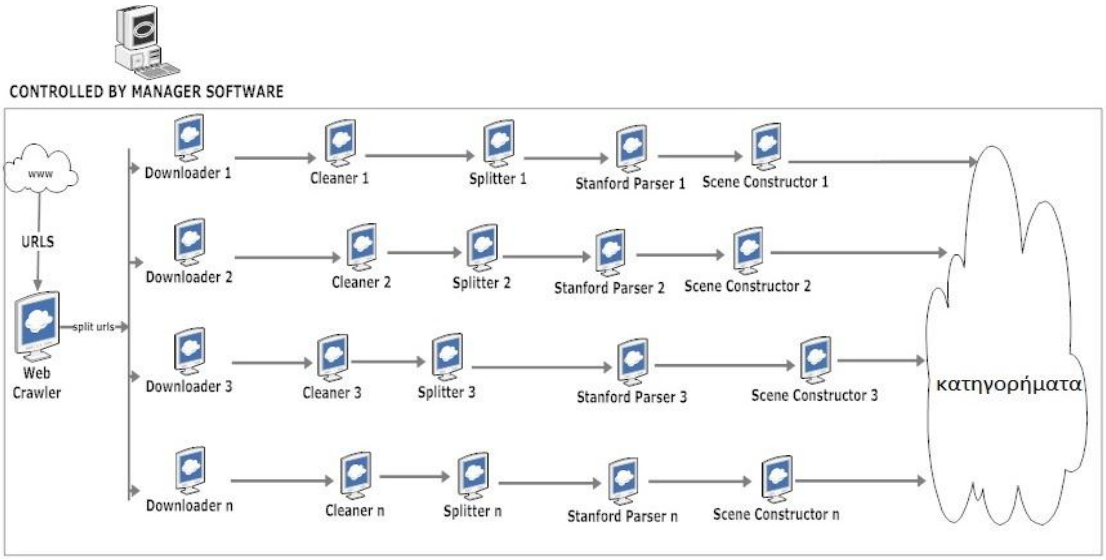


Σχήμα 3.1.1.γ | *Παράδειγμα αρχείου εξόδου του Web Crawler.*

3.1.2 Manager

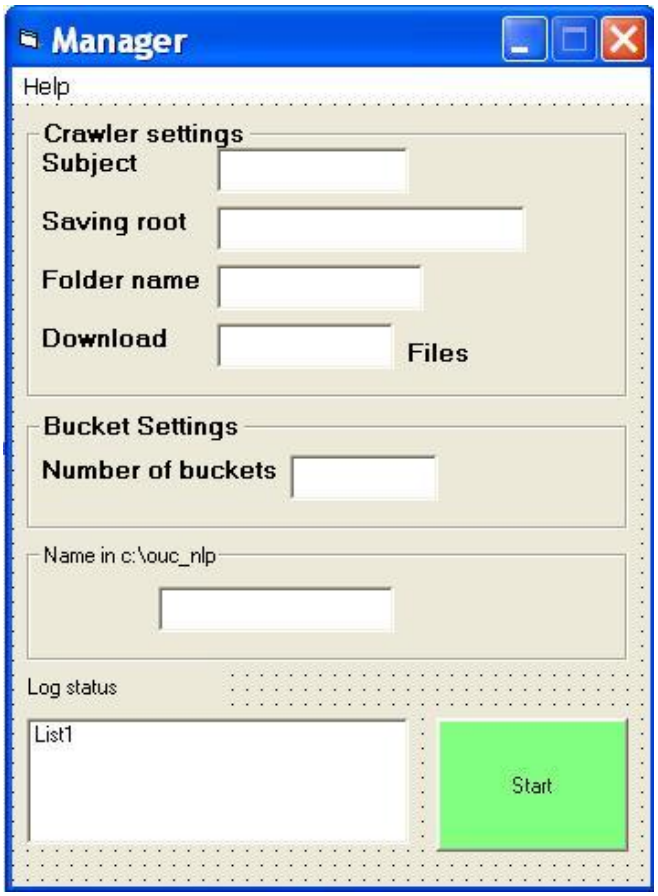
Το σύστημα σχεδιάστηκε για να τρέχει σε παράλληλους επεξεργαστές, για την εκμετάλλευση της ταχύτητας της παράλληλης επεξεργασίας. Από τον εντοπισμό ιστοσελίδων από το διαδίκτυο μέχρι και τη δημιουργία κατηγορημάτων, το μόνο λογισμικό που δεν τρέχει παράλληλα, όπως αναφέραμε προηγουμένως είναι ο Web Crawler.

Στο επόμενο σχήμα, παρουσιάζουμε στιγμιότυπο από τον παράλληλο σχεδιασμό μέρους του συστήματος, δείχνοντας τα λογισμικά που συμμετέχουν στη δημιουργία κατηγορημάτων με γρηγορότερους ρυθμούς. Για τη διεκπεραίωση της παράλληλης επεξεργασίας, αναπτύχθηκε λογισμικό, το οποίο αναλαμβάνει ρόλο διαχειριστή σε σχέση με την εκτέλεση των υπόλοιπων προγραμμάτων, με την ονομασία Manager.



Σχήμα 3.1.2.α | *Παράλληλη επεξεργασία για τη γρηγορότερη δημιουργία των κατηγορημάτων.*

Ο Manager ελέγχει όλα τα λογισμικά, ξεκινώντας με το κατέβασμα των ιστοσελίδων μέσω του Web Crawler και φτάνοντας μέχρι και τη δημιουργία κατηγορημάτων (*scenes*), μέσω κλήσης πολλών στιγμιότυπων του λογισμικού Scene Constructor. Το σημαντικότερο στοιχείο στο Manager είναι ο καθορισμός των *buckets* (όπως φαίνεται στο επόμενο σχήμα στην επιλογή *Number of buckets*). Μέσω της επιλογής αυτής, είναι σαν να καθορίζουμε το μέγιστο αριθμό στιγμιότυπων που θα τρέχουν παράλληλα την κάθε φορά. Για παράδειγμα, ένας αριθμός *bucket* να ισούται με δέκα, αυτό σημαίνει ότι θα δημιουργηθούν δέκα φάκελοι και θα τρέχει ανεξάρτητα δέκα φορές η αλληλουχία των προγραμμάτων Downloader->Cleaner->Splitter-> Stanford_Parser->Scene_Constructor μέχρι τη δημιουργία κατηγορημάτων στον κάθε φάκελο (bucket).

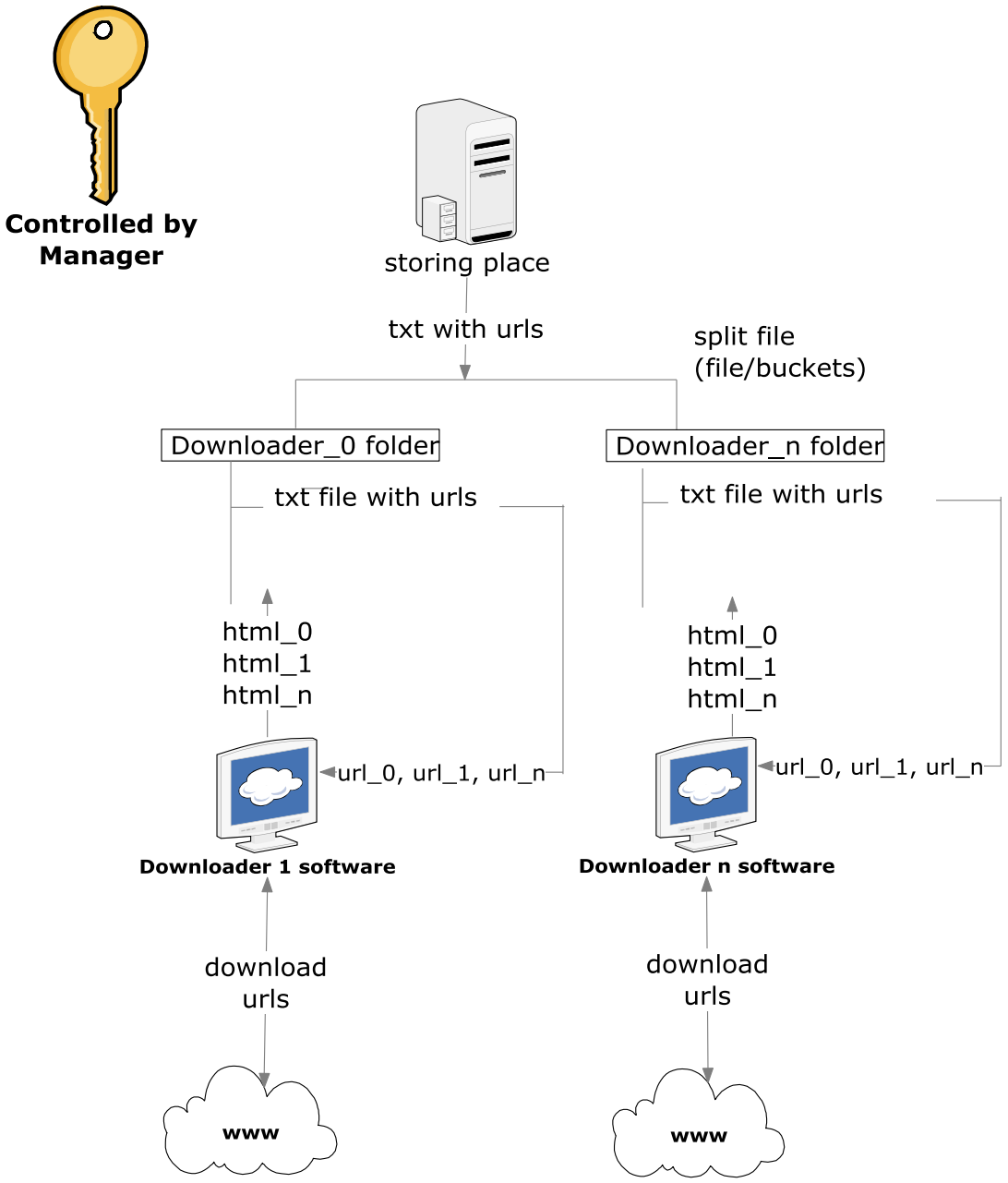


Σχήμα 3.1.2.β | *Κύρια οθόνη Manager.*

3.1.3 Downloader

Πρόκειται για λογισμικό, το οποίο κατεβάζει τις ιστοσελίδες που προτείνονται από το Web Crawler. Δημιουργήσαμε ξεχωριστό πρόγραμμα για το κατέβασμα ιστοσελίδων και δε βάλαμε αυτήν τη λειτουργικότητα στο Web Crawler, γιατί θέλαμε, έτσι, να πετύχουμε το παράλληλο κατέβασμα ιστοσελίδων. Με αυτόν τον τρόπο, μπορούμε να εκμεταλλευτούμε μεγάλες ταχύτητες κατεβάσματος ιστοσελίδων (download speed), μέσω πολλών στιγμιότυπων του λογισμικού, με αποτέλεσμα το κατέβασμα των ιστοσελίδων σε λιγότερο χρόνο. Επίσης, μπορεί να ξεκινήσει και ο διαμοιρασμός των εργασιών, μέσω του διαχειριστή Manager, για το ξεκίνημα της παράλληλης εκτέλεσης των άλλων λογισμικών.

Μόλις ολοκληρωθεί ο εντοπισμός των διευθύνσεων από το Web Crawler, μέσω του Manager γίνεται διαμοιρασμός του αρχείου με τις διευθύνσεις που εντοπίστηκαν, δημιουργώντας ένα αρχείο κειμένου με ανάλογο αριθμό διευθύνσεων σε κάθε *bucket.* Για παράδειγμα, αν έχουμε ένα αρχείο κειμένου με 200 διευθύνσεις ιστοσελίδων και 5 buckets τότε σε κάθε bucket θα δημιουργηθεί ένα αρχείο με 40 διευθύνσεις ιστοσελίδων (200/5=40). Επειδή, το πρώτο πρόγραμμα που ξεκινάει με την παράλληλη επεξεργασία είναι ο Downloader, για αυτόν το λόγο τα *buckets* έχουν ονομασίες του τύπου Downloader_xx (όπου xx ο αριθμός του κάθε *bucket*). Στη συνέχεια, για κάθε *bucket* καλείται αντίστοιχο στιγμιότυπο του λογισμικού Downloader, όπου αναλαμβάνει να κατεβάσει στο αντίστοιχο bucket ιστοσελίδες που οι διευθύνσεις τους περιέχονται στο αντίστοιχο αρχείο κειμένου (σχήμα 3.1.3.α).



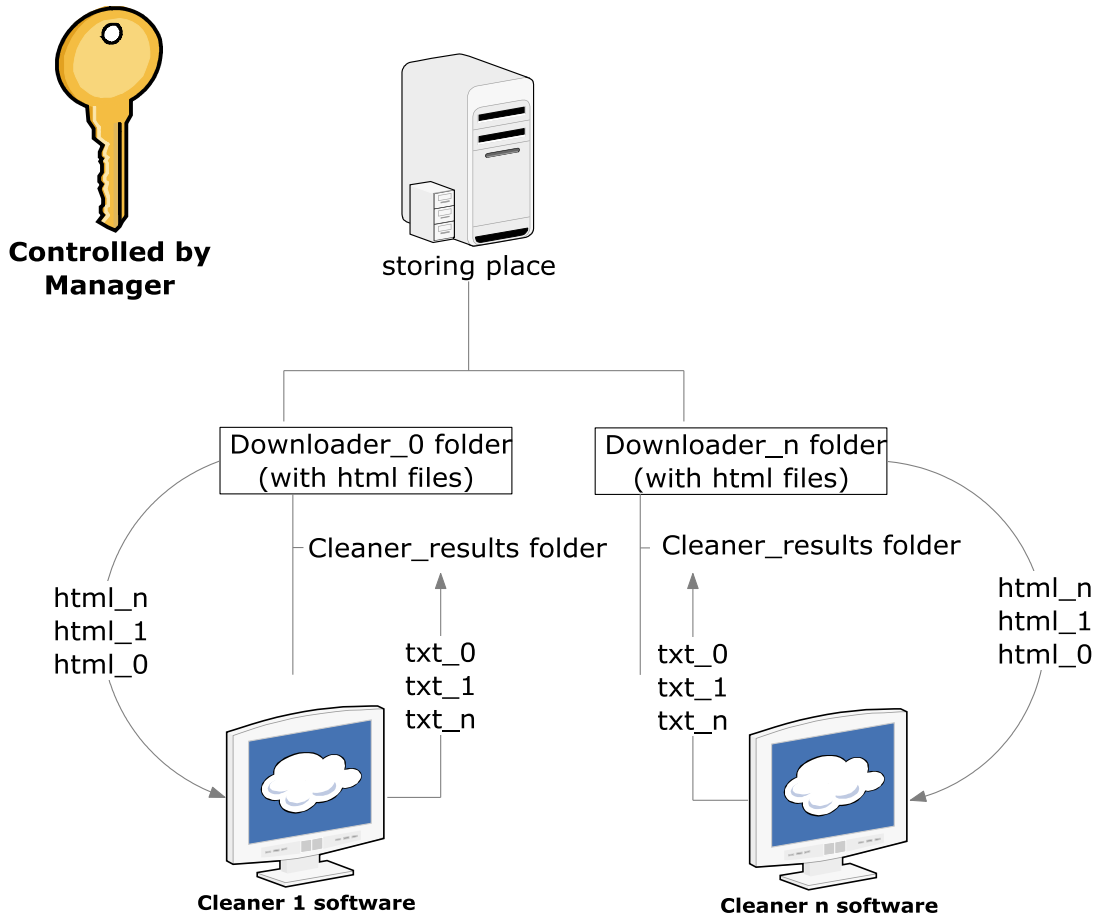
Σχήμα 3.1.3.α | *Κατέβασμα ιστοσελίδων από το διαδίκτυο.*

Σχήμα 3.1.3.β | *Κύρια οθόνη Downloader.*

3.1.4 Cleaner

Για να είναι δυνατή η δημιουργία κατηγορημάτων, πρέπει οι *html* σελίδες, οι οποίες είναι αποθηκευμένες στο δίσκο, να μετατρέπονται σε απλές σελίδες κειμένου, χωρίς την παρουσία *html tags*. Για τον καθαρισμό των *html tags* αναπτύχθηκε λογισμικό με την ονομασία Cleaner, το οποίο λαμβάνει τις *html* σελίδες και τις μετατρέπει, αφαιρώντας τα *html tags*, σε καθαρές σελίδες κειμένου (σχήμα 3.1.4.α). Και αυτό το λογισμικό τρέχει παράλληλα μέσω του διαχειριστή (Manager), για την ανάπτυξη υψηλών ταχυτήτων επεξεργασίας (μέσω της χρήσης πολλών Cleaners ταυτόχρονα). Συγκεκριμένα, για κάθε *bucket* που περιέχει τα αρχεία που κατέβηκαν από τους Downloaders, τρέχουμε ισάριθμους Cleaners και τα αποτελέσματα αποθηκεύονται σε καινούργιο υποφάκελο, που φέρει την ονομασία *Cleaner_results* (σχήμα 3.1.4.β).

Σχήμα 3.1.4.α | Παράδειγμα μετατροπής αρχείου html σε αρχείο κειμένου μέσω του Cleaner.

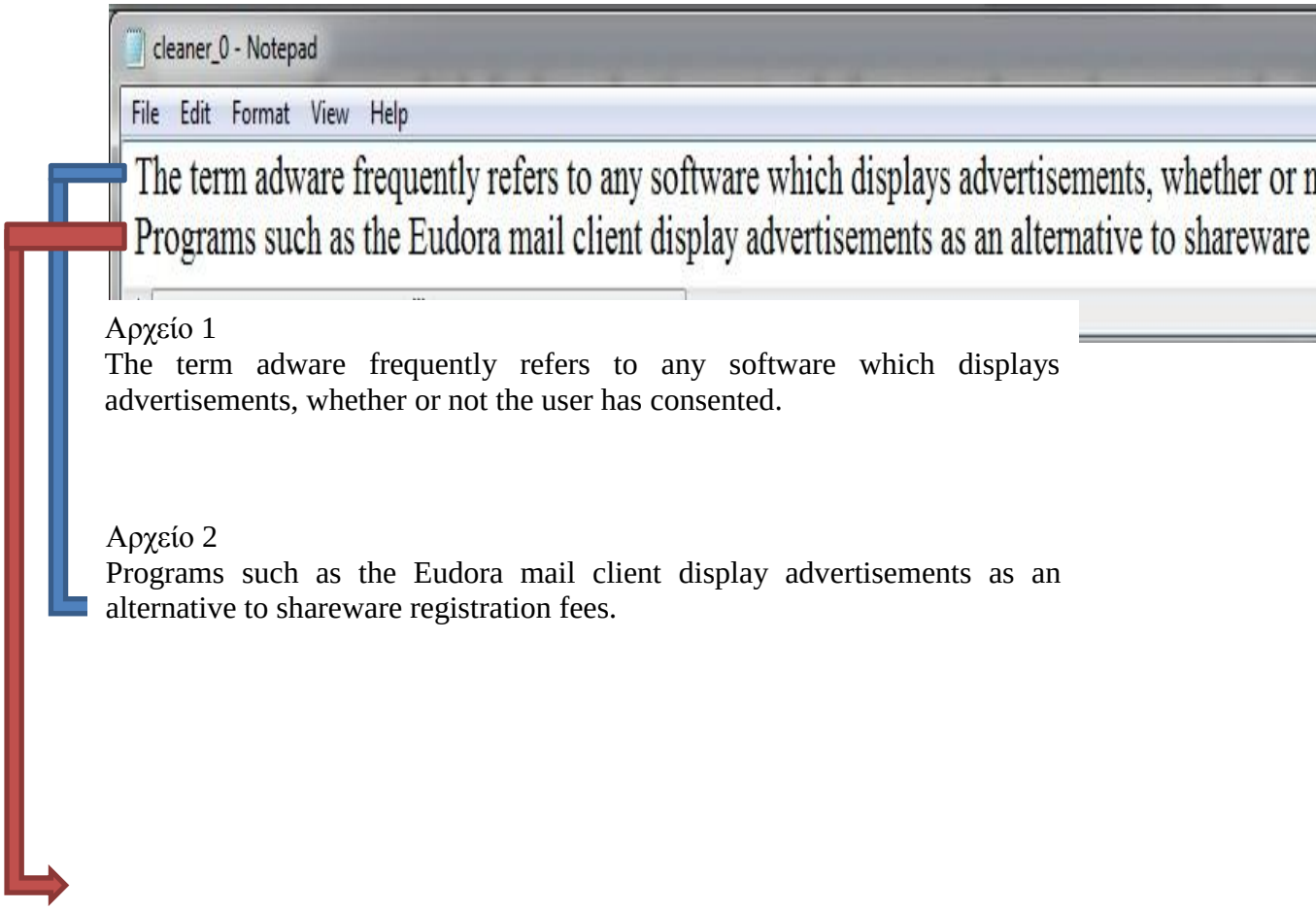


Σχήμα 3.1.4.β | *Μετατροπή αρχείων html σε αρχεία κειμένου.*

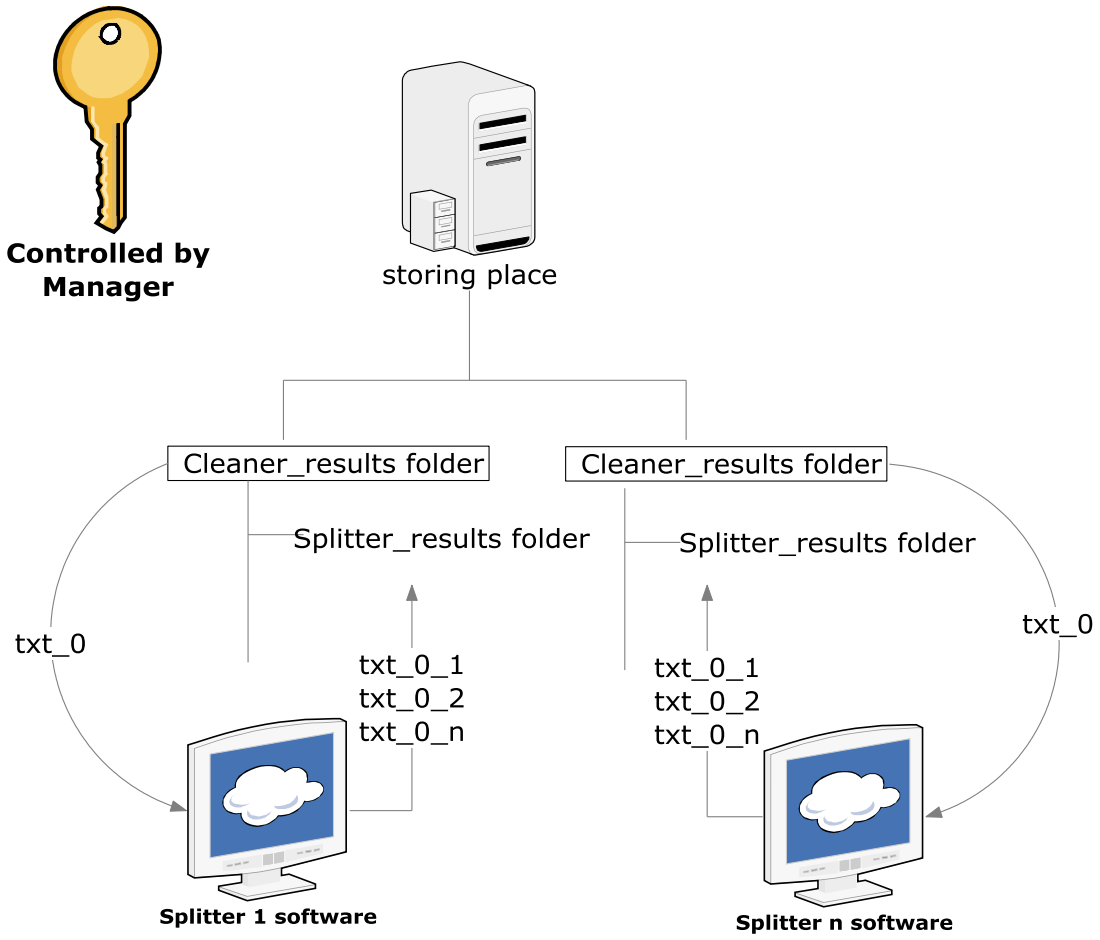
Σχήμα 3.1.4.γ | *Κύρια οθόνη Cleaner.*

3.1.5 Splitter

Για να είναι δυνατό το *parsing* των αρχείων, μέσω συγκεκριμένου *parser* (βλέπε 3.1.6 Parser, σελίδα 30), πρέπει κάθε αρχείο που υπάρχει στο δίσκο, να μοιράζεται σε περισσότερα μικρότερα αρχεία. Αυτό, πρέπει να γίνει λόγω του άμεσου συσχετισμού που υπάρχει μεταξύ μεγέθους του κάθε αρχείου και αντίστοιχης ποσότητας κύριας μνήμης που χρειάζεται ο συγκεκριμένος *parser*. Για αυτό το λόγο αναπτύξαμε λογισμικό που κάνει αυτό το διαμοιρασμό, με την ονομασία Splitter. Συγκεκριμένα, μπορεί να γίνει διαμοιρασμός κάθε αρχείου ή ανά παράγραφο, όπου για κάθε παράγραφο δημιουργείται ένα καινούργιο αρχείο ή ανά πρόταση, όπου για κάθε πρόταση του αρχικού αρχείου δημιουργείται ένα καινούργιο αρχείο.



Σχήμα 3.1.5.α | *Παράδειγμα διαμοιρασμού αρχείου μέσω του Splitter.*



Σχήμα 3.1.5.β | *Διαμοιρασμός αρχείων σε περισσότερα αρχεία.*

Σχήμα 3.1.5.γ | *Κύρια οθόνη Splitter.*

3.1.6 Parser

Έπρεπε να βρούμε ή να δημιουργήσουμε ένα λογισμικό (parser), το οποίο θα δημιουργούσε κανόνες συσχετισμού μεταξύ των λέξεων των διαφόρων κειμένων, για να μπορέσουμε να προχωρήσουμε στο επόμενο στάδιο που ήταν η δημιουργία κατηγορημάτων. Έχοντας δει τον τρόπο με τον οποίο διαβάζουμε τα κατηγορήματα, τα οποία εισάγονται στο Learner (βλέπε Ανασκόπηση Σχετικής Βιβλιογραφίας, 2.2.1 Μάθηση, σελίδα 11) και αφού αυτά, για παράδειγμα, μπορεί να συσχετίζουν υποκείμενα με αντικείμενα μέσω του άμεσου ρήματος που τα συνδέει, έπρεπε να έχουμε κάποιο λογισμικό, το οποίο θα μας εξήγαγε πληροφορίες για τα μέρη του λόγου των λέξεων, αλλά και για το πώς συνδέονται οι λέξεις σε μια πρόταση (ποιο είναι το υποκείμενο, το αντικείμενο, το ρήμα, το κατηγόρημα κτλ.). Στη συνέχεια, μέσω αυτών των πληροφοριών, θα προχωρούσαμε προς τη δημιουργία των κατηγορημάτων. Επίσης, σχετική έρευνα [9] που έγινε για τη διευκόλυνση δημιουργίας κατηγορημάτων χρησιμοποιεί *parser*, μέσω του οποίου συσχετίζονται οι λέξεις σε σχέση με το μέρος του λόγου τους και εξάγονται αποτελέσματα της μορφής X (s-1, q-2) όπου το s και το q είναι λέξεις και το X ένας τύπος που υποδηλώνει διάφορες συσχετίσεις, όπως π.χ. συσχέτιση υποκειμένου και ρήματος (όπου το q είναι το υποκείμενο και το s το ρήμα).

Παράδειγμα:

X y t z.

Π.χ. Μια πρόταση όπου τα σύμβολα αντιστοιχούν σε λέξεις.

Αν ξέραμε το μέρος του λόγου της κάθε μιας λέξης, π.χ.:
X=ουσιαστικό
y=ρήμα
t=ρήμα
z=ουσιαστικό

Και αν παίρναμε και πληροφορίες για το πώς συνδέονται μεταξύ τους (εξαρτήσεις) για το ποιο είναι το υποκείμενο, το αντικείμενο κτλ., π.χ.:
(X, t) υποκείμενο με ρήμα
(t, z) ρήμα με αντικείμενο

Θα μπορούσαμε να δημιουργήσουμε δυαδικά και μοναδιαία κατηγορήματα.

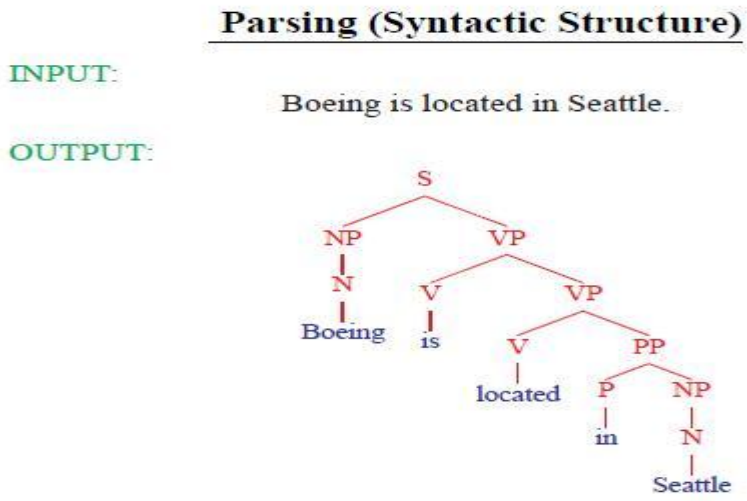
o___OP_WRD_OP__X_o([token:1]) is true.
o___OP_WRD_OP__z_o([token:2]) is true.
o___OP_REL_OP__t_o([token:1, token:2]) is true.

Σχήμα 3.1.6.α | Επεξήγηση χρησιμότητας των μερών του λόγου των λέξεων για τη μετέπειτα παραγωγή κατηγορημάτων.

Στη συνέχεια, προχωρήσαμε στην αναζήτηση *parser* όπου έγινε αρκετή έρευνα με αρκετά προβλήματα μέχρι να φτάσουμε στο τελικό αποτέλεσμα. Ας αναφέρουμε με λίγα λόγια ολόκληρη τη διαδικασία.

Γενικά, υπάρχουν δύο κατηγορίες *parsers* και συγκεκριμένα, δομής (structural) και συσχετισμού (dependency) [9].

Δομής
Οι *parsers* που ανήκουν σε αυτήν την κατηγορία δίνουν έμφαση στην επεξήγηση των λέξεων, ανάλογα με το σε ποια συντακτική οικογένεια ανήκουν (*part of speech*) και συγκεκριμένα, αν οι λέξεις είναι ουσιαστικά, ρήματα, επίθετα, κτλ. Παρουσιάζουν τα αποτελέσματα και με τη μορφή δέντρου, όπου στα φύλλα είναι οι λέξεις και στους εσωτερικούς κόμβους είναι οι ονοματικές φράσεις, οι ρηματικές φράσεις κτλ.



Σχήμα 3.1.6.β | Παράδειγμα εξόδου parser δομής.

Συσχετισμού
Οι *parsers* που ανήκουν σε αυτήν την κατηγορία δίνουν έμφαση στη δημιουργία συσχετίσεων ανάμεσα στις λέξεις και παρουσιάζουν αυτήν την πληροφορία και υπό μορφή δέντρου. Στα φύλλα και στους εσωτερικούς κόμβους είναι οι λέξεις της πρότασης ή παραγράφου μαζί με το μέρος του λόγου τους και στα μονοπάτια βρίσκονται οι συσχετίσεις ανάμεσα στις λέξεις που εμφανίζουν πληροφορίες που μας ενδιαφέρουν (αντικείμενα, ποιητικά αίτια, εμπρόθετους προσδιορισμούς, υποκείμενα κτλ.). Εμείς, χρειαζόμαστε ένα *dependency parser*, ο οποίος θα εξήγαγε τις συσχετίσεις ανάμεσα στις λέξεις σε κάποιο αρχείο, για να μπορούμε να τις λαμβάνουμε με εύκολο τρόπο για τη δημιουργία κατηγορημάτων.

Έγινε έρευνα για διάφορα εργαλεία, που σχετίζονταν με το *parsing* και εντοπίστηκαν οι εξής *parsers*:

- Charniak Parser
- Chunk Parser
- Link Grammar Parser
- Relex Parser
- Srl Parser
- Proxem Antelope
- Stanford NLP Parser

Charniak Parser [8] *[1]*
Πρόκειται για *parser* δομής, με ποσοστά επιτυχίας γύρω στο 80%. Επειδή χρειαζόμαστε *dependency parser*, απορρίφθηκε η επιλογή του. Επίσης, ο συγκεκριμένος *parser* τρέχει μόνο σε λειτουργικό *Linux*, ενώ οι εφαρμογές μας αναπτύχθηκαν για λειτουργικό *Windows*.

Chunk Parser[9] *[19]*
Πρόκειται για *parser δομής* αρκετά γρήγορο και αξιόπιστο, με ποσοστά επιτυχίας γύρω στο 80%. Τρέχει σε πλατφόρμα *Windows* και *Linux*, αλλά, επειδή δεν εξάγει συσχετισμούς μεταξύ των λέξεων, απορρίφθηκε και αυτός.

Link Grammar Parser[10] *[6]*
Πρόκειται για *dependency parser* που διατίθεται για *Linux* και για *Windows*, αλλά, λόγω προβλημάτων στην εγκατάσταση δεν καταφέραμε να πειραματιστούμε μαζί του σε περιβάλλον *Windows*.

Relex Parser[11] *[3]*
Πρόκειται για *dependency parser* που στηρίζεται στο *Link Grammar Parser*, αλλά βρίσκεται σε στάδιο ανάπτυξης. Δεν μπορέσαμε όμως να το χρησιμοποιήσουμε, γιατί δεν τρέχει σε λειτουργικό *Windows*.

Srl Parser [12] *[17]*
Πρόκειται για *dependency parser*, ο οποίος χρησιμοποιεί εμμέσως το Charniak Parser. Λόγω του ότι δε βρήκαμε έκδοση για *Windows*, δε χρησιμοποιήσαμε ούτε αυτόν τον *parser*.

Proxem Antelope[13]
Πρόκειται για ένα εμπορικό πακέτο εργαλείων που περιέχει διάφορους *parsers* (Charniak, Stanford κτλ.) και που εξάγει πολύ καλούς συσχετισμούς λέξεων. Τα αποτελέσματα, όμως, δεν εξάγονται σε αρχείο και έτσι δε θα μπορούσε να χρησιμοποιηθεί στην περίπτωσή μας, γιατί θέλουμε αυτοματοποιημένο *parsing* για χιλιάδες προτάσεις.

Stanford NLP Parser[14] *[5]*
Πρόκειται για *dependency parser*, ο οποίος αναπτύχθηκε από το πανεπιστήμιο *Stanford* και έχει τα εξής πλεονεκτήματα:

- Είναι γραμμένος σε *java*, πράγμα που επιτρέπει το τρέξιμο του σε όλα τα λειτουργικά συστήματα.
- Εκτός από την εξαγωγή των αποτελεσμάτων σε δεντρική μορφή, εξάγει και καθαρά δυαδική μορφή συσχετίσεων, τις οποίες μπορούμε να επεξεργαζόμαστε πιο εύκολα.
- Μπορεί να εξάγει τα αποτελέσματα για κάθε αρχείο εισόδου σε αντίστοιχο αρχείο εξόδου. Αυτό είναι πάρα πολύ σημαντικό πλεονέκτημα, γιατί

[8] http://www.cs.brown.edu/~ec/#software [Μάιος 2011].
[9] http://www-tsujii.is.s.u-tokyo.ac.jp/~tsuruoka/chunkparser/ [Μάιος 2011].
[10] http://www.link.cs.cmu.edu/link/index.html [Μάιος 2011].
[11] http://opencog.org/wiki/RelEx [Μάιος 2011].
[12] http://l2r.cs.uiuc.edu/ [Μάιος 2011].
[13] http://www.proxem.com/Antelope.aspx [Μάιος 2011].
[14] http://nlp.Stanford.edu/software/lex-parser.shtml [Μάιος 2011].

μπορούμε να το αξιοποιήσουμε στο μέγιστο, για να αυτοματοποιήσουμε ολόκληρη τη διαδικασία.

- Εξάγει και συσχετίσεις (collapsed dependencies) οι οποίες σπάζουν τη δομή του δέντρου, μετατρέποντάς την σε ένα κατευθυνόμενο κυκλικό γράφο, όπου αυτού του τύπου οι συσχετισμοί είναι πολύ χρήσιμοι στη δημιουργία κατηγορημάτων [10].

Λόγω των πιο πάνω, χρησιμοποιούμε αυτόν τον *parser* για επεξεργασία των αρχείων που παράχθησαν με το Splitter για παραγωγή συσχετίσεων μεταξύ των λέξεων.

Ένα παράδειγμα μέσα από το Stanford NLP Parser

Ο *parser* δέχεται μια πρόταση ή μια φράση σαν είσοδο και εξάγει τους συσχετισμούς (dependencies), που υπάρχουν στη συγκεκριμένη πρόταση ή φράση. Μαζί, αν θέλουμε, μπορούμε να εξάγουμε και το μέρος του λόγου κάθε λέξης (δηλαδή, αν η λέξη είναι ουσιαστικό, ή επίθετο ή ρήμα κτλ.).

Ας δούμε ένα παράδειγμα μέσα από την εισαγωγή της ακόλουθης πρότασης:
Fireade 2000 eliminates the complexity of choosing the right firefighting foam agent.

```
Fireade/NNP 2000/CD eliminates/VBZ the/DT complexity/NN of/IN choosing/VBG the/DT right/JJ firefighting/NN foam/NN agent/NN ./.

nsubj(eliminates-3, Fireade-1)
num(Fireade-1, 2000-2)
det(complexity-5, the-4)
dobj(eliminates-3, complexity-5)
prepc_of(complexity-5, choosing-7)
det(agent-12, the-8)
amod(agent-12, right-9)
nn(agent-12, firefighting-10)
nn(agent-12, foam-11)
dobj(choosing-7, agent-12)
```

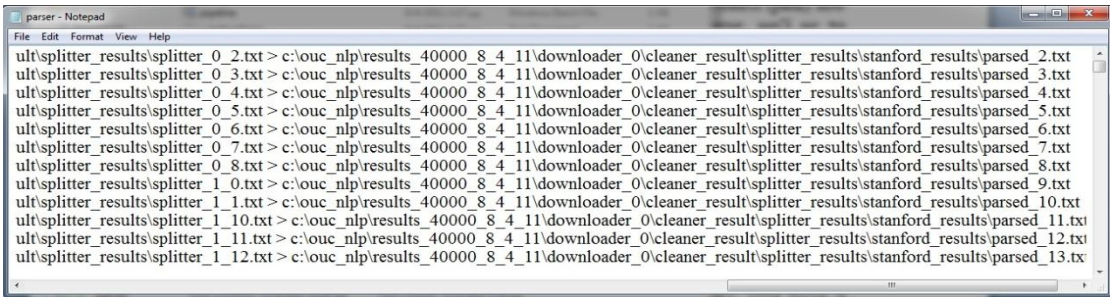
Σχήμα 3.1.6.γ | *Παράδειγμα εξόδου του Stanford Parser.*

Ως *dependency parser*, εξάγει τα αποτελέσματά του σε μορφή δυαδικών συσχετίσεων (π.χ. nsubj, dobj, det, prepc, nn κτλ.), οι οποίες, με τη σειρά τους, συσχετίζουν συντακτικά τις λέξεις σε σχέση με το μέρος του λόγου της κάθε μιας. Για παράδειγμα, η συσχέτιση *dobj* υποδηλώνει τη σχέση μεταξύ άμεσου αντικειμένου και ρήματος, ενώ η *nsubj* υποδηλώνει τη σχέση μεταξύ υποκειμένου και ρήματος (βλέπε Παράρτημα, Εγχειρίδιο επεξήγησης δυαδικών συσχετίσεων Stanford Parser, σελίδα 97).

Κατά την εξαγωγή των συσχετίσεων βλέπουμε και το μέρος του λόγου της κάθε λέξης (*part of speech*), το οποίο χρειαζόμαστε για να μπορούμε να καταλαβαίνουμε τη σημασία της κάθε λέξης που συμμετέχει σε κάθε συσχέτιση. Παράδειγμα, αν γνωρίζουμε και το μέρος του λόγου της κάθε λέξης, θα μπορούμε να ξέρουμε ότι οι λέξεις που συμμετέχουν π.χ. στη συσχέτιση dobj (eliminates-3, complexity-5) είναι *VBZ* και *NN*, δηλαδή, με απλά λόγια, είναι μια συσχέτιση ανάμεσα σε ένα ρήμα και ένα ουσιαστικό.

Ο *parser* δέχεται αρχεία με προτάσεις ή φράσεις και εξάγει ανάλογο αριθμό αρχείων με τους συσχετισμούς των λέξεων. Η όλη λειτουργία γίνεται εύκολα μέσω συγκεκριμένου *batch file*, στο οποίο περιέχονται τα μονοπάτια των αρχείων που θα γίνουν *parsed* μαζί με τα μονοπάτια των αρχείων εξόδου.

Για παράδειγμα, στην επόμενη εικόνα, σε κάθε γραμμή υπάρχει το μονοπάτι (path) που οδηγεί σε αρχείο που περιέχει πρόταση που θέλουμε να γίνει *parsed*, μαζί με το μονοπάτι του αρχείου που θα δημιουργηθεί και θα περιέχει την έξοδο του *parser* (με τους συσχετισμούς και τα μέρη του λόγου των λέξεων).



Σχήμα 3.1.6.δ | Παράδειγμα περιεχομένων ενός batch file του Stanford Parser.

Για να επιτύχουμε παράλληλη εκτέλεση, δημιουργήσαμε λογισμικό με το όνομα Modificator, το οποίο δημιουργεί ανάλογο *batch file* του Stanford Parser σε κάθε *bucket (*που δημιουργείται με το λογισμικό Manager) και με το οποίο καλείται ο *parser* για την εξαγωγή δυαδικών συσχετίσεων. Ο Modificator, ανάμεσα στα παράλληλα στάδια της επεξεργασίας, μπορεί να δημιουργεί *n batch files*, μέσα από τα οποία να καλούνται *n x p parsers* συνολικά για τη δημιουργία δυαδικών συσχετίσεων (όπου n ισούται με τον αριθμό των buckets και p ισούται με τον αριθμό των γραμμών του κάθε batch file). Για παράδειγμα, το αρχείο του σχήματος 3.1.6.δ αναφέρεται σε ένα *batch file* ενός *bucket* και περιέχει εκατό γραμμές, πράγμα που σημαίνει ότι θα καλεστεί σειριακά εκατό φορές ο Stanford Parser. Αν έχουμε συνολικά δέκα *buckets* που το καθένα περιέχει ένα *batch file* με εκατό γραμμές, σημαίνει ότι συνολικά θα τρέξουν χίλια στιγμιότυπα του *parser* (10 x 100).

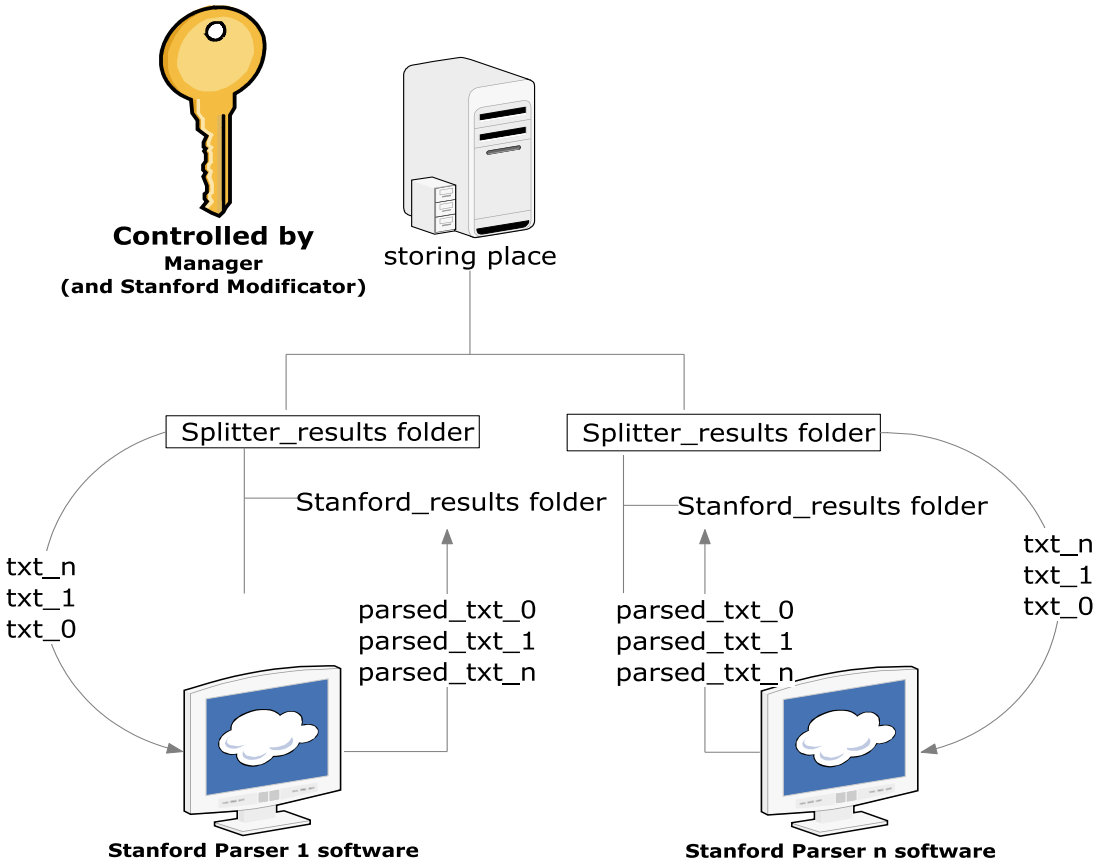
Γιατί φτάσαμε όμως στο σημείο, να διασπούμε ένα αρχείο με το λογισμικό Splitter και να μην το δίνουμε απευθείας στον *parser*; Η απάντηση στο πιο πάνω ερώτημα έχει μεγάλη σχέση με τους πόρους που χρησιμοποιεί ο *parser* και συγκεκριμένα, με την κύρια μνήμη. Χαρακτηριστικά, δίνεται το επόμενο σχήμα, όπου το *length* παρουσιάζει τον αριθμό των λέξεων κάθε πρότασης και δίπλα φαίνεται η χωρητικότητα κύριας μνήμης που απαιτείται από τον *parser* την κάθε φορά:

Length	PCFG
20	50 MB
50	125 MB
100	350 MB

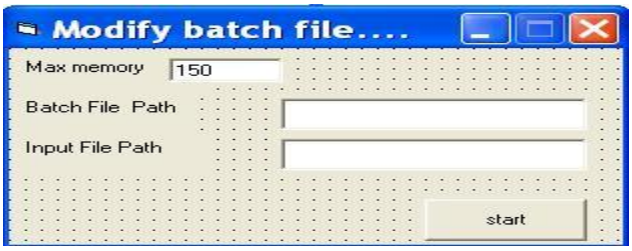
Σχήμα 3.1.6.ε | Αντιστοιχία αριθμού λέξεων των προτάσεων με την ποσότητα μνήμης Ram που χρειάζεται ο Stanford Parser.

Ανάλογα με τον αριθμό των *buckets* που καθορίζονται, έχουμε και ανάλογο αριθμό *parsers*, που τρέχουν σε ένα σύστημα με χωρητικότητα μνήμης 4GB-8GB. Άρα,

λόγω περιορισμένης κύριας μνήμης χρησιμοποιούμε το Splitter, για να μας δημιουργεί αρχεία που περιέχει το καθένα μια πρόταση, για να προχωρά γρήγορα η διαδικασία που αφορά το κάθε *bucket*, ανεξάρτητα προς την εκτέλεση του επόμενου λογισμικού που είναι υπεύθυνο για τη δημιουργία των κατηγορημάτων (Scene Constructor).



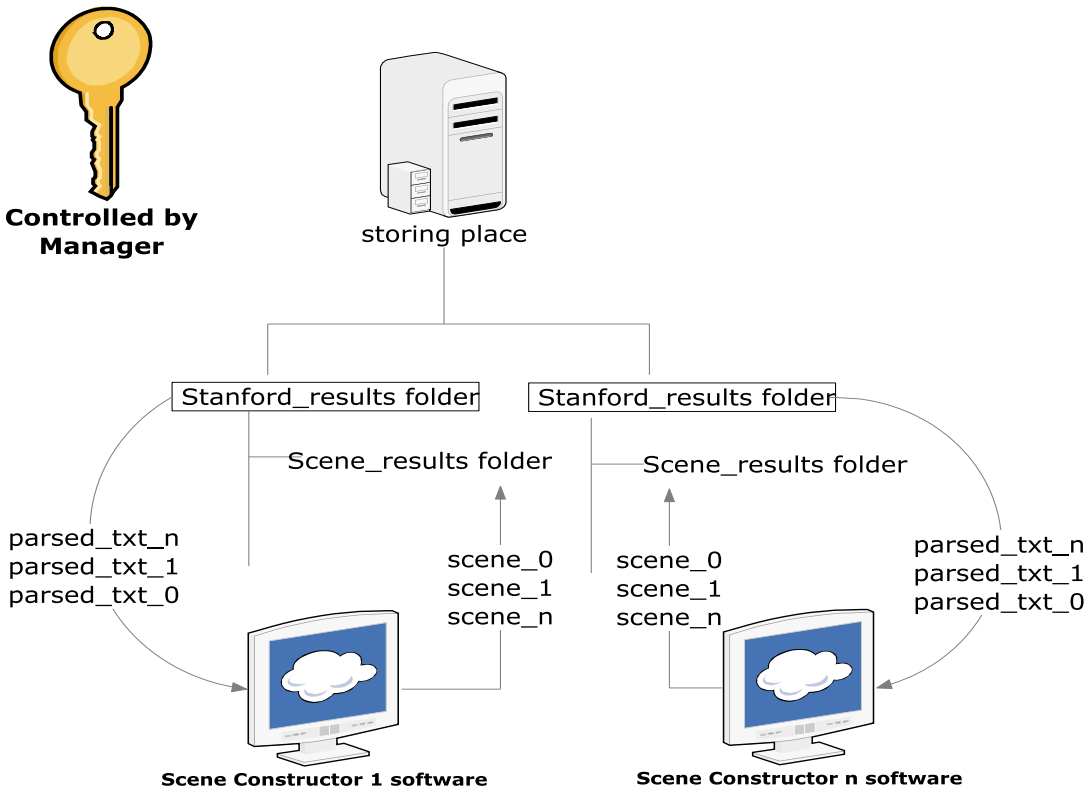
Σχήμα 3.1.6.ζ | *Parsing αρχείων μέσω του Stanford Parser.*



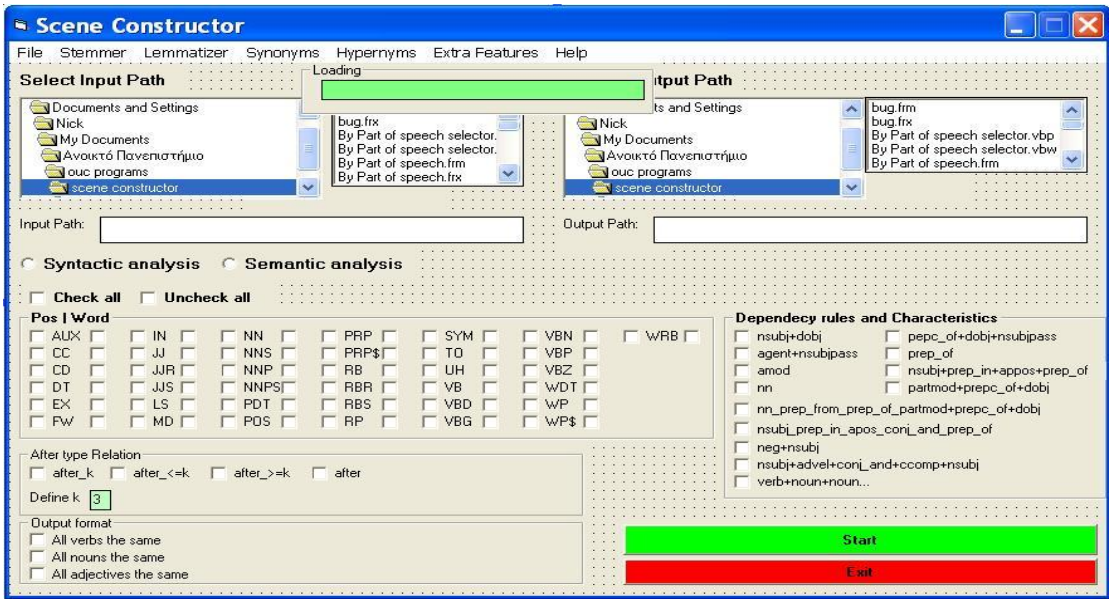
Σχήμα 3.1.6.η | *Κύρια οθόνη Modificator.*

3.1.7 Scene Constructor

Τα λογισμικά μάθησης και συλλογισμού (Learner & Reasoner) δέχονται για είσοδο κατηγορήματα και για αυτό δημιουργήσαμε λογισμικό με την ονομασία Scene Constructor, το οποίο παίρνει τα αρχεία εξόδου του Stanford Parser και δημιουργεί δυαδικά και μοναδιαία κατηγορήματα.



Σχήμα 3.1.7.α | Δημιουργία κατηγορημάτων χρησιμοποιώντας το Scene Constructor.



Σχήμα 3.1.7.β | Κύρια οθόνη Scene Constructor.

Πιο κάτω ακολουθεί η επεξήγηση του τρόπου δημιουργίας κατηγορημάτων με το λογισμικό Scene Constructor.

3.1.7.1 Part of speech relations

Η είσοδος προς το Scene Constructor είναι η έξοδος του Stanford Parser και η οποία, συγκεκριμένα, αφορά συντακτικές πληροφορίες λέξεων (αν η λέξη είναι *NN, NNP, VBZ* κτλ.), καθώς και δυαδικές συσχετίσεις μεταξύ τους (*nsubj, dobj* κτλ.). Θεωρήσαμε σημαντικό να επιλέγουμε το είδος των λέξεων που θα συμμετέχουν στην υλοποίηση των κανόνων για τη δημιουργία κατηγορημάτων. Δηλαδή, ποια είδη λέξεων (*nouns, noun phrases, adjectives, verbs* κτλ.) θέλουμε να ληφθούν υπόψη για τη δημιουργία των κατηγορημάτων. Όπως φαίνεται και στο προηγούμενο σχήμα, υπάρχουν τα εξής μέρη του λόγου, όπου μπορούμε να επιλέγουμε κάθε φορά ποια θέλουμε από αυτά.

AUX - Auxilliary	PRP$ - Possessive pronoun (prolog version PRP-S)
CC - Coordinating conjunction	RB - Adverb
CD - Cardinal number	RBR - Adverb, comparative
DT - Determiner	RBS - Adverb, superlative
EX - Existential there	RP - Particle
FW - Foreign word	SYM - Symbol
IN - Preposition or subordinating conjunction	TO - to
	UH - Interjection
JJ - Adjective	VB - Verb, base form
JJR - Adjective, comparative	VBD - Verb, past tense
JJS - Adjective, superlative	VBG - Verb, gerund or present participle
LS - List item marker	VBN - Verb, past participle
MD - Modal	VBP - Verb, non-3rd person singular present
NN - Noun, singular or mass	VBZ - Verb, 3rd person singular present
NNS - Noun, plural	WDT - Wh-determiner
NNP - Proper noun, singular	WP - Wh-pronoun
NNPS - Proper noun, plural	WP$ - Possessive wh-pronoun (prolog version WP-S)
PDT - Predeterminer	WRB - Wh-adverb
POS - Possessive ending	
PRP - Personal pronoun	

Σχήμα 3.1.7.1.α | Τα μέρη του λόγου των λέξεων μέσα από το περιβάλλον του Scene Constructor.

Η επιλογή, κάθε φορά, διαφορετικών μερών του λόγου (π.χ. τη μια φορά τα ουσιαστικά και τα ρήματα και την άλλη τα ουσιαστικά, τα ρήματα και τα επίθετα), οδηγεί στη δημιουργία διαφορετικών κατηγορημάτων.

Για παράδειγμα, έχουμε το ακόλουθο αρχείο εξόδου του Stanford Parser:

Fireade*/NNP 2000/CD* ***eliminates****/VBZ the/DT* ***complexity****/NN of/IN* ***choosing****/VBG the/DT* ***right****/JJ* ***firefighting****/NN* ***foam****/NN* ***agen****t/NN ./.*

*nsubj(**eliminates**-3, **Fireade**-1)*
num(Fireade-1, 2000-2)
det(complexity-5, the-4)
*dobj(**eliminates**-3, **complexity**-5)*
*prepc_of(**complexity**-5, **choosing**-7)*
det(agent-12, the-8)
*amod(**agen**t-12, **right**-9)*
*nn(**agent**-12, **firefighting**-10)*
*nn(**agent**-12, **foam**-11)*
*dobj(**choosing**-7, **agent**-12)*

Σχήμα 3.1.7.1.β | Έξοδος του Stanford Parser, όπου με υπογράμμιση φαίνονται οι συσχετίσεις που αφορούν μόνο ουσιαστικά, ρήματα και επίθετα.

Αν επιλέξουμε μόνο τα ουσιαστικά, τα ρήματα και τα επίθετα (*IN, JJ, JJR, JJS, NN, NNS, NNP, NNPS, VB, VBD, VBG, VBN, VBP, VBZ),* ο Scene Constructor θα λάβει υπόψη για το σχηματισμό των κανόνων μόνο τις λέξεις *Fireade, eliminates, complexity, choosing, right, firefighting, foam, agent* και θα αγνοήσει τις υπόλοιπες, αγνοώντας και τις συσχετίσεις του Stanford Parser που δεν αναφέρονται μόνο σε αυτές τις λέξεις. Δηλαδή, στις πιο πάνω δυαδικές συσχετίσεις θα λάβει υπόψη μόνο αυτές, όπου και οι δύο λέξεις είναι ουσιαστικά ή ρήματα ή επίθετα, ή τις συσχετίσεις που είναι συνδυασμός ουσιαστικού-ρήματος ή ουσιαστικού-επιθέτου ή ρήματος-επιθέτου.

3.1.7.2 Lemmatizer

Θεωρούσαμε πως η εφαρμογή κανόνων *lemmatization* θα βοηθούσε τα επόμενα λογισμικά της αλυσίδας, που θα ήταν υπεύθυνα για τη μάθηση και το συλλογισμό (Learner & Reasoner). Μέσω κανόνων *lemmatization*, μπορούμε να γνωρίζουμε τη ρίζα της κάθε λέξης και έτσι, εφαρμόζοντάς τους, να καταλαβαίνουμε πότε δύο λέξεις έχουν την ίδια ρίζα, παρόλο που μπορεί να διαφέρουν χρονικά. Για παράδειγμα, αν έχουμε τη λέξη *go* και τη λέξη *went*, μετά από την εφαρμογή *lemmatization*, θα πάρουμε και στις δύο περιπτώσεις, τη λέξη *go.*

Για αυτό εφαρμόσαμε επιλογή για ενεργοποίηση κανόνων *lemmatization*, τους οποίους εντοπίσαμε στα εγχειρίδια και το λογισμικό του WordNet [13] και τροποποιήσαμε κατάλληλα, για να εφαρμόζονται στο περιβάλλον της γλώσσας προγραμματισμού που χρησιμοποιούμε.

Ενεργοποιώντας το *lemmatizer στο λογισμικό Scene Constructor*, οι λέξεις φιλτράρονται και για κάθε λέξη παράγεται το κοινό *lemma* της, το οποίο πρέπει να είναι έγκυρη λέξη. Ο κανόνας για την εύρεση του *lemma* μιας λέξης έχει ως εξής:

- Ψάχνουμε να δούμε αν η λέξη αυτή ανήκει στο *exception list*, το οποίο υπάρχει στο φάκελο του WordNet [13]. Το *exception list* είναι μια λίστα, που περιέχει όλες τις αγγλικές λέξεις που δε διέπονται από κανόνες τροποποίησης μέσω *lemmatization.* Είναι λέξεις πάνω στις οποίες δεν μπορούμε να εφαρμόσουμε κανόνες *lemmatization*, όπως αυτούς τους κανόνες που εφαρμόζουμε στη συνέχεια.
- Αν δε βρεθεί αυτή η λέξη στο *exception list*, τότε προσπαθούμε να εφαρμόσουμε κάποιους κανόνες, για να εντοπίσουμε το *lemma* της λέξης. Κατά την εφαρμογή κάποιου κανόνα, μπορεί να χρειάζεται να ελέγξουμε, αν το αποτέλεσμα μας είναι έγκυρη λέξη. Για αυτό το λόγο, κάποιες φορές, όταν προτείνονται δύο λύσεις σε κάποιο κανόνα, ψάχνουμε ένα αρχείο έγκυρων λέξεων (σαν λεξικό), για να εντοπίσουμε ποια από τις δύο μορφές θα επιστραφεί σαν αποτέλεσμα. Η διαδικασία χρειάζεται και το μέρος του λόγου κάθε λέξης, γιατί άλλοι κανόνες εφαρμόζονται π.χ. για ρήματα και άλλοι για ουσιαστικά και επίθετα.

Αλγόριθμος για lemmatization

Βήμα 0 Ψάξε στο exception list για να δεις αν υπάρχει η λέξη
Βήμα 1 Αν ναι επέστρεψε την ίδια λέξη και τερμάτισε
Βήμα 2 Βρες το part of speech της λέξης
Βήμα 3
Αν POS(λέξης)=noun τότε πήγαινε στο βήμα 4
ΑλλιώςΑν POS=verb πήγαινε στο βήμα 5
ΑλλιώςΑν POS=adjective πήγαινε στο βήμα 6
Αλλιώς επέστρεψε την ίδια λέξη.
ΤΕΛΟΣ ΑΝ
Βήμα 4
Αν τελειώνει (λέξη) σε "ses" τότε θέσε το "ses" σε " s"
ΑλλιώςΑν τελειώνει(λέξη) σε "ches" τότε θέσε το "ches" σε "ch "
ΑλλιώςΑν τελειώνει(λέξη) σε "shes" τότε θέσε το "shes" σε " sh"
ΑλλιώςΑν τελειώνει(λέξη) σε "men" τότε θέσε το "men" σε "man"
ΑλλιώςΑν τελειώνει(λέξη) σε "ies" τότε θέσε το "ies" σε "y"
ΑλλιώςΑν τελειώνει(λέξη) σε "xes" τότε θέσε το "xes" σε "x "
ΑλλιώςΑν τελειώνει(λέξη) σε "zes" τότε θέσε το "zes" σε " z"
ΑλλιώςΑν τελειώνει(λέξη) σε "s" τότε θέσε το "s" σε " "
ΤΕΛΟΣ ΑΝ
Επέστρεψε(Λέξη)
Βήμα 5
Αν τελειώνει(λέξη) σε "ies" τότε θέσε το "ies" σε " y"
ΑλλιώςΑν τελειώνει (λέξη) σε "es" τότε θέσε το "es" σε "e "
Αν WORD (λέξη)=false τότε θέσε το "es" σε " "
ΤΕΛΟΣ ΑΝ
ΑλλιωςΑν τελειώνει(λέξη) σε "ed" τότε θέσε το "ed" σε "e "
Αν WORD(λέξη)=false τότε θέσε το "ed" σε " "
ΤΕΛΟΣ ΑΝ

ΑλλιώςΑν τελειώνει(λέξη) σε "ing" τότε θέσε το "ing" σε "e"
Αν WORD(λέξη)=false τότε θέσε το "ing" σε " "
ΤΕΛΟΣ ΑΝ

ΑλλιώςΑν τελειώνει(λέξη) σε "s" τότε θέσε το "s" σε " "

ΤΕΛΟΣ ΑΝ
Επέστρεψε (Λέξη)
Βήμα 6
Αν τελειώνει(λέξη) σε "est" τότε θέσε το "est" σε "e "
Αν WORD(λέξη)=false τότε θέσε το "est" σε " "
ΤΕΛΟΣ ΑΝ
ΑλλιωςΑν τελειώνει(λέξη) σε "er" τότε θέσε το "er" σε "e "
Αν WORD(λέξη)=false τότε θέσε το "er" σε " "
ΤΕΛΟΣ ΑΝ
ΤΕΛΟΣ ΑΝ
Επέστρεψε(Λέξη)

3.1.7.3 Stemming[15]

Στο λογισμικό του Scene Constructor υπάρχει και ως επιλογή η τεχνική *stemming*, χρησιμοποιώντας τον αλγόριθμο Porter[16], για να παράγουμε τη ρίζα της κάθε λέξης. Σημείο διαφοράς με το *lemmatization* είναι ότι, κατά την παράγωγή του *stemming*, το αποτέλεσμα μπορεί να μην είναι έγκυρη λέξη (π.χ. η λέξη *proposals* με εφαρμογή κανόνων *stemming* γίνεται *propos*, ενώ με την εφαρμογή *lemmatization* γίνεται *proposal*). Αν και μπορεί το αποτέλεσμα, τις περισσότερες φορές, να μην είναι έγκυρη λέξη, αυτό δε σημαίνει ότι η εφαρμογή του *stemming* δεν είναι χρήσιμη. Τα λογισμικά, τα οποία είναι υπεύθυνα για τη μάθηση και το συλλογισμό (Learner & Reasoner), μπορούν να εξάγουν συλλογισμούς, ασχέτως αν τα κατηγορήματα μας θυμίζουν ή όχι έγκυρες λέξεις.

3.1.7.4 Σημασιολογική Ανάλυση

Ο Scene Constructor μπορεί να παράγει σημασιολογικά κατηγορήματα. Δηλαδή, λαμβάνοντας υπόψη τις επιλογές μας (βλέπε 3.1.7.1 Part of speech relations, σελίδα 37), παράγει σχέσεις ανάμεσα σε διάφορες οντότητες.
Οι οντότητες (π.χ 1, 2, 3 κτλ.) χαρακτηρίζονται από διάφορα χαρακτηριστικά (λέξεις) π.χ. άνθρωπος(1), ζώο(2), σκύλος(2) και, επίσης, συμμετέχουν σε δυαδικές σχέσεις (δυαδικά κατηγορήματα), όπως π.χ. ανήκει(2, 1), καλύτερος_φίλος(2, 1).

Κατά την ενεργοποίηση του Scene Constructor και αφού επιλέξουμε τη σημασιολογική ανάλυση, ενεργοποιούνται διάφοροι κανόνες δημιουργίας κατηγορημάτων, όπου μπορούμε να επιλέξουμε ποιους από αυτούς θέλουμε να εφαρμόσουμε. Όλοι οι κανόνες εφαρμόζονται στα αρχεία που παράχθησαν με το Stanford Parser και, σαν αποτέλεσμα, εξάγεται αντίστοιχος αριθμός αρχείων που περιέχουν κατηγορήματα.

Επεξήγηση σημασιολογικών κανόνων συσχέτισης(δημιουργίας κατηγορημάτων)

Συνολικά, μπορούν να εφαρμοστούν μέχρι και δεκατρείς σημασιολογικοί κανόνες στα αρχεία εξόδου του Stanford Parser. Μέσω αυτών των δεκατριών κανόνων, δημιουργούμε μοναδιαία και δυαδικά κατηγορήματα, τα οποία αναφέρονται, γενικότερα, σε συσχετίσεις ανάμεσα σε λέξεις των προτάσεων. Πιο κάτω, εξηγούμε αναλυτικά αυτούς τους κανόνες, παραθέτοντας κάθε φορά και παραδείγματα εισόδου και εξόδου. Η επιλογή των κανόνων προέκυψε μέσα από μελέτη του εγχειριδίου του Stanford Parser (βλέπε Παράρτημα, Επεξήγηση δυαδικών συσχετίσεων Stanford Parser, σελίδα 97) και μέσα από πειράματα με τον *parser*, όπου προσπαθούσαμε να καταλάβουμε τον τρόπο δημιουργίας των αρχικών δυαδικών συσχετίσεων. Δηλαδή, κυρίως μέσα από μελέτη του τρόπου εξαγωγής των συσχετίσεων από τον *parser*, εφαρμόσαμε κανόνες περαιτέρω συσχετισμού σε ένα δεύτερο επίπεδο, λαμβάνοντας υπόψη και το μέρος του λόγου της κάθε λέξης, με αποτέλεσμα τη δημιουργία των κατηγορημάτων.

Πριν προχωρήσουμε στην αποτύπωση του τρόπου δημιουργίας των δυαδικών και μοναδιαίων κατηγορημάτων και για πιο εύκολη αναφορά, θα παρουσιάσουμε, σε έναν πίνακα, τις συσχετίσεις που χρησιμοποιούμε από την έξοδο του Stanford Parser, παραθέτοντας δίπλα (σε πιστή μετάφραση από την Αγγλική γλώσσα), τι σημαίνει η κάθε μία.

[15] http://en.wikipedia.org/wiki/Stemming [Μάιος 2011].
[16] http://tartarus.org/~martin/PorterStemmer/ [Μάιος 2011].

nsubj	Ένα ονομαστικό υποκείμενο είναι μια φράση που ενέχει θέση ουσιαστικού, η οποία είναι το συντακτικό υποκείμενο μιας πρότασης.
dobj	Το άμεσο αντικείμενο μιας ρηματικής φράσης είναι μια φράση που ενέχει θέση ουσιαστικού και είναι (στην αιτιατική) το αντικείμενο του ρήματος.
agent: agent	Το ποιητικό αίτιο είναι ένα δεύτερο αντικείμενο ενός ρήματος της παθητικής φωνής, το οποίο εισάγεται με την πρόθεση «από» και υποδηλώνει τον εκτελεστή της πράξης.
nsubjpass: passive nominal subject	Ένα ονομαστικό υποκείμενο, στην παθητική φωνή, είναι μια φράση που ενέχει θέση ουσιαστικού και η οποία είναι το συντακτικό υποκείμενο μιας πρότασης στην παθητική φωνή.
prep/prepc	Ο εμπρόθετος προσδιορισμός ενός ρήματος, επιθέτου ή ουσιαστικού είναι οποιαδήποτε εμπρόθετη φράση, η οποία εξυπηρετεί στο να διαφοροποιήσει το νόημα του ρήματος, επιθέτου ή ουσιαστικού. Κάποτε, μπορεί να εμφανίζεται και με επιπλέον πληροφορία, όπως prepc_of, prepc_in κτλ.
appos	Ένας παραθετικός προσδιορισμός μιας φράσης, που ενέχει θέση ουσιαστικού, είναι μια φράση που ενέχει θέση ουσιαστικού και μπαίνει, αμέσως μετά τη φράση που ενέχει θέση ουσιαστικού. Εξυπηρετεί στο να προσδιορίσει ή να διαφοροποιήσει αυτή τη φράση που ενέχει θέση ουσιαστικού.
partmod	Ένας μετοχικός προσδιορισμός μιας φράσης που ενέχει θέση ουσιαστικού ή μιας ρηματικής φράσης, είναι μια μετοχική μορφή ρήματος, η οποία εξυπηρετεί στο να διαφοροποιήσει το νόημα της φράσης που ενέχει θέση ουσιαστικού ή της ρηματικής φράσης.
nn	Ο προσδιορισμός ενός σύνθετου ουσιαστικού μιας φράσης, που ενέχει θέση ουσιαστικού, είναι οποιοδήποτε ουσιαστικό, το οποίο εξυπηρετεί στο να διαφοροποιήσει το κύριο ουσιαστικό. (Προσέξτε ότι, στο τωρινό σύστημα απομάκρυνσης από εξαρτήσεις, όλα τα ουσιαστικά διαφοροποιούν το κύριο ουσιαστικό μιας φράσης που ενέχει θέση ουσιαστικού).
ccomp	Μια συμπληρωματική πρόταση μιας ρηματικής φράσης ή μιας φράσης, η οποία ενέχει θέση επιθέτου, είναι μια πρόταση με εσωτερικό υποκείμενο, το οποίο λειτουργεί, όπως το αντικείμενο του ρήματος ή του επιθέτου. Μια συμπληρωματική πρόταση μιας επιμέρους πρότασης είναι η συμπληρωματική πρόταση μιας ρηματικής φράσης ή μιας φράσης, η οποία ενέχει θέση επιθέτου και η οποία είναι το κατηγορούμενο εκείνης της πρότασης .
advcl	Ένας προσδιορισμός μιας επιρρηματικής πρότασης, μιας ρηματικής φράσης, είναι μια πρόταση που διαφοροποιεί το ρήμα (χρονική πρόταση, αποτέλεσμα, υποθετική πρόταση κτλ.), όπως το «και» ή το «ή» κτλ.
neg	Ένας αντιθετικός προσδιορισμός είναι η σχέση ανάμεσα σε μια αντιθετική λέξη με τη λέξη, με την οποία τη διαφοροποιεί.
conj_and	Ένας σύνδεσμος είναι η σχέση ανάμεσα σε δυο στοιχεία, τα οποία συνδέονται από ένα συντονιστικό σύνδεσμο.
amod: adjectival modifer	Εξυπηρετεί στο να διαφοροποιεί την έννοια ενός ουσιαστικού.

Σχήμα 3.1.7.4 | Συσχετίσεις Stanford Parser που χρησιμοποιούμε για τη δημιουργία κατηγορημάτων.

1ος Σημασιολογικός κανόνας

Αρχείο που παράχθηκε με το Stanford Parser.		Μετά την εφαρμογή του Scene Constructor.
nsubj(x-3, y-1) dobj(x-3, z-5)	=>	o___OP_WRD_OP__y_o([token:1]) is true. o___OP_WRD_OP__z_o([token:5]) is true. o___OP_REL_OP__x_o([token:1, token:5]) is true.

Υπάρχει άμεση σχέση μεταξύ των οντοτήτων που συμμετέχουν σε συσχετίσεις *nsubj* και *dobj*, γιατί, μέσω αυτών, μπορούμε να κάνουμε συσχετισμούς ανάμεσα στα αντικείμενα και τα υποκείμενα, μέσω ρηματικών φράσεων. Το μόνο που χρειαζόμαστε είναι να εντοπίζουμε σε κάθε πρόταση, που παράγεται με τον *parser*, τις συσχετίσεις *nsubj* και *dobj* που αναφέρονται στις ίδιες οντότητες.

Παράδειγμα		
nsubj(play-3, Nick-1) dobj(play-3, ball-5)	=>	o___OP_WRD_OP__Nick_o([token:1]) is true. o___OP_WRD_OP__ball_o([token:5]) is true. o___OP_REL_OP__play_o([token:1, token:5]) is true.

2ος Σημασιολογικός κανόνας

Αρχείο που παράχθηκε με το Stanford Parser.		Μετά την εφαρμογή του Scene Constructor.
agent(x-29, y-34) nsubjpass(x-29, z-27)	=>	o___OP_WRD_OP__y_o([token:34]) is true. o___OP_WRD_OP__z_o([token:27]) is true. o___OP_REL_OP__x_o([token:34, token:27]) is true.

Με αυτόν τον κανόνα συσχετίζουμε, μέσω του ρήματος, το υποκείμενο που βρίσκεται σε παθητική φωνή με τον εκτελεστή της πράξης, ο οποίος, με τη σειρά του, υποδηλώνει το δεύτερο αντικείμενο του ρήματος.

Παράδειγμα		
agent(rejected-29, court-34) nsubjpass(rejected-29, appeal-27)	=>	o___OP_WRD_OP__court_o([token:34]) is true. o___OP_WRD_OP__appeal_o([token:27]) is true. o___OP_REL_OP__rejected_o([token:34, token: 27]) is true.

3ος Σημασιολογικός κανόνας

Αρχείο που παράχθηκε με το Stanford Parser.		Μετά την εφαρμογή του Scene Constructor.
nsubjpass(x-11, y-9) prepc_of(x-11, z-13) dobj(z-13, w-15)	=>	o___OP_WRD_OP__y_o([token:9]) is true. o___OP_WRD_OP__x_o([token:9]) is true. o___OP_WRD_OP__w_o([token:15]) is true. o___OP_REL_OP__z_o([token:9, token:15]) is true.

Ο εμπρόθετος προσδιορισμός (prepc) είναι μια φράση, η οποία διαφοροποιεί το νόημα του ρήματος και μέσω αυτού, μπορούμε να συνδέσουμε το υποκείμενο της πρότασης, που βρίσκεται σε παθητική φωνή, με ένα άλλο ουσιαστικό της δευτερεύουσας αιτιολογικής πρότασης.

Παράδειγμα		
nsubjpass(convicted-11, David-9) prepc_of(convicted-11, killing-13) dobj(killing-13, MacPhilip-15)	=>	o___OP_WRD_OP__David_o([token:9]) is true. o___OP_WRD_OP__convicted_o([token:9]) is true. o___OP_WRD_OP__Macphilip_o([token:15]) is true. o___OP_REL_OP__killing_o([token:9, token:15]) is true.

4ος Σημασιολογικός κανόνας

Αρχείο που παράχθηκε με το Stanford Parser.		Μετά την εφαρμογή του Scene Constructor.
nsubj(x-3, y-2) prep_in(x-3, z-6) appos(z-6, t-9) prep_of(t-9, v-18)	=>	o___OP_WRD_OP__y_o([token:2]) is true. o___OP_WRD_OP__v_o([token:9]) is true. o___OP_WRD_OP__t_o([token:9]) is true. o___OP_REL_OP__z_o([token:2, token:9]) is true.

Μέσω της ονοματικής φράσης (nsubj) και μέσω του εμπρόθετου προσδιορισμού (prep_in), πληροφορούμαστε για την ενέργεια του υποκειμένου και ακολούθως, μέσω του παραθετικού προσδιορισμού (appos), ο οποίος προσδιορίζει, με τη σειρά του, την ενέργεια του υποκειμένου, οδηγούμαστε στη δημιουργία κατηγορημάτων. Στο δυαδικό κατηγόρημα συνδέεται το υποκείμενο της κύριας πρότασης με ουσιαστικό δευτερεύουσας πρότασης.

Παράδειγμα		
nsubj(participated-3, Ng-2) prep_in(participated-3, kidnapping-6) appos(kidnapping-6, torture-9) prep_of(torture-9, victims-18)	=>	o___OP_WRD_OP__Ng_o([token:2]]) is true. o___OP_WRD_OP__Victims_o([token:9]]) is true. o___OP_WRD_OP__Torture_o([token:9]]) is true. o___OP_REL_OP__kidnapping_o([token:2, token:9]]) is true.

5ος Σημασιολογικός κανόνας

Αρχείο που παράχθηκε με το Stanford Parser.		Μετά την εφαρμογή του Scene Constructor.
partmod(x-26, y-27) prepc_of(y-27, z-29) dobj(z-29, w-31)	=>	o___OP_WRD_OP__x_o([token:26]) is true. o___OP_WRD_OP__w_o([token:31]) is true. o___OP_WRD_OP__y_o([token:26]) is true. o___OP_REL_OP__z_o([token:26, token:31]) is true.

Μέσω του μετοχικού προσδιορισμού (partmod), που, στην ουσία, είναι μετοχική μορφή ρήματος, διαφοροποιούμε το νόημα της ονοματικής φράσης. Ακολούθως, μέσω του εμπρόθετου προσδιορισμού (prepc), ο οποίος έμμεσα, διαφοροποιεί και αυτός το νόημα της ονοματικής φράσης, μπορούμε να συνδυάσουμε τα πιο πάνω με το αντικείμενο του ρήματος (dobj) και να δημιουργήσουμε κατηγορήματα.

Παράδειγμα		
partmod(man-26, accused-27) prepc_of(accused-27, killing-29) dobj(killing-29, women-31)	=>	o___OP_WRD_OP__man_o([token:26]) is true. o___OP_WRD_OP__women_o([token:31]) is true. o___OP_WRD_OP__accused_o([token:26]) is true. o___OP_REL_OP__killing_o([token:26, token:31]) is true.

6ος Σημασιολογικός κανόνας

Αρχείο που παράχθηκε με το Stanford Parser.		Μετά την εφαρμογή του Scene Constructor.
nn(x-7, y-6) prep_from(r-20, x-7) prep_from(r-20, c-23) prep_of(c-23, m-26) partmod(m-26, a-27) prepc_of(a-27, k-29) dobj(k-29, w-31)	=>	o___OP_WRD_OP__y_o([token:7]) is true. o___OP_WRD_OP__w_o([token:31]) is true. o___OP_WRD_OP__a_o([token:7]) is true. o___OP_REL_OP__k_o([token:7, token:31]) is true.

Μέσω της συσχέτισης *nn*, προσδιορίζουμε το υποκείμενο (στο πιο κάτω παράδειγμα το Sowel) και ακολούθως, μέσω των εμπρόθετων προσδιορισμών (prepc) και μέσω της αλληλουχίας των μετοχικών προσδιορισμών (partmod), μπορούμε να εντοπίσουμε το αντικείμενο του υποκειμένου.

Παράδειγμα		
nn(Case_Tuesday-7, Sowell-6) prep_from(removed-20, Case_Tuesday-7) prep_from(removed-20, case-23) prep_of(case-23, man-26) partmod(man-26, accused-27) prepc_of(accused-27, killing-29) dobj(killing-29, women-31)	=>	o___OP_WRD_OP__sowell_o([token:7]) is true. o___OP_WRD_OP__women_o([token:31]) is true. o___OP_WRD_OP__accused_o([token:7]) is true. o___OP_REL_OP__killing_o([token:7, token:31]) is true.

7ος Σημασιολογικός κανόνας

Αρχείο που παράχθηκε με το Stanford Parser.		Μετά την εφαρμογή του Scene Constructor.
nsubj(x-5, y-2) prep_in(x-5, k-7) apos(k-7, t-9) conj_and(t_9, m-11) prep_of(t-9, v-18)	=>	o___OP_WRD_OP__v_o([token:9]) is true. o___OP_WRD_OP__t_o([token:9]) is true. o___OP_WRD_OP__y_o([token:2]) is true. o___OP_REL_OP__m_o([token:2, token:9]) is true. o___OP_REL_OP__k_o([token:2, token:9]) is true.

Μέσω της ονοματικής φράσης (nusbj) και μέσω του εμπρόθετου προσδιορισμού (prep), μπορούμε να συσχετίσουμε το υποκείμενο, έμμεσα, με το αντικείμενο του ρήματος. Ακολούθως, μέσω του παραθετικού προσδιορισμού και του συνδέσμου (conj_and), μπορούμε να συσχετίσουμε το υποκείμενο με το ουσιαστικό μιας δευτερεύουσας φράσης, μέσω συγκεκριμένου ρήματος.

Παράδειγμα		
nsubj(paricipated-5, Ng-2) prep_in(participated-5, kidnapping-7) apos(kidnapping-7, torture-9) conj_and(torture_9, murder-11) prep_of(torture-9, victims-18)	=>	o___OP_WRD_OP__victims_o([token:9]) is true. o___OP_WRD_OP__torture_o([token:9]) is true. o___OP_WRD_OP__Ng_o([token:2]) is true. o___OP_REL_OP__murder_o([token:2, token:9]) is true. o___OP_REL_OP__kidnappinng_o([token:2, token:9]) is true.

8ος Σημασιολογικός κανόνας

Αρχείο που παράχθηκε με το Stanford Parser.		Μετά την εφαρμογή του Scene Constructor.
nsubj(x-2, y-1) advcl(x-2, r-12) conj_and(r-12, b-14) ccomp(b-14, z-18) nsubj(z-18, g-17)	=>	o___OP_WRD_OP__y_o([token:1]) is true. o___OP_WRD_OP__r_o([token:12]) is true. o___OP_REL_OP__x_o([tokan:1, token:12]) is true. o___OP_WRD_OP__b_o([token:14]) is true. o___OP_REL_OP__x_o([token:1, token:14]) is true. o___OP_WRD_OP__g_o([token:17]) is true. o___OP_REL_OP__r_o([token:1, token:17]) is true. o___OP_REL_OP__b_o([token1, token:17]) is true.

Μέσω της ονοματικής φράσης (nsubj) και μέσω του προσδιορισμού επιρρηματικής πρότασης (advcl), μπορούμε, έμμεσα, να συνδέσουμε το υποκείμενο με το άμεσο αντικείμενο της κύριας πρότασης και να δημιουργήσουμε μοναδιαία και δυαδικά κατηγορήματα.
Ακολούθως, μέσω του συνδέσμου (conj_and), μπορούμε να δημιουργήσουμε περισσότερα κατηγορήματα, γιατί συνδυάζουμε, έτσι, το υποκείμενο με άλλα αντικείμενα.

Στη συνέχεια, με το κατηγορούμενο της κύριας πρότασης (πρόταση με εσωτερικό υποκείμενο, το οποίο λειτουργεί, όπως το αντικείμενο του ρήματος) συνδέουμε το εσωτερικό υποκείμενο με το άμεσο υποκείμενο της κύριας πρότασης.

Παράδειγμα		
nsubj(committed-2, Alcala-1) advcl(committed-2, raped-12) conj_and(raped-12, battered-14) ccomp(battered-14, referred-18) nsubj(referred-18, girl-17)	=>	o___OP_WRD_OP__Alcala_o([token:1]) is true. o___OP_WRD_OP__raped_o([token:12]) is true. o___OP_REL_OP__commited_o([tokan:1, token:12]) is true. o___OP_WRD_OP__battered_o([token:14]) is true. o___OP_REL_OP__commited_o([token:1, token:14]) is true. o___OP_WRD_OP__girl_o([token:17]) is true. o___OP_REL_OP__raped_o([token:1, token:17]) is true. o___OP_REL_OP__battered_o([token1, token:17]) is true.

9ος Σημασιολογικός κανόνας

Αρχείο που παράχθηκε με το Stanford Parser.		Μετά την εφαρμογή του Scene Constructor.
neg(x-18, not-16) nsubj(x-18, y-4)	=>	o___OP_WRD_OP__y_o([token:4]) is true. o___OP_WRD_OP__x_o([token:18]) is true. o___OP_REL_OP__not_o([token:4, token:18]) is true.

Μέσω του αντιθετικού προσδιορισμού, μπορούμε να δημιουργήσουμε μοναδιαία και δυαδικά κατηγορήματα, τα οποία υποδηλώνουν σχέση άρνησης ανάμεσα στην αντιθετική λέξη (not) με το υποκείμενο της πρότασης.

Παράδειγμα		
neg(Powel-18, not-16) nsubj(Powel-18, remains-4)	=>	o___OP_WRD_OP__remains_o([token:4]) is true. o___OP_WRD_OP__Powel_o([token:18]) is true. o___OP_REL_OP__not_o([token:4, token:18]) is true.

10ος Σημασιολογικός κανόνας

Αρχείο που παράχθηκε με το Stanford Parser.		Μετά την εφαρμογή του Scene Constructor.
nsubj(x-2, y-1) advcl(x-2, z-12) Όπου x=ρήμα Όπου y=ουσιαστικό Όπου z=ουσιαστικό	=>	o___OP_WRD_OP__y_o([token:1]) is true. o___OP_WRD_OP__z_o([token:12]) is true. o___OP_REL_OP__x_o([token:1, token:12]) is true.

Πρόκειται για έναν απλό κανόνα με τον οποίο, για κάθε συσχετιζόμενο ουσιαστικό (μέσω κάποιου ρήματος), δημιουργείται ένα δυαδικό κατηγόρημα. Επίσης, για κάθε ουσιαστικό δημιουργείται ένα μοναδιαίο κατηγόρημα.

Παράδειγμα		
nsubj(committed-2, Alcala-1) advcl(committed-2, rape-12)	=>	o___OP_WRD_OP__Alcala_o([token:1]) is true. o___OP_WRD_OP__rape_o([token:12]) is true. o___OP_REL_OP__commited_o([token:1, token:12]) is true.

Εκτός από τους πιο πάνω κανόνες που παράγουν δυαδικά και μοναδιαία κατηγορήματα, δημιουργήσαμε, επιπλέον, και κάποιους που δημιουργούν μόνο μοναδιαία. Στόχος ήταν, σε συνδυασμό με τους σημασιολογικούς κανόνες 1 μέχρι και 10, να εμπλουτιστούν τα αρχεία με περισσότερα κατηγορήματα, με απώτερο σκοπό την εξαγωγή καλύτερων συμπερασμάτων από την Έξυπνη Μηχανή.

11ος Σημασιολογικός κανόνας

Αρχείο που παράχθηκε με το Stanford Parser.		Μετά την εφαρμογή του Scene Constructor.
amod(x-1, y-2)	=>	o___OP_WRD_OP__x_o([token:1]) is true. o___OP_WRD_OP__y_o([token:1]) is true.

Επειδή είναι το επίθετο που τροποποιεί το ουσιαστικό, μέσω αυτής της συσχέτισης, μπορούμε να δημιουργήσουμε δύο μοναδιαία κατηγορήματα που αναφέρονται στην ίδια οντότητα.

Παράδειγμα		
amod(court-1, federal-2)	=>	o___OP_WRD_OP__court_o([token:1]) is true. o___OP_WRD_OP__federal_o([token:1]) is true.

12ος Σημασιολογικός κανόνας

Αρχείο που παράχθηκε με το Stanford Parser.		Μετά την εφαρμογή του Scene_constructor
nn(x-5, y-1)	=>	o___OP_WRD_OP__x_o([token:5]) is true. o___OP_WRD_OP__y_o([token:5]) is true.

Επειδή είναι το ουσιαστικό που τροποποιεί το κύριο ουσιαστικό, μπορούμε να δημιουργήσουμε δύο μοναδιαία κατηγορήματα που αναφέρονται στην ίδια οντότητα.

Παράδειγμα		
nn(Alcala-5, Convicted-1)	=>	o___OP_WRD_OP__Alcala_o([token:5]) is true. o___OP_WRD_OP__Convicted_o([token:5]) is true.

13ος Σημασιολογικός κανόνας

Αρχείο που παράχθηκε με το Stanford Parser.		Μετά την εφαρμογή του Scene Constructor.
prep_of(x-9, y-13)	=>	o___OP_WRD_OP__x_o([token:9]) is true. o___OP_WRD_OP__y_o([token:9]) is true.

Είναι αυτό που τροποποιεί το κύριο ουσιαστικό ή το επίθετο ή το ρήμα. Επομένως, μπορούμε να δημιουργήσουμε δύο μοναδιαία κατηγορήματα που αναφέρονται στην ίδια οντότητα.

Παράδειγμα		
prep_of(criteria-9, clarity-13)	=>	o___OP_WRD_OP__criteria_o([token:9]) is true. o___OP_WRD_OP__clarity_o([token:9]) is true.

3.1.7.4.1 Υπερώνυμα (Hypernym Relations)

Τα υπερώνυμα αποτυπώνουν σχέσεις «is-a» μεταξύ λέξεων. Για παράδειγμα, το *color* είναι υπερώνυμο του *red*, γιατί το *red is-a colour*. Επίσης, μερικά από τα υπερώνυμα της λέξης *man* είναι τα εξής (μέσα από το WordNet):
=> person, individual, someone, somebody, mortal, soul
=> organism, being

Με την ενεργοποίηση των υπερωνύμων, μέσω του λογισμικού Scene Constructor, εμπλουτίζουμε τα αποτελέσματα που εξάγονται κατά τη δημιουργία των κατηγορημάτων. Αν, διαβάζοντας ένα αρχείο που περιέχει μία πρόταση καταλήγουμε στη δημιουργία δύο κατηγορημάτων χωρίς τη χρήση υπερωνύμων, τότε, ενεργοποιώντας τα υπερώνυμα, μπορούμε να καταλήξουμε στη δημιουργία π.χ. δέκα κατηγορημάτων. Αυτό, σαν αποτέλεσμα, πιστεύαμε πως θα διευκόλυνε το λογισμικό Reasoner στη διαδικασία του συλλογισμού, αφού θα εμπλουτιζόταν το πεδίο της γνώσης με περισσότερα κατηγορήματα.

Παράδειγμα (υποθέτουμε ότι έχουμε τα πιο κάτω κατηγορήματα):
o___OP_WRD_OP__man_o([token:1]) is true.
o___OP_WRD_OP__car_o([token:2]) is true.
o___OP_REL_OP__drive_o([token:1, token:2]) is true.

Με την ενεργοποίηση των υπερωνύμων π.χ. για τη λέξη *drive* θα έχουμε, επιπλέον, και τα εξής δυαδικά κατηγορήματα:
o___OP_REL_OP__operate_o([token:1, token:2]) is true.
o___OP_REL_OP__control_o([token:1, token:2]) is true.

Άρα, βλέπουμε ότι μπορούμε να εμπλουτίσουμε ακόμα περισσότερο την πληροφορία που εξάγεται, γιατί βλέπουμε ότι ο άνθρωπος, εκτός από το να οδηγεί αυτοκίνητο, επιπλέον μπορεί και να το ελέγχει.

Για τη διαδικασία χρήσης και αξιοποίησης των υπερωνύμων, χρησιμοποιούμε τα αρχεία του λογισμικού WordNet [13], γράφοντας κώδικα μέσα από το περιβάλλον του λογισμικού Scene Constructor. Συγκεκριμένα, αξιοποιούμε οκτώ αρχεία, όπου τα τέσσερα είναι τα *index files* για τα ουσιαστικά, επίθετα, ρήματα και επιρρήματα και τα άλλα τέσσερα είναι τα αρχεία με τα υπερώνυμα. Σε κάθε περίπτωση, για την ενεργοποίηση των υπερωνύμων, πρέπει πρώτα να γίνει ενεργοποίηση του *lemmatization*, γιατί η συνάρτηση (των υπερωνύμων) λειτουργεί με τη ρίζα της λέξης.

Για τη διαδικασία εντοπισμού των υπερωνύμων μιας λέξης (π.χ. *car*), ακολουθείται ο εξής αλγόριθμος:

Βήμα 1: Βρες το POS της λέξης, όπου p=pos(car)
Βήμα 2: Άνοιξε το ανάλογο index file και βρες τον 8_ψήφιο κωδικό t=index_file(p, car)
Βήμα 3: Άνοιξε το ανάλογο data_file, βρες και επέστρεψε τις λέξεις, όπου n=data_file(p, t)

/*όπου, στην περίπτωσή μας, επιστρέφονται οι λέξεις car, auto, automobile, machine, motorcar*/

Βάθος και έννοιες υπερωνύμων
Μέσω του Scene Constructor, μπορούμε να πάμε σε αρκετά επίπεδα βάθους (depth) και σε αρκετές έννοιες (senses), σε σχέση με την επιλογή των υπερωνύμων.

Συγκεκριμένα, υπάρχει επιλογή μέσα από την οποία καθορίζουμε τα εξής:

- Senses
- Depth
- Output Type

Επιλογή Sense (0, 1, 2, 3, 4, 5, 6, 7, max)
Μέσα από αυτή την επιλογή, διαλέγουμε πόσες έννοιες της ζητούμενης λέξης θέλουμε να επιστραφούν και να ερευνηθούν για τα υπερώνυμα τους.

Επιλογή Depth (0-0, 0-1, 0-2.....1-1, 1-2, 1-3...2-3, 2-7..3-3...->max)
Με την επιλογή αυτή, καθορίζουμε το επίπεδο βάθους των υπερωνύμων, όπου μερικά παραδείγματα είναι τα εξής:
0-0 Σημαίνει ότι θέλουμε μόνο την αρχική επιστροφή των λέξεων.
0-4 Σημαίνει ότι θέλουμε την επιστροφή των λέξεων, βάθους 5 επιπέδων.
5-7 Σημαίνει ότι θέλουμε την επιστροφή των λέξεων, ξεκινώντας από το 5^{ο} επίπεδο βάθους μέχρι και το 7^{ο}.
6-6 Σημαίνει ότι θέλουμε την επιστροφή των λέξεων του 6^{ου} επιπέδου βάθους μόνο.

Πιο κάτω, δίνουμε ένα παράδειγμα επιπέδων βάθους και εννοιών για τη λέξη *car*, που δίνεται από το WordNet.

```
---------------------------------------------------------------------------------------------------
    Sense 1
    car, auto, automobile, machine, motorcar -- (a motor vehicle with four wheels; usually propelled by an internal
    combustion engine; "he needs a car to get to work")
        => motor vehicle, automotive vehicle -- (a self-propelled wheeled vehicle that does not run on rails)
            => self-propelled vehicle -- (a wheeled vehicle that carries in itself a means of propulsion)
                => wheeled vehicle -- (a vehicle that moves on wheels and usually has a container for transporting things
    or people; "the oldest known wheeled vehicles were found in Sumer and Syria and date from around 3500 BC")
                    => vehicle -- (a conveyance that transports people or objects)
                        => conveyance, transport -- (something that serves as a means of transportation)
                            => instrumentality, instrumentation -- (an artifact (or system of artifacts) that is instrumental in
    accomplishing some end)
                                => artifact, artefact -- (a man-made object taken as a whole)
                                    => whole, unit -- (an assemblage of parts that is regarded as a single entity; "how big is that
    part compared to the whole?"; "the team is a unit")
                                        => object, physical object -- (a tangible and visible entity; an entity that can cast a shadow;
    "it was full of rackets, balls and other objects")
                                            => physical entity -- (an entity that has physical existence)
                                                => entity -- (that which is perceived or known or inferred to have its own distinct
    existence (living or nonliving))

---------------------------------------------------------------------------------------------------
```

Σχήμα 3.1.7.4.1 | Έξοδος του WordNet με τα υπερώνυμα της λέξης car.

Αν ζητήσουμε, μέσω του Scene Constructor το επίπεδο βάθους 0-0, θα επιστραφούν οι λέξεις: (car, auto, automobile, machine, motorcar).
Αν ζητήσουμε το επίπεδο βάθος 4-5, θα επιστραφούν οι λέξεις: (vehicle, conveyance, transport).
[Σημείωση: Τα πιο πάνω παραδείγματα αφορούν μόνο το sense 1].

Επιλογή Output Type
Μαζί με τις επιλογές των υπερωνύμων έχουμε και τις εξής επιλογές:

- *Select all*
  Επιστρέφονται όλες οι λέξεις των επιθυμητών επιπέδων βάθους (όχι διπλότυπα).
- *Select first*
  Επιστρέφεται η πρώτη λέξη του κάθε επιπέδου βάθους. Αν επιλέξουμε *sense* 2 και *depth* 3-4, θα επιστραφούν τέσσερις λέξεις, συγκεκριμένα, η πρώτη λέξη του επιπέδου βάθους 3 και η πρώτη λέξη του επιπέδου βάθους 4 και των 2 *senses (του sense 1 και του sense 2).*
- *Select rare*
  Σε αυτήν την περίπτωση, επιστρέφεται η πιο σπάνια λέξη του επιλεγμένου συνόλου. Για παράδειγμα, αν έχουμε επιλεγμένο επίπεδο βάθους 1-5 και *sense* 3, τότε, ανάμεσα στα 3 *senses* και ανάμεσα στα επίπεδα βάθους 1, 2, 3, 4 και 5 του κάθε *sense*, θα βρει και θα επιστρέψει την πιο σπάνια λέξη (επιστρέφεται μόνο μία λέξη). Όταν επιλέξουμε *select rare*, έχουμε και τη δυνατότητα (Rare options) να επιλέξουμε, αν θέλουμε τα εξής:
    - Individual depth
    - Group of depth

  (Για την καλύτερη εξήγηση των πιο κάτω, θεωρούμε ότι επιλέγουμε επίπεδο βάθους 1-4 και *sense* 2).
  -Με την επιλογή *Individual depth* θα επιστραφεί η πιο σπάνια λέξη του $1^{ου}$, του $2^{ου}$, του $3^{ου}$ και του $4^{ου}$ επιπέδου βάθους ανεξάρτητα για όλα τα *senses* (του sense 1 και του *sense* 2). Δηλαδή, θα επιστραφούν οκτώ λέξεις σε αυτήν την περίπτωση.
  -Με την επιλογή *Group of depth* θα επιστραφεί η πιο σπάνια λέξη ανάμεσα στο επίπεδο βάθους 1 μέχρι 4 του κάθε *sense* (sense 1 και sense 2). Δηλαδή, θα επιστραφούν δύο λέξεις σε αυτήν την περίπτωση.

3.1.7.5 Συντακτική Ανάλυση
Κατά τη συντακτική ανάλυση, όπως και στη σημασιολογική, λαμβάνονται πρώτα οι επιλογές μας και συγκεκριμένα, τι είδους λέξεις (επίθετα, ουσιαστικά, ρήματα κτλ.) θέλουμε να συμμετέχουν στη δημιουργία των κατηγορημάτων. Ακολούθως, ο Scene Constructor μπορεί να εξάγει διάφορα κατηγορήματα, διαβάζοντας τα αρχεία που δημιουργήθηκαν με το Stanford Parser και συγκεκριμένα, κατηγορήματα τα οποία περιλαμβάνουν μόνο τις προτάσεις με τα μέρη του λόγου που επιλέξαμε. Δε λαμβάνονται υπόψη οι δυαδικές συσχετίσεις που εξάγει ο *parser*, παρά μόνο το μέρος του λόγου κάθε λέξης.

Υποθέτουμε ότι δίνουμε την ακόλουθη πρόταση στον *parser*:
Convicted serial killer Rodney Alcala may have had more victims.

Και παίρνουμε μετά το parsing την ακόλουθη ανάλυση:
Convicted/NNP serial/NN killer/NN Rodney/NNP Alcala/NNP may/MD have/VB had/VBN more/JJR victims/NNS ./.

Κατά την έξοδο του αρχείου από το Scene Constructor, αν επιλέξουμε μόνο την επιλογή *nn*, τότε, θα εξαχθούν τα εξής:
serial(2)
NN(2)

killer(3)
NN(3)

Άρα, κατά τη συντακτική ανάλυση, η έξοδος του Scene Constructor περιέχει τις λέξεις μαζί με τον τύπο τους. Περιέχει επίσης, αν θέλουμε, το συσχετισμό *after* [π.χ. after(3, 2)], ο οποίος δείχνει την απόσταση των δύο λέξεων, όπως και για το ποια προηγείται στο κείμενο. Π.χ. στο πιο πάνω παράδειγμα καταλαβαίνουμε τα εξής:

- Ότι η λέξη *serial* είναι ουσιαστικό (singular or mass).
- Ότι η λέξη *killer* είναι ουσιαστικό (singular or mass).
- Ότι η λέξη *serial* προηγείται σε σειρά εμφάνισης της λέξης *killer* κατά μία θέση.

Θεωρούμε ότι οι συσχετισμοί τύπου *after* είναι χρήσιμοι, γιατί, μέσα από αυτούς, μπορούμε να εξάγουμε κανόνες (με τη βοήθεια των λογισμικών Learner & Reasoner) της μορφής «*η λέξη x προηγείται πάντα της λέξης y, όταν η λέξη x είναι π.χ. ρήμα*».

Γενικά, κατά τη συντακτική ανάλυση, μέσα από το Scene Constructor έχουμε τις εξής επιλογές για την έξοδο των αποτελεσμάτων:

- Να εμφανίζονται μόνο τα μέρη του λόγου και όχι οι λέξεις.
- Να εμφανίζονται μαζί και τα μέρη του λόγου και οι λέξεις.
- Να εμφανίζονται όλες οι αποστάσεις, οι οποίες είναι μεγαλύτερες από απόσταση k (after_k) και η οποία καθορίζεται από εμάς.
- Να εμφανίζονται όλες οι αποστάσεις, οι οποίες έχουν απόσταση μικρότερη ή ίση από k (after<=k).
- Να εμφανίζονται όλες οι αποστάσεις, οι οποίες έχουν απόσταση μεγαλύτερη ή ίση από k (after>=k).
- Να εμφανίζονται όλες οι αποστάσεις των λέξεων (όλοι οι συνδυασμοί).

Για υποβοήθηση των ενεργειών μάθησης και συλλογισμού, βάλαμε και τις εξής επιλογές κατά την έξοδο:

- Να εμφανίζονται όλοι οι γενικότεροι τύποι ρημάτων (vb, vbz κτλ.) με το ίδιο όνομα (VERB). Δηλαδή, αντί να έχουμε *have(8), VB(8), had(9), VBN(9),* να έχουμε *have(8), VERB(8), had(9), VERB(9).*
- Να εξάγονται όλοι οι γενικότεροι τύποι ουσιαστικών με το κοινό όνομα *NOUN*.
- Να εξάγονται όλοι οι γενικότεροι τύποι επιθέτων με το κοινό όνομα *ADJECTIVE*.

3.1.7.5.1 Συνώνυμα (Synonym Relations)

Ανάμεσα στις διάφορες επιλογές που έχουμε στο Scene Constructor, μπορούμε να ενεργοποιήσουμε και τα συνώνυμα που, στην ουσία, είναι το επίπεδο 0_0 του *sense* 1 των υπερωνύμων της σημασιολογικής ανάλυσης.

Έχουμε, ακόμα, δύο επιλογές που έχουν σχέση με τα συνώνυμα:

- *Επιλογή μόνο συνώνυμης λέξης:*

Σε αυτήν την επιλογή, στο αρχείο εξόδου τοποθετείται μόνο η πρώτη λέξη (συνώνυμο) και όχι το αρχικό *lemma* που δόθηκε, για να επιστραφεί το συνώνυμο.

- *Επιλογή συνωνύμου και αρχικής λέξης*:

Σε αυτήν την επιλογή, στο αρχείο εξόδου φαίνεται και το *lemma* και το συνώνυμο που επεστράφη.

Γενικά, στη συντακτική ανάλυση, επειδή θεωρούμε ότι δεν είναι τόσο πλούσια η πληροφορία όσο στη σημασιολογική (από πλευράς κατά πόσο κοντά είναι στην ανθρώπινη λογική και όχι από πλευράς σημασίας και αποτελεσμάτων), επιλέξαμε να επιστρέφεται μόνο μια λέξη από τα συνώνυμα (η πρώτη) και η οποία, ανάλογα με τις επιλογές μας, θα καταχωρείται μαζί με το *lemma* ή χωρίς αυτό. Επειδή στη συντακτική ανάλυση προσπαθούμε να βγάλουμε κανόνες του τύπου «η λέξη x που ακολουθεί τη λέξη y είναι πάντα ρήμα ή πάντα ουσιαστικό ή πάντα επίθετο», δεν υπήρχε λόγος να εμπλουτιστεί με περισσότερες λέξεις και για αυτό, αποφασίσαμε να τοποθετείται μόνο η πρώτη (συνήθως η πρώτη είναι και η πιο διαδεδομένη).

3.2 Δεύτερο Στάδιο [Είσοδος κατηγορημάτων στο Learner και εξαγωγή συμπερασμάτων κοινής λογικής μέσω του Reasoner]

2

- Είσοδος κατηγορημάτων στο Learner.
- Reasoner

Όπως αναφέραμε στο κεφάλαιο Ανασκόπηση Σχετικής Βιβλιογραφίας, ο Learner αφορά τη μάθηση και ο Reasoner αφορά τη διαδικασία συλλογισμών για εξαγωγή συμπερασμάτων. Δίνουμε στο Learner τα κατηγορήματα, τα οποία δημιουργήθηκαν από το Scene Constructor και δημιουργεί θετικά και αρνητικά παραδείγματα μάθησης, με βάση κάποιο λεξικό (που του δίνουμε), το οποίο περιέχει τις λέξεις που θέλουμε να εκπαιδευτεί. Στην περίπτωσή μας, το λεξικό δημιουργείται αυτόματα, μέσω λογισμικού που δημιουργήσαμε με βάση τα κατηγορήματα που παράχθησαν με τον Scene Constructor και εμπεριέχει λέξεις, οι οποίες εμφανίζονται πιο συχνά. Ακολούθως, αφού τρέξει η διαδικασία, δημιουργείται το *knowledge file* (ολοκληρώνεται, δηλαδή, η διαδικασία που αφορά το Learner), μέσα από το οποίο μπορεί να προβλέπει, κάθε φορά, αν ισχύει ή όχι κάτι ο Reasoner (να εξάγει δηλαδή συμπεράσματα μέσω συλλογισμών).

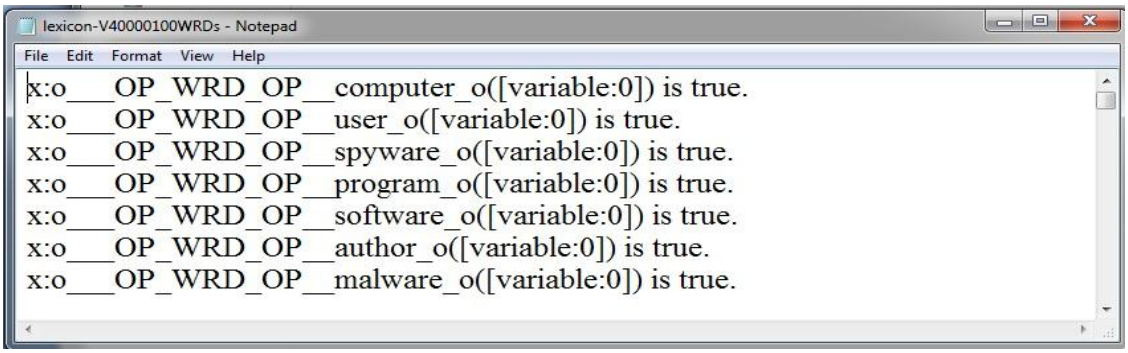
Παράδειγμα εισόδου προς το Learner

Πρώτα, καθορίζουμε το μονοπάτι που περιέχει τα αρχεία κατηγορημάτων που δημιουργήθηκαν με το λογισμικό Scene Constructor:

o___OP_WRD_OP__rogue_o([token:2]) is true. o___OP_WRD_OP__center_o([token:9]) is true. o___OP_REL_OP__show_o([token:2, token:9]) is true.	Αρχείο 1
o___OP_WRD_OP__computer_o([token:2]) is true. o___OP_WRD_OP__time_o([token:6]) is true. o___OP_REL_OP__crash_o([token:2, token:6]) is true.	Αρχείο 2
o___OP_WRD_OP__center_o([token:3]) is true. o___OP_WRD_OP__alert_o([token:7]) is true. o___OP_REL_OP__create_o([token:3, token:7]) is true.	Αρχείο 3
o___OP_WRD_OP__sunbelt_o([token:1]) is true. o___OP_WRD_OP__guarantee_o([token:6]) is true. o___OP_REL_OP__offer_o([token:1, token:6]) is true. o___OP_WRD_OP__version_o([token:11]) is true. o___OP_WRD_OP__removal_o([token:14]) is true. o___OP_REL_OP__include_o([token:11, token:14]) is true.	Αρχείο n

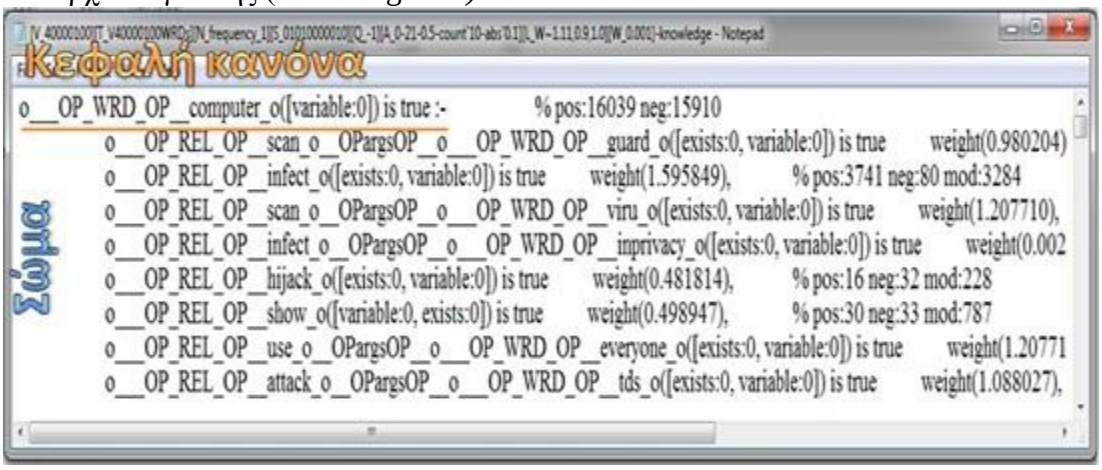
Σχήμα 3.2.α | *Παράδειγμα αρχείων με κατηγορήματα.*

Ακολούθως, δίνουμε το λεξικό (ένα αρχείο), το οποίο περιέχει τις λέξεις που μας ενδιαφέρουν και για τις οποίες θα ξεκινήσει η διαδικασία της μάθησης:



Σχήμα 3.2.β | Στιγμιότυπο από το λεξικό το οποίο δίνουμε στο Learner.

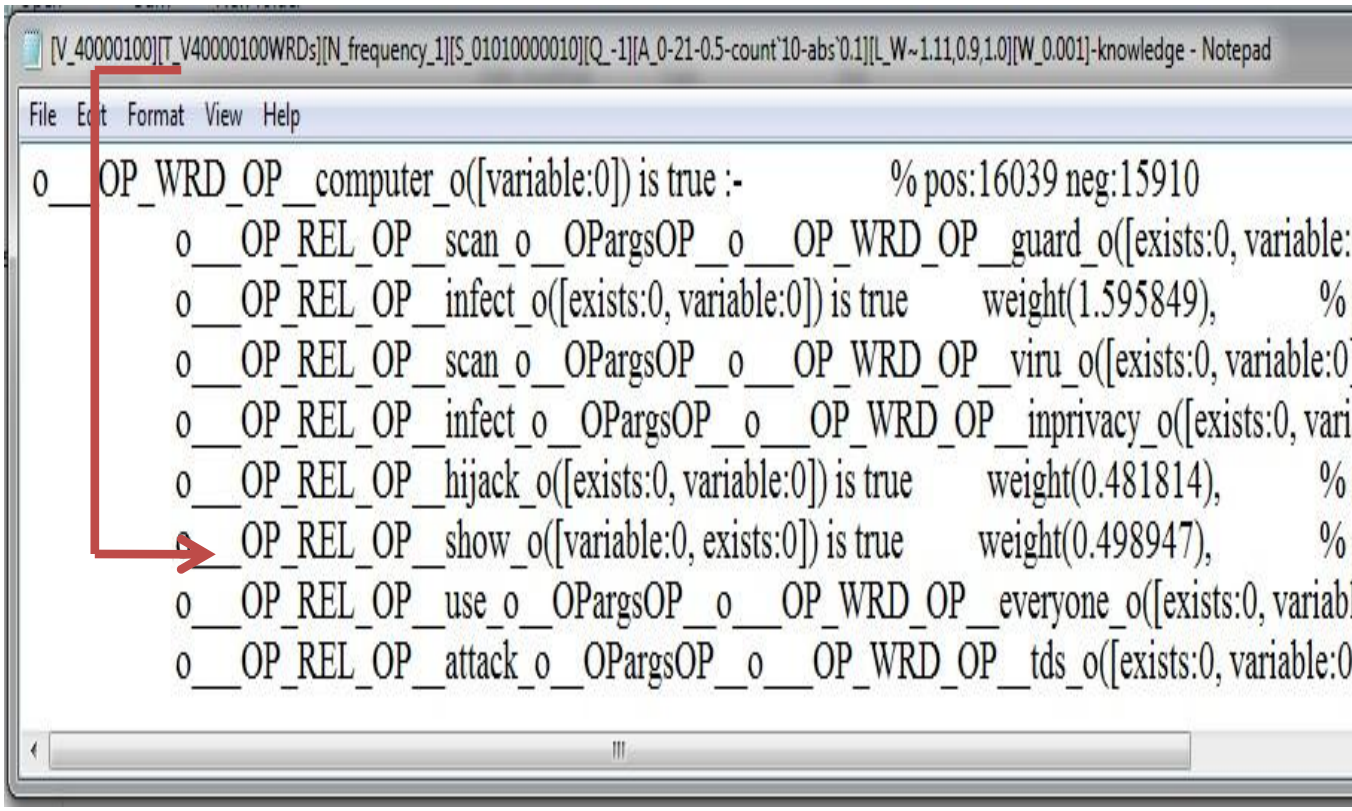
Στη συνέχεια, ξεκινάμε τη διαδικασία της μάθησης, με αποτέλεσμα τη δημιουργία του αρχείου γνώσης (knowledge file):

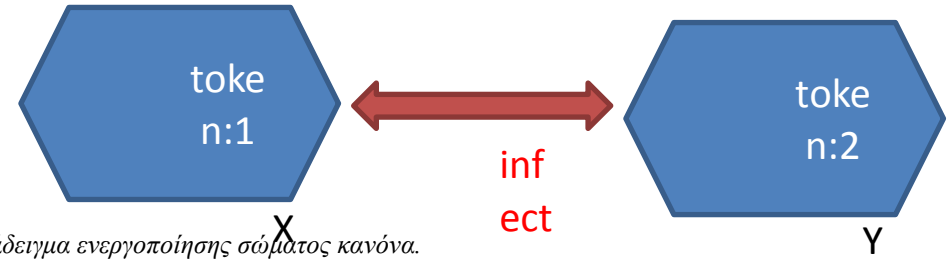


Σχήμα 3.2.γ | Στιγμιότυπο από το αρχείο γνώσης του Learner.

Το αρχείο γνώσης περιλαμβάνει κανόνες με τα σώματά τους.
Για παράδειγμα, η πρώτη γραμμή αφορά το σώμα του κανόνα (o___OP_WRD__computer_o([variable:0]) is true :-) και μας πληροφορεί ότι κάτι είναι computer, αν ισχύει κάποια πρόταση στο σώμα του κανόνα.

Ας δούμε όμως ένα παράδειγμα:





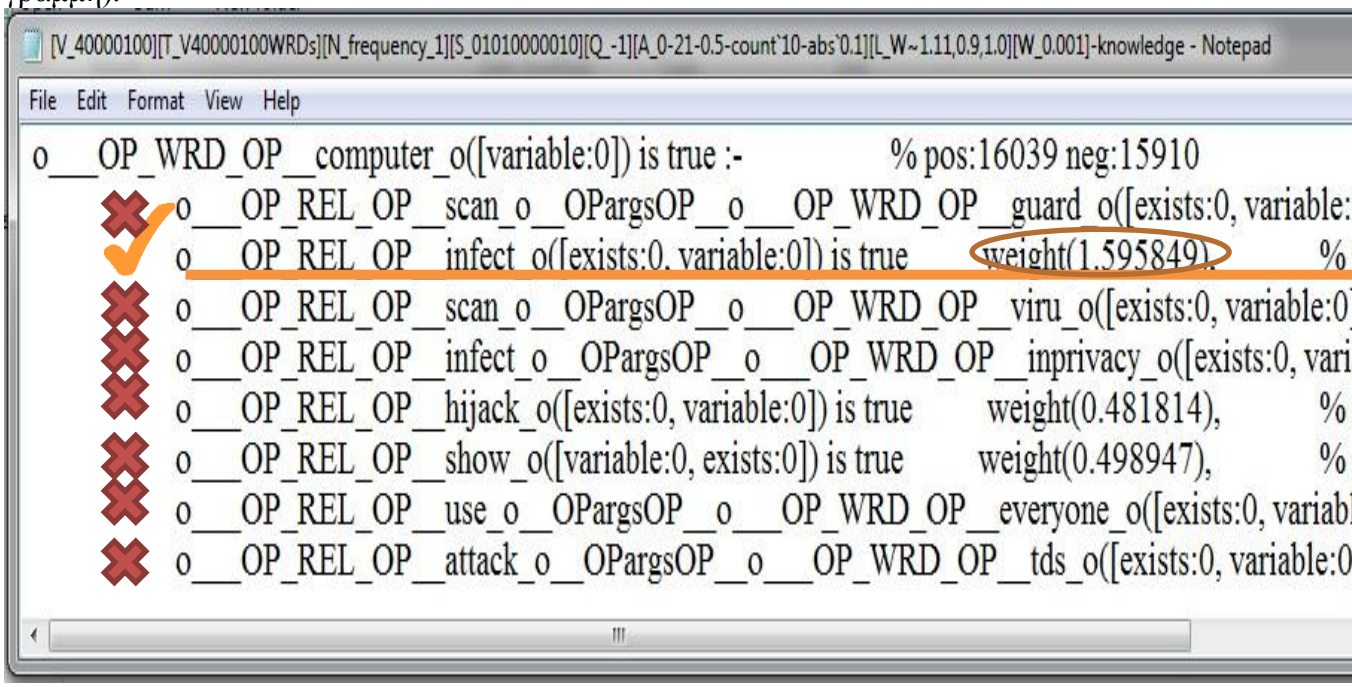
Σχήμα 3.2.δ | Παράδειγμα ενεργοποίησης σώματος κανόνα.

Εδώ πληροφορούμαστε πως, αν υπάρχει η μια οντότητα (π.χ. token:1) που έχει το χαρακτηριστικό x και κάνει *infect* μια άλλη οντότητα (π.χ. token:2) που έχει το χαρακτηριστικό y, τότε η δεύτερη οντότητα (token:2) έχει και το χαρακτηριστικό computer με βάρος 1.59. Θα πρέπει να αναφέρουμε πως το σώμα του κανόνα ενεργοποιείται όταν ισχύει όπως στην περίπτωσή μας και έχει βάρος πάνω από ένα.
Οι πιο πάνω όμως έλεγχοι, για το αν ισχύει ή όχι κάτι, δε γίνονται από το Learner, αλλά από το Reasoner, γιατί αυτό το λογισμικό κάνει τους συλλογισμούς για την επιστροφή των συμπερασμάτων κοινής λογικής. Εισάγουμε κατηγορήματα και, μέσω του Reasoner, μέσω ελέγχου του αρχείου γνώσης, γίνονται οι απαραίτητοι συλλογισμοί για εξαγωγή συμπερασμάτων. Αν ισχύει μια πρόταση στο σώμα κάποιου κανόνα με βάρος μεγαλύτερο του ενός, τότε επιστρέφεται ως συμπέρασμα η κεφαλή του αντίστοιχου κανόνα. Επίσης, αν ισχύουν κάποιες προτάσεις με συνολικό βάρος μεγαλύτερο του ενός στο σώμα κάποιου κανόνα, πάλι επιστρέφεται η κεφαλή του κανόνα ως συμπέρασμα.

Έστω ότι εισάγουμε μέσω του Reasoner τα εξής κατηγορήματα, που δηλώνουν ότι ένας ιός μόλυνε ένα σύστημα:
o__OP_REL_OP__infect([token:1, token:3]) is true.

o__OP__WRD__OP_virus_o([token:1]) is true.
o__OP__WRD__OP_system_o([token:3]) is true.

Ακολούθως, γίνονται οι απαραίτητοι συλλογισμοί μέσω του αρχείου γνώσης και ο Reasoner εντοπίζει αν ισχύει κάτι στο σώμα του κανόνα. Στο παράδειγμά μας ισχύει κάτι στο σώμα του κανόνα με βάρος 1.59 (το έχουμε υπογραμμίσει με κίτρινη γραμμή).

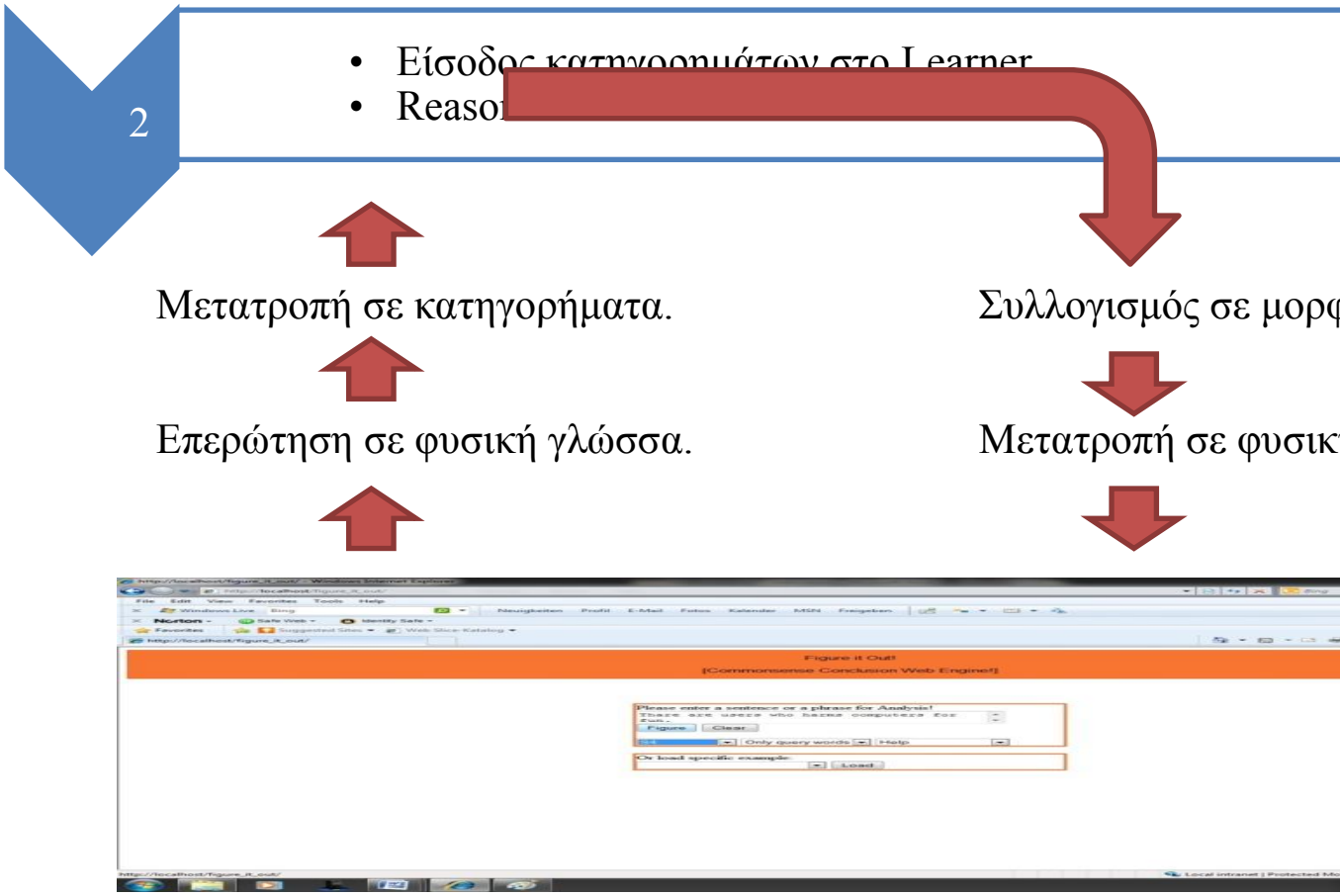


Σχήμα 3.2.ε | Παράδειγμα συλλογισμού μέσω του Reasoner.

Ακολούθως, ο Reasoner εξάγει ως συμπέρασμα κοινής λογικής το εξής:
O___OP_WRD__OP_computer__OP_conc_OP___o([token:3]) is true.

Ότι δηλαδή, η οντότητα *token:3*, εκτός από το χαρακτηριστικό *system* έχει και το χαρακτηριστικό *computer*. Ότι δηλαδή, αυτό που δέχεται επίθεση από virus είναι ένας ηλεκτρονικός υπολογιστής.

3.3 Τελικό Στάδιο [Επικοινωνία Έξυπνης Μηχανής και Reasoner]



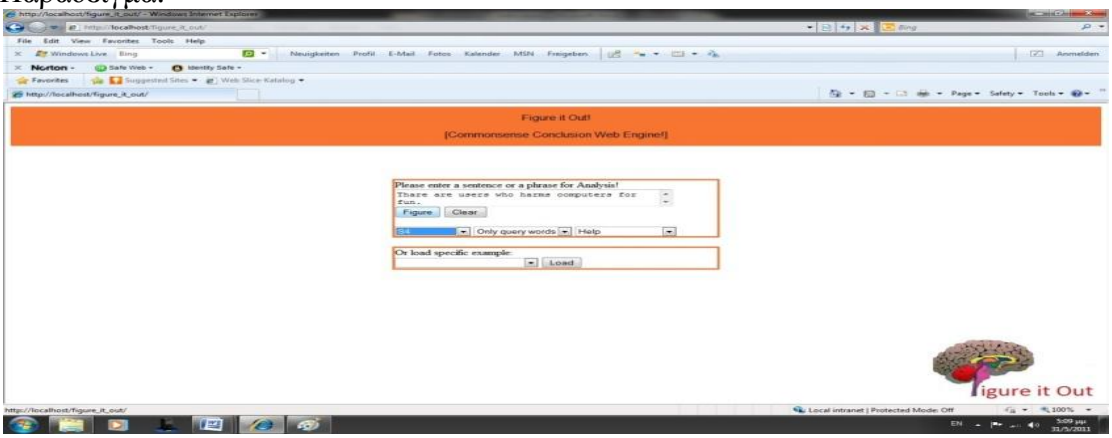
Σχήμα 3.3.α | *Τρόπος επικοινωνίας Έξυπνης Μηχανής με το Reasoner.*

Επειδή ο στόχος ήταν η δημιουργία της διαδικτυακής μηχανής συμπερασμάτων κοινής λογικής (Έξυπνης Μηχανής), μέσα από την οποία θα μπορούσαν οι χρήστες να κάνουν εξαγωγή του κρυμμένου νοήματος μιας πρότασης-φράσης, έπρεπε να συνδυαστεί η όλη διαδικασία που κάναμε μέχρι τώρα, για να είναι σε θέση η Έξυπνη Μηχανή να απαντάει, δίνοντας το κρυμμένο νόημα μιας οποιασδήποτε πρότασης ή φράσης σε μορφή κοντινή προς τη φυσική γλώσσα.

Για αυτό προγραμματίσαμε την Έξυπνη Μηχανή, η οποία κάνει τα εξής:

1. Αφού δεχθεί επερώτηση σε φυσική γλώσσα, παράγει αντίστοιχο αρχείο με την επερώτηση και το δίνει στο Stanford Parser.
2. Ακολούθως, περνάει το αρχείο εξόδου του Stanford Parser μέσα από το Scene Constructor για τη δημιουργία κατηγορημάτων.
3. Στη συνέχεια, το αρχείο εξόδου του Scene Constructor το δίνει στο Reasoner και επιστρέφεται αν υπάρχει το συμπέρασμα κοινής λογικής σε αρχείο, μέσα από έλεγχο της γνώσης που παράχθηκε με το Learner.
4. Μετά, παίρνει το αρχείο συμπερασμάτων του Reasoner που περιέχει κατηγορήματα και το δίνει σε λογισμικό με την ονομασία NLP Extractor, το οποίο δημιουργήσαμε και το οποίο μετατρέπει τα κατηγορήματα σε φυσική γλώσσα παράγοντας αντίστοιχο αρχείο.
5. Τέλος, παρουσιάζει με τη σειρά τα περιεχόμενα των πάνω αρχείων (εκτός του αρχείου εξόδου του Stanford Parser).

Παράδειγμα:



Σχήμα 3.3.β | Περιβάλλον αλληλεπίδρασης Έξυπνης Μηχανής.

Αρχικά, ο επισκέπτης στην ιστοσελίδα της Έξυπνης Μηχανής (χρήστης) εισάγει την επερώτηση και επιλέγει έναν κανόνα (έχει σχέση με επιλογή υπερωνύμων, επίπεδο βάθους και γενικά του τρόπου δημιουργίας των κατηγορημάτων), με τον οποίο θα τρέξει στη συνέχεια το λογισμικό του Scene Constructor, για να τροποποιηθεί η επερώτηση πριν δοθεί στο Reasoner.

Όνομα κανόνα	Ρυθμίσεις που ισχύουν
S1	• Ενεργοποιημένη η σημασιολογική ανάλυση. • Ενεργοποιημένο το lemmatization. • Ενεργοποιημένα τα υπερώνυμα (depth 0-0, sense 1). • Επιλογή όλων των υπερωνύμων του επιπέδου. • Ενεργοποιημένοι και οι 13 κανόνες δημιουργίας κατηγορημάτων.
S2	• Ενεργοποιημένη η σημασιολογική ανάλυση. • Ενεργοποιημένο το lemmatization. • Ενεργοποιημένα τα υπερώνυμα (depth 0-0, sense 1). • Επιλογή όλων των υπερωνύμων του επιπέδου. • Ενεργοποιημένοι μόνο οι δυαδικοί κανόνες δημιουργίας κατηγορημάτων.
S3	• Ενεργοποιημένη η σημασιολογική ανάλυση. • Ενεργοποιημένο το lemmatization. • Ενεργοποιημένα τα υπερώνυμα (depth 0-0, sense 1). • Επιλογή όλων των υπερωνύμων του επιπέδου. • Ενεργοποιημένος μόνο ο κανόνας nsubj-dobj για τη δημιουργία κατηγορημάτων.
S4	• Ενεργοποιημένη η σημασιολογική ανάλυση. • Ενεργοποιημένο το lemmatization. • Ενεργοποιημένοι και οι 13 κανόνες δημιουργίας κατηγορημάτων.
S5	• Ενεργοποιημένη η σημασιολογική ανάλυση. • Ενεργοποιημένο το lemmatization. • Ενεργοποιημένοι μόνο οι δυαδικοί κανόνες δημιουργίας κατηγορημάτων.

S6	• Ενεργοποιημένη η σημασιολογική ανάλυση. • Ενεργοποιημένο το lemmatization. • Ενεργοποιημένος μόνο ο κανόνας nsubj-dobj για τη δημιουργία κατηγορημάτων.
S7	• Ενεργοποιημένη η σημασιολογική ανάλυση. • Ενεργοποιημένοι και οι 13 κανόνες δημιουργίας κατηγορημάτων.
S8	• Ενεργοποιημένη η σημασιολογική ανάλυση. • Ενεργοποιημένοι μόνο οι δυαδικοί κανόνες δημιουργίας κατηγορημάτων.
S9	• Ενεργοποιημένη η σημασιολογική ανάλυση. • Ενεργοποιημένος μόνο ο κανόνας nsubj-dobj για τη δημιουργία κατηγορημάτων.
S10	• Ενεργοποιημένη η σημασιολογική ανάλυση. • Ενεργοποιημένο το lemmatization. • Ενεργοποιημένα τα υπερώνυμα (depth 0-0, sense 1). • Επιλογή μόνο της πρώτης λέξης του επιπέδου των υπερωνύμων. • Ενεργοποιημένοι και οι 13 κανόνες δημιουργίας κατηγορημάτων.
S11	• Ενεργοποιημένη η σημασιολογική ανάλυση. • Ενεργοποιημένο το lemmatization. • Ενεργοποιημένα τα υπερώνυμα (depth 0-0, sense 1). • Επιλογή μόνο της πρώτης λέξης του επιπέδου των υπερωνύμων. • Ενεργοποιημένοι μόνο οι δυαδικοί κανόνες δημιουργίας κατηγορημάτων.
S12	• Ενεργοποιημένη η σημασιολογική ανάλυση. • Ενεργοποιημένο το lemmatization. • Ενεργοποιημένα τα υπερώνυμα (depth 0-0, sense 1). • Επιλογή της πιο σπάνιας λέξης του επιπέδου των υπερωνύμων. • Ενεργοποιημένοι μόνο οι δυαδικοί κανόνες δημιουργίας κατηγορημάτων.

Σχήμα 3.3.γ | *Κανόνες τροποποίησης κατηγορημάτων.*

Υποθέτουμε ότι ο χρήστης εισάγει την πρόταση (επιλέγοντας τον κανόνα S4):
There are users who harms computers for fun.

Αν θέλει τα αποτελέσματα να εξάγονται μόνο για λέξεις του ερωτήματος στην Έξυπνη Μηχανή στην επιλογή *NL Options*, επιλέγει *Only query words* ή αν θέλει να εξάγονται και αποτελέσματα για τα υπερώνυμα που δεν υπάρχουν στο αρχικό ερώτημα στο *NL Options*, επιλέγει *All words*.

Υποθέτουμε ότι επιλέγει:
Only query words.

Ακολούθως, αφού επιλεγεί το *submit*, γίνεται έλεγχος για την ύπαρξη κενού ερωτήματος ή και για εξ ολοκλήρου αριθμητική τιμή (δηλαδή, αν ο χρήστης εισήγαγε μόνο αριθμούς). Σε περίπτωση που διαπιστωθεί ότι ισχύει κάτι από τα πιο πάνω, τότε ο χρήστης ενημερώνεται με ανάλογο μήνυμα για να εισάγει κείμενο (ή συνδυασμό κειμένου με αριθμούς).

Στη συνέχεια, περνάει το ερώτημα μέσα από το Stanford Parser και παράγεται αντίστοιχο αρχείο εξόδου.

Στο παράδειγμά μας εξάγεται το ακόλουθο αρχείο:
There/EX are/VBP users/NNS who/WP harms/VBZ computers/NNS for/IN fun/NN ./.
expl(are-2, There-1)
nsubj(are-2, users-3)
rel(harms-5, who-4)
rcmod(users-3, harms-5)

dobj(harms-5, computers-6)
prep_for(computers-6, fun-8)

Ακολούθως, το αρχείο εξόδου του Stanford Parser περνάει μέσα από το Scene Constructor. Σε περίπτωση που δεν μπορεί να εξαχθεί κάτι, τότε ενημερώνεται ο χρήστης, για να ξανατρέξει το ερώτημα, επιλέγοντας έναν άλλον κανόνα διαμόρφωσης του Scene Constructor, αλλιώς περνάει το αρχείο εξόδου μέσα στο Reasoner.

Στο παράδειγμά μας το αρχείο εξόδου του Scene Constructor έχει τα εξής κατηγορήματα:
o___OP_WRD_OP__user_o([token:3]) is true.
o___OP_WRD_OP__computer_o([token:6]) is true.
o___OP_REL_OP__harm_o([token:3, token:6]) is true.

Ο Reasoner, με τη σειρά του, αν δώσει κάποιο αρχείο με συμπεράσματα κοινής λογικής, τότε η Έξυπνη Μηχανή περνάει το αρχείο με τα συμπεράσματα προς το λογισμικό NLP Extractor, για να εξαχθούν τα αποτελέσματα σε φυσική γλώσσα, αλλιώς ενημερώνεται ο χρήστης για να καταχωρήσει άλλο ερώτημα ή να ξανατρέξει το Scene Constructor με άλλον κανόνα διαμόρφωσης (S_).

Στο παράδειγμα το αρχείο συμπερασμάτων κοινής λογικής του Reasoner περιέχει τα εξής:
o___OP_WRD_OP__computer__OP_conc_OP___o([token:6]) is true.
o___OP_WRD_OP__system__OP_conc_OP___o([token:6]) is true.
o___OP_WRD_OP__pc__OP_conc_OP___o([token:6]) is true.
o___OP_WRD_OP__rat__OP_conc_OP___o([token:3]) is true.

Με το τρέξιμο του NLP Extractor συνδυάζεται το αποτέλεσμα του Reasoner με το αποτέλεσμα που παράχθηκε από το Scene Constructor και γίνεται προσπάθεια εξαγωγής των αποτελεσμάτων προς το χρήστη, σε μορφή κοντινή προς τη φυσική γλώσσα. Απλά, εντοπίζονται τα συμπεράσματα, τα οποία βρίσκονται σε μορφή μοναδιαίων κατηγορημάτων, που παράχθησαν από το Reasoner και συνδυάζονται με τα δυαδικά και μοναδιαία κατηγορήματα που παράχθησαν από το Scene Constructor και αναφέρονται στις ίδιες οντότητες (token:_).

Στο παράδειγμά μας τα κατηγορήματα μετατρέπονται ως εξής:
user [is a kind of || is || is similar with] **rat.**
rat harm computer.
computer [is a kind of || is || is similar with] **system**.
user harm **system**.
computer [is a kind of || is || is similar with] **pc**.
user harm **pc**.

Τέλος, η Έξυπνη Μηχανή επιστρέφει συνολικά τις πάνω πληροφορίες (εκτός το αρχείο εξόδου του Stanford Parser) ως εξής:

User Query

"There are users who harms computers for fun."

Parsed Query Results

o___OP_WRD_OP__user_o([token:3]) is true.
o___OP_WRD_OP__computer_o([token:6]) is true.
o___OP_REL_OP__harm_o([token:3, token:6]) is true.

Reasoner Results

o___OP_WRD_OP__computer__OP_conc_OP___o([token:6]) is true.
o___OP_WRD_OP__system__OP_conc_OP___o([token:6]) is true.
o___OP_WRD_OP__pc__OP_conc_OP___o([token:6]) is true.
o___OP_WRD_OP__rat__OP_conc_OP___o([token:3]) is true.

Output Results In Natural Language

user [is a kind of || is || is similar with] **rat**.

rat harm computer.

computer [is a kind of || is || is similar with] **system**.

user harm **system**.

computer [is a kind of || is || is similar with] **pc**.

user harm **pc**.

Σχήμα 3.3.δ | Επιστροφή συμπερασμάτων κοινής λογικής από την Έξυπνη Μηχανή σε επερώτηση χρήστη.

4

Πειράματα

Τι έδειξαν τα Πειράματα;

Σε αυτό το κεφάλαιο παρουσιάζουμε τη διεξαγωγή πειραμάτων σε δύο φάσεις. Στην Α΄ φάση κατεβάσαμε και επεξεργαστήκαμε 1000 ιστοσελίδες από το διαδίκτυο, ενώ στη Β΄ φάση 40000 ιστοσελίδες. Σε κάθε φάση παρουσιάζουμε αναλυτικά και τα αποτελέσματα στα οποία καταλήξαμε.

Αρχικά, να αναφέρουμε πως η διαδικασία των πειραμάτων εκτελέστηκε σε δύο φάσεις. Κατά την πρώτη φάση, μέσω του λογισμικού Manager κατεβάσαμε και επεξεργαστήκαμε 1000 αρχεία και κατά τη δεύτερη φάση 40000. Και στις δύο περιπτώσεις επιλέξαμε συγκεκριμένο θέμα (spyware), για την αναζήτηση των πληροφοριών (ιστοσελίδων), με κύριο στόχο τη σύγκριση των αποτελεσμάτων των δύο φάσεων.

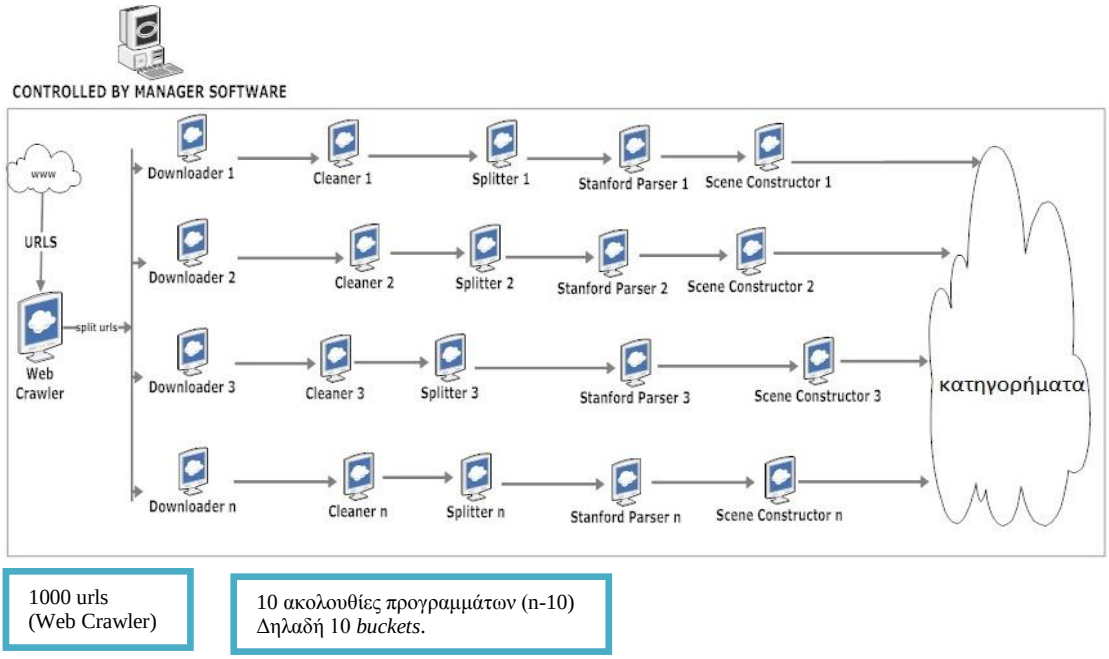
Κατά τη διάρκεια της παρουσίασης των αποτελεσμάτων να έχουμε συνεχώς υπόψη το θέμα, για το οποίο επιλέγηκαν οι πληροφορίες από το διαδίκτυο. Αυτό το λέμε, γιατί, κατά τη διαδικασία των συλλογισμών και μέσω της αλληλεπίδρασης της με τους χρήστες, η Έξυπνη Μηχανή είναι λογικό να «σκέφτεται» μέσα σε ένα ευρύ φάσμα του *domain spyware* (είναι σαν να έχουμε ένα παιδί, το οποίο αναλαμβάνουμε να μορφώσουμε μόνο με πληροφορίες γύρω από την ευρύτερη έννοια *spyware*).

Η επιλογή του θέματος μάς είχε απασχολήσει αρκετά, αλλά επιλέξαμε το συγκεκριμένο, γιατί είναι ένας χώρος, τον οποίο γνωρίζαμε και ειδικότερα, γιατί, μέσω ενός γνωστού θέματος, θα μπορούσαμε να ελέγχαμε την ποιότητα, αλλά και την ορθότητα των συμπερασμάτων που θα επέστρεφε η Έξυπνη Μηχανή.

4.1 *Α΄ φάση πειραμάτων*

Αφού κατεβάσαμε 1000 *html* αρχεία, τα επεξεργαστήκαμε με δεκατρείς συνολικά διαφορετικές ρυθμίσεις και, στη συνέχεια, πειραματιζόμαστε με την Έξυπνη Μηχανή (ξεχωριστά για την καθεμιά ρύθμιση, μέσω συγκεκριμένων επερωτήσεων). Στόχος ήταν ο εντοπισμός ποιοτικών συμπερασμάτων κοινής λογικής, αλλά και ο έλεγχος και η εύρεση του συνόλου των κατηγορημάτων με τις πιο ποιοτικές πληροφορίες (αυτό, δηλαδή, με το οποίο καταφέραμε να μορφώσουμε καλύτερα την Έξυπνη Μηχανή).

Αυτή η διαδικασία φαίνεται καλύτερα στο επόμενο σχήμα:



Σχήμα 4.1.α | *Τρέξιμο πειραμάτων.*

Όπως φαίνεται από το σχήμα 4.1.α, μέσω του Manager θέσαμε τη δημιουργία δέκα *buckets* και ξεκινήσαμε τη διαδικασία. Επειδή ο Downloader, ο Cleaner και ο Stanford Parser έκαναν πάντα την ίδια δουλειά (με τις ίδιες ρυθμίσεις), επικεντρωθήκαμε στις ρυθμίσεις του Splitter και, κυρίως, του Scene Constructor όπου είχαμε τα εξής:

- Ο Splitter είχε εντολή να διαμοιράζει το αρχείο εισόδου (αυτό που παράχθηκε με το Cleaner) και να δημιουργεί ένα καινούργιο αρχείο για κάθε πρόταση.
- Ο Scene Constructor, είχε εντολή να δημιουργήσει κατηγορήματα με τις εξής ρυθμίσεις:
    - Ενεργοποιημένο *lemmatization.*
    - Ενεργοποιημένα τα υπερώνυμα (sense 1, depth 0-0, επιλογή όλων των λέξεων).
    - Ενεργοποιημένοι και οι δεκατρείς σημασιολογικοί κανόνες δημιουργίας κατηγορημάτων.

Ακολούθως, με το τέλος της διαδικασίας, όλα τα αρχεία, που δημιουργήθηκαν στα δέκα *buckets* (μέσω λογισμικού που δημιουργήσαμε), μεταφέρθηκαν σε ένα φάκελο με την ονομασία V2 και ξεκινήσαμε τη διαδικασία της μάθησης, μέσω του λογισμικού Learner. Ο Learner έτρεξε με συγκεκριμένο λεξικό, το οποίο παράχθηκε με λογισμικό που δημιουργήσαμε και το οποίο περιείχε τις εκατό πιο σημαντικές

λέξεις, ανάμεσα στα κατηγορήματα που δημιουργήθηκαν (αυτές που εμφανίζονταν τις περισσότερες φορές). Στη συνέχεια, με επερωτήσεις μέσω της Έξυπνης Μηχανής, ελέγχαμε τα επιστρεφόμενα συμπεράσματα αξιολογώντας την ποιότητά τους.

Επειδή θέλαμε να δοκιμάσουμε και τα αποτελέσματα, μέσω αλλαγών στις ρυθμίσεις του Scene Constructor (για να δούμε τι ρόλο έπαιζαν οι αλλαγές στη δημιουργία των κατηγορημάτων), ξανατρέξαμε την πιο πάνω διαδικασία (βλέπε σχήμα 4.1.α) από το τελευταίο σημείο (δηλαδή, τρέχοντας ξανά δέκα στιγμιότυπα του λογισμικού Scene Constructor). Συνολικά, τρέξαμε την πιο πάνω διαδικασία δεκατρείς φορές και δημιουργήσαμε δεκατρείς φακέλους, συμπεριλαμβανομένου και του V2, που περιείχαν κατηγορήματα, τα οποία δημιουργήθηκαν με τις εξής ρυθμίσεις (για ευκολία αναφοράς βλέπε σχήμα 4.1.β):

- V3 [L, Y, Y_ALL, S_2]
- V4 [L, Y, Y_ALL, S_ND]
- V5 [L, Y, Y_F, S_ALL]
- V6 [L, Y, Y_F, S_ND]
- V7 [L, Y, Y_F, S_2]
- V8 [L, Y, Y_R, S_ND]
- V9 [L, S_ALL]
- V10 [L, S_2]
- V11 [L, S_ND]
- V12 [S_ALL]
- V13 [S_2]
- V14 [S_ND]

Συμβολισμοί:

Υπερώνυμα	Y
Lemmatization	L
Επιλογή όλων των λέξεων στα υπερώνυμα.	Y_ALL
Επιλογή μόνο της πρώτης λέξης στα υπερώνυμα.	Y_F
Επιλογή μόνο της σπάνιας λέξης στα υπερώνυμα.	Y_R
Επιλογή και των δεκατριών σημασιολογικών κανόνων δημιουργίας κατηγορημάτων.	S_ALL
Επιλογή μόνο των σημασιολογικών κανόνων που παράγουν δυαδικά κατηγορήματα.	S_2
Επιλογή μόνο του σημασιολογικού κανόνα *nsubj-dobj* δημιουργίας κατηγορημάτων.	S_ND
Σε όσες περιπτώσεις έχουμε ενεργοποιημένα τα υπερώνυμα, αυτό σημαίνει πως έχουμε βάθος (depth) υπερωνύμων 0-0 και έννοια (sense) 1.	

Σχήμα 4.1.β | *Συμβολισμοί για ευκολία αναφοράς.*

Στη συνέχεια, θέσαμε επερωτήσεις στην Έξυπνη Μηχανή που επιλέγηκαν τυχαία και οι οποίες είχαν έμμεσα σχέση με το *spyware*, όπως:
«Adware refers to software that harms computers».
«Spyware is a type of malware that can be installed on computers, and which collects small pieces of information about users without their knowledge».
«Spyware programs can collect various types of personal information, such as Internet surfing habits and sites that have been visited».

«Computer installed something».
«t installed something».
«The term adware frequently refers to any software which displays advertisements, whether or not the user has consented».
«Many spyware programs display advertisements».

Αφού ελέγξαμε τα αποτελέσματα, εντοπίσαμε τα εξής:

- Γενικά, δεν υπήρχαν ποιοτικά αποτελέσματα.
- Στις περισσότερες περιπτώσεις δεν παίρναμε απάντηση από την Έξυπνη Μηχανή.
- Τα μόνα σύνολα κατηγορημάτων που έδωσαν θετικές ενδείξεις ήταν τα V5, V9 και V10.

4.2 *Β΄ φάση πειραμάτων*

Επειδή τα αποτελέσματα που πήραμε δεν ήταν ποιοτικά, αποφασίσαμε να μεγαλώσουμε το δείγμα μας και έτσι κατεβάσαμε και επεξεργαστήκαμε 40000 αρχεία. Αν και χρειάστηκε πολύ περισσότερος χρόνος επεξεργασίας (βλέπε Παράρτημα, Χρήσιμες τεχνικές πληροφορίες για μελλοντική δουλειά, σελίδα 100), πήραμε πολύ καλά αποτελέσματα.

Πιο κάτω, μέσα από αρκετά παραδείγματα, θα επιβεβαιώσουμε την ικανότητα της Έξυπνης Μηχανής, όσον αφορά τη διαδικασία επιστροφής συμπερασμάτων κοινής λογικής. Σίγουρα, υπάρχουν και περιπτώσεις, όπου οι συλλογισμοί της μηχανής δεν είναι ποιοτικοί και επιβεβαιώνουν τη μεγάλη ερευνητική δουλειά που χρειάζεται να γίνει ακόμη στο συγκεκριμένο τομέα (βλέπε Συμπεράσματα και Μελλοντική δουλειά, σελίδα 75).

Για τη διεξαγωγή των πειραμάτων και συγκεκριμένα για σκοπούς ελέγχου, προσθέσαμε ακόμα ένα σύνολο κατηγορημάτων και συγκεκριμένα, το V10n. Το V10n περιείχε τις ίδιες ρυθμίσεις με το V10, εκτός του τελευταίου σημασιολογικού κανόνα, ο οποίος αφορούσε τη δημιουργία δυαδικών κατηγορημάτων, με βάση συσχετισμό μέσω ρημάτων. (Το κάναμε αυτό, γιατί θέλαμε να δούμε την αναγκαιότητα αυτού του κανόνα σε σχέση με τους υπόλοιπους).

Πιο κάτω, θα παραθέσουμε μερικές από τις επερωτήσεις που θέσαμε στην Έξυπνη Μηχανή, μαζί με τα συμπεράσματα κοινής λογικής που επιστρέφονταν, με σχολιασμό στο τέλος για τα αποτελέσματα.

User Query

"Members sharing something."

Parsed Query Results

o___OP_WRD_OP__member_o([token:1]) is true.
o___OP_WRD_OP__something_o([token:3]) is true.
o___OP_REL_OP__share_o([token:1, token:3]) is true.

Reasoner Results

o___OP_WRD_OP__article__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__file__OP_conc_OP___o([token:3]) is true.

Output Results In Natural Language

something [is a kind of || is || is similar with] **article**.

member share **article**.

something [is a kind of || is || is similar with] **file**.

member share **file**.

User Query

"spyware harms systems."

Parsed Query Results

o___OP_WRD_OP__spyware_o([token:1]) is true.
o___OP_WRD_OP__system_o([token:3]) is true.
o___OP_REL_OP__harm_o([token:1, token:3]) is true.

Reasoner Results

o___OP_WRD_OP__computer__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__system__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__pc__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__rat__OP_conc_OP___o([token:1]) is true.

Output Results In Natural Language

spyware [is a kind of || is || is similar with] **rat**.

rat harm system.

system [is a kind of || is || is similar with] **computer**.

spyware harm **computer**.

system [is a kind of || is || is similar with] **pc**.

spyware harm **pc**.

User Query

"There are people who steal from others."

Parsed Query Results

o___OP_WRD_OP__people_o([token:3]) is true.
o___OP_WRD_OP__other_o([token:7]) is true.
o___OP_REL_OP__steal_o([token:3, token:7]) is true.

Reasoner Results

o___OP_WRD_OP__hacker__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__thief__OP_conc_OP___o([token:3]) is true.

Output Results In Natural Language

people [is a kind of || is || is similar with] **hacker**.

hacker steal other.

people [is a kind of || is || is similar with] **thief**.

thief steal other.

User Query

"something attacked computer."

Parsed Query Results

o___OP_WRD_OP__something_o([token:1]) is true.
o___OP_WRD_OP__computer_o([token:3]) is true.
o___OP_REL_OP__attack_o([token:1, token:3]) is true.

Reasoner Results

o___OP_WRD_OP__viru__OP_conc_OP___o([token:1]) is true.

Output Results In Natural Language

something [is a kind of || is || is similar with] **viru**.

viru attack computer.

User Query

"Users pay with something."

Parsed Query Results

o___OP_WRD_OP__user_o([token:1]) is true.
o___OP_WRD_OP__something_o([token:4]) is true.
o___OP_REL_OP__pay_o([token:1, token:4]) is true.

Reasoner Results

o___OP_WRD_OP__money__OP_conc_OP___o([token:4]) is true.

Output Results In Natural Language

something [is a kind of || is || is similar with] **money**.

user pay **money**.

User Query

"Hackers inject users."

Parsed Query Results

o___OP_WRD_OP__hacker_o([token:1]) is true.
o___OP_WRD_OP__user_o([token:3]) is true.
o___OP_REL_OP__inject_o([token:1, token:3]) is true.

Reasoner Results

o___OP_WRD_OP__code__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__ad__OP_conc_OP___o([token:3]) is true.

Output Results In Natural Language

user [is a kind of || is || is similar with] **code**.

hacker inject **code**.

user [is a kind of || is || is similar with] **ad**.

hacker inject **ad**.

User Query

"Thare are users who harms computers for fun."

Parsed Query Results

o___OP_WRD_OP__user_o([token:3]) is true.
o___OP_WRD_OP__computer_o([token:6]) is true.
o___OP_REL_OP__harm_o([token:3, token:6]) is true.

Reasoner Results

o___OP_WRD_OP__computer__OP_conc_OP___o([token:6]) is true.
o___OP_WRD_OP__system__OP_conc_OP___o([token:6]) is true.
o___OP_WRD_OP__pc__OP_conc_OP___o([token:6]) is true.
o___OP_WRD_OP__rat__OP_conc_OP___o([token:3]) is true.

Output Results In Natural Language

user [is a kind of || is || is similar with] **rat**.

rat harm computer.

computer [is a kind of || is || is similar with] **system**.

user harm **system**.

computer [is a kind of || is || is similar with] **pc**.

user harm **pc**.

User Query

"Spyware can alert everyone."

Parsed Query Results

o___OP_WRD_OP__spyware_o([token:1]) is true.
o___OP_WRD_OP__everyone_o([token:4]) is true.
o___OP_REL_OP__alert_o([token:1, token:4]) is true.

Reasoner Results

o___OP_WRD_OP__security__OP_conc_OP___o([token:1]) is true.

Output Results In Natural Language

spyware [is a kind of || is || is similar with] **security**.

security alert everyone.

User Query

"virus installed something."

Parsed Query Results

o___OP_WRD_OP__viru_o([token:1]) is true.
o___OP_WRD_OP__something_o([token:3]) is true.
o___OP_REL_OP__instal_o([token:1, token:3]) is true.
o___OP_REL_OP__install_o([token:1, token:3]) is true.
o___OP_REL_OP__put_in_o([token:1, token:3]) is true.
o___OP_REL_OP__set_up_o([token:1, token:3]) is true.

Reasoner Results

o___OP_WRD_OP__software__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__trojan__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__adware__OP_conc_OP___o([token:3]) is true.

Output Results In Natural Language

something [is a kind of || is || is similar with] **software**.

viru instal **software**.

something [is a kind of || is || is similar with] **trojan**.

viru instal **trojan**.

something [is a kind of || is || is similar with] **adware**.

viru instal **adware**.

User Query

"spyware attacked user."

Parsed Query Results

o___OP_WRD_OP__spyware_o([token:1]) is true.
o___OP_WRD_OP__user_o([token:3]) is true.
o___OP_REL_OP__attack_o([token:1, token:3]) is true.
o___OP_REL_OP__assail_o([token:1, token:3]) is true.

Reasoner Results

o___OP_WRD_OP__system__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__PC__OP_conc_OP___o([token:3]) is true.

Output Results In Natural Language

user [is a kind of || is || is similar with] **system**.

spyware attack s**ystem**.

user [is a kind of || is || is similar with] **PC**.

spyware attack **PC**.

User Query

"t installed something."

Parsed Query Results

o___OP_WRD_OP__t_o([token:1]) is true.
o___OP_WRD_OP__something_o([token:3]) is true.
o___OP_REL_OP__instal_o([token:1, token:3]) is true.
o___OP_WRD_OP__thymine_o([token:1]) is true.
o___OP_REL_OP__install_o([token:1, token:3]) is true.
o___OP_REL_OP__put_in_o([token:1, token:3]) is true.
o___OP_REL_OP__set_up_o([token:1, token:3]) is true.

Reasoner Results

o___OP_WRD_OP__trojan__OP_conc_OP___o([token:1]) is true.
o___OP_WRD_OP__parasite__OP_conc_OP___o([token:1]) is true.

Output Results In Natural Language

t [is a kind of || is || is similar with] **trojan**.

trojan instal something.

t [is a kind of || is || is similar with] **parasite.**

parasite instal something.

User Query

"Many spyware programs display advertisments."

Parsed Query Results

o___OP_WRD_OP__program_o([token:3]) is true.
o___OP_WRD_OP__advertisment_o([token:5]) is true.
o___OP_REL_OP__display_o([token:3, token:5]) is true.
o___OP_WRD_OP__programme_o([token:3]) is true.
o___OP_REL_OP__expose_o([token:3, token:5]) is true.
o___OP_REL_OP__exhibit_o([token:3, token:5]) is true.

Reasoner Results

o___OP_WRD_OP__trojan__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__application__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__rogue__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__antiviru__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__parasite__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__threat__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__defender__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__site__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__worm__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__pro__OP_conc_OP___o([token:3]) is true.
o___OP_WRD_OP__scanner__OP_conc_OP___o([token:3]) is true.

Output Results In Natural Language

program [is a kind of || is || is similar with] **trojan**.

trojan display advertisment.

program [is a kind of || is || is similar with] **application**.

application display advertisment.

program [is a kind of || is || is similar with] **rogue**.

rogue display advertisment.

program [is a kind of || is || is similar with] **antiviru**.

antiviru display advertisment.

program [is a kind of || is || is similar with] **parasite**.

parasite display advertisment.

program [is a kind of || is || is similar with] **threat**.

threat display advertisment.

program [is a kind of || is || is similar with] **defender**.

defender display advertisment.

program [is a kind of || is || is similar with] **site**.

site display advertisment.

program [is a kind of || is || is similar with] **worm**.

worm display advertisment.

program [is a kind of || is || is similar with] **pro**.

pro display advertisment.

program [is a kind of || is || is similar with] **scanner**.

scanner display advertisment.

Μέσα από τα πιο πάνω αποτελέσματα, φαίνεται ξεκάθαρα ότι η Έξυπνη Μηχανή είναι σε θέση να αλληλεπιδρά με τους χρήστες και να εξάγει καλά συμπεράσματα κοινής λογικής.

Ας δούμε, όμως, και μερικά αποτελέσματα, τα οποία μπορεί να συγχύσουν το χρήστη και δεν μπορούν να επιστρέψουν ποιοτικά συμπεράσματα κοινής λογικής.

Στο επόμενο παράδειγμα φαίνεται ξεκάθαρα ότι η Έξυπνη Μηχανή συγχύζει τον υπολογιστή με το spyware. Το ελαφρυντικό, σε αυτήν την περίπτωση, είναι ότι η μηχανή ξέρει πράγματα γύρω από την ευρεία έννοια *spyware* και θεωρεί πως, ό,τι κάνει εγκατάσταση, είναι *spyware*, *trojan* ή και *user*.

User Query
"Computer installed something."

Parsed Query Results

o___OP_WRD_OP__computer_o([token:1]) is true.
o___OP_WRD_OP__something_o([token:3]) is true.
o___OP_REL_OP__instal_o([token:1, token:3]) is true.
o___OP_WRD_OP__computing_machine_o([token:1]) is true.
o___OP_WRD_OP__computing_device_o([token:1]) is true.
o___OP_WRD_OP__data_processor_o([token:1]) is true.
o___OP_WRD_OP__electronic_computer_o([token:1]) is true.
o___OP_WRD_OP__information_processing_system_o([token:1]) is true.
o___OP_REL_OP__install_o([token:1, token:3]) is true.

o___OP_REL_OP__put_in_o([token:1, token:3]) is true.
o___OP_REL_OP__set_up_o([token:1, token:3]) is true.

Reasoner Results

o___OP_WRD_OP__users__OP_conc_OP___o([token:1]) is true.
o___OP_WRD_OP__trojan__OP_conc_OP___o([token:1]) is true.
o___OP_WRD_OP__t__OP_conc_OP___o([token:1]) is true.

Output Results In Natural Language

computer [is a kind of || is || is similar with] **users**.

users instal something.

computer [is a kind of || is || is similar with] **trojan**.

trojan instal something.

computer [is a kind of || is || is similar with] **t.**

t instal something.

Στο επόμενο παράδειγμα, βγάζει πάρα πολλά αποτελέσματα, που στο τέλος συγχύζουν το χρήστη, αφού εμπεριέχουν και κάποια συμπεράσματα που δεν είναι ποιοτικά (π.χ. *lot is kind of number, lot is kind of ad, damage is kind of lot*).

User Query
"spyware can cause a lot of damage."

Parsed Query Results
o___OP_WRD_OP__spyware_o([token:1]) is true.
o___OP_WRD_OP__lot_o([token:5]) is true.
o___OP_REL_OP__cause_o([token:1, token:5]) is true.
o___OP_WRD_OP__damage_o([token:5]) is true.

Reasoner Results
o___OP_WRD_OP__viru__OP_conc_OP___o([token:1]) is true.
o___OP_WRD_OP__adware__OP_conc_OP___o([token:1]) is true.
o___OP_WRD_OP__application__OP_conc_OP___o([token:1]) is true.
o___OP_WRD_OP__problem__OP_conc_OP___o([token:5]) is true.
o___OP_WRD_OP__lot__OP_conc_OP___o([token:5]) is true.
o___OP_WRD_OP__antiviru__OP_conc_OP___o([token:1]) is true.
o___OP_WRD_OP__parasite__OP_conc_OP___o([token:1]) is true.
o___OP_WRD_OP__virus__OP_conc_OP___o([token:1]) is true.
o___OP_WRD_OP__number__OP_conc_OP___o([token:5]) is true.
o___OP_WRD_OP__ad__OP_conc_OP___o([token:5]) is true.
o___OP_WRD_OP__harm__OP_conc_OP___o([token:5]) is true.
o___OP_WRD_OP__error__OP_conc_OP___o([token:5]) is true.
o___OP_WRD_OP__issue__OP_conc_OP___o([token:5]) is true.

Output Results In Natural Language

spyware [is a kind of || is || is similar with] **viru**.

viru cause lot.

spyware [is a kind of || is || is similar with] **adware**.

adware cause lot.

spyware [is a kind of || is || is similar with] **application**.

application cause lot.

spyware [is a kind of || is || is similar with] **antiviru**.

antiviru cause lot.

spyware [is a kind of || is || is similar with] **parasite**.

parasite cause lot.

spyware [is a kind of || is || is similar with] **virus**.

virus cause lot.

lot [is a kind of || is || is similar with] **problem**.

spyware cause **problem**.

lot [is a kind of || is || is similar with] **number**.

spyware cause **number**.

lot [is a kind of || is || is similar with] **ad**.

spyware cause **ad**.

lot [is a kind of || is || is similar with] **harm**.

spyware cause **harm**.

lot [is a kind of || is || is similar with] **error**.

spyware cause **error**.

lot [is a kind of || is || is similar with] **issue**.

spyware cause **issue**.

damage [is a kind of || is || is similar with] **problem**.

spyware cause **problem**.

damage [is a kind of || is || is similar with] **lot**.

spyware cause **lot**.

damage [is a kind of || is || is similar with] **number**.

spyware cause **number**.

damage [is a kind of || is || is similar with] **ad**.

spyware cause **ad**.

damage [is a kind of || is || is similar with] **harm**.

spyware cause **harm**.

damage [is a kind of || is || is similar with] **error**.

spyware cause **error**.

damage [is a kind of || is || is similar with] **issue**.

spyware cause **issue**.

4.3 *Γενικά σχόλια για τα πειράματα*

Κατά τη διάρκεια και των δύο φάσεων των πειραμάτων, καταλήξαμε στα εξής συμπεράσματα:

Με το κατέβασμα 40000 αρχείων πετύχαμε ασυγκρίτως καλύτερη εξαγωγή συμπερασμάτων κοινής λογικής, παρά με τα 1000 αρχεία. Αυτό οφείλεται και σε ποσοτικούς αλλά και σε ποιοτικούς λόγους.

- Ποσοτικούς γιατί, έχοντας στη διάθεσή μας περισσότερη πληροφορία, μορφώνουμε με περισσότερη γνώση τη μηχανή μας.
- Ποιοτικούς γιατί, με 40000 αρχεία, ο Web Crawler έχει περισσότερες πιθανότητες να εντοπίσει πλούσιες σε περιεχόμενο ιστοσελίδες, μέσα από τον αλγόριθμο A*.

Με τις 40000 αρχεία είχαμε περισσότερες πιθανότητες να δημιουργήσουμε πλούσια κατηγορήματα, μέσω του Scene Constructor.

- Κατά την Α΄ φάση πειραμάτων, τα μόνα σύνολα κατηγορημάτων που επέστρεψαν καλά αποτελέσματα (όχι όμως και τόσο ποιοτικά), ήταν τα V5, V9 και V10, ενώ κατά τη Β΄ φάση, πήραμε ασυγκρίτως ποιοτικότερα κατηγορήματα μέσα από τα V5, V7, V9, V10, V12, V13 (και V10n που υπήρχε μόνο στη Β΄ φάση).

Με τα σύνολα των κατηγορημάτων V2, V3 και V4, τα οποία αφορούν κατηγορήματα που δημιουργήθηκαν με πρόσθεση υπερωνύμων (υπενθυμίζουμε την επιλογή select_all, η οποία αφορούσε τη λήψη όλων των υπερωνύμων που αφορούσαν την κάθε λέξη από το λεξικό WordNet), δεν πήραμε καλά αποτελέσματα.

- Αυτό θα μπορούσε να είχε σχέση με την ύπαρξη τόσης πολλής πληροφορίας στα αρχεία, που στο τέλος σύγχυζαν ή και υπερφόρτωναν τη διαδικασία της μάθησης. Μοιάζει με ένα παιδί, το οποίο προσπαθείς να μορφώσεις, δίνοντάς του αστραπιαία πάρα πολλές πληροφορίες που μοιάζουν πάρα πολύ μεταξύ τους (υπερώνυμα) και που δεν ξέρει ποια να χρησιμοποιήσει κάθε φορά για την εξαγωγή συμπερασμάτων κοινής λογικής.

Οι σημασιολογικοί κανόνες, που χρησιμοποιήσαμε, παράγουν πολύ καλά κατηγορήματα, αλλά δε φαίνεται αναγκαίος ο 13^{ος}.

- Από τα πειράματα, φάνηκε ότι το V10n, το οποίο περιείχε κατηγορήματα που δημιουργήθηκαν χωρίς τη χρήση του 13^{ου} σημασιολογικού κανόνα, ήταν εξίσου πλούσιο με το V10 που δημιουργήθηκε λαμβάνοντας υπόψη και αυτόν τον κανόνα. Άρα, φαίνεται ότι ο συσχετισμός ουσιαστικών απευθείας μέσω ρημάτων για τη δημιουργία δυαδικών κατηγορημάτων, δεν είναι αναγκαίος, γιατί καλύπτεται από τους υπόλοιπους κανόνες.

Με 40000 αρχεία, αρχίζουν να βελτιώνονται τα V12 και V13 που δημιουργήθηκαν χωρίς lemmatization και χωρίς χρήση υπερωνύμων.

- Χαρακτηριστική παρατήρηση είναι το γεγονός ότι, ενώ με τα 1000 αρχεία στα V12 και V13 δεν πήραμε τίποτα (υπενθυμίζουμε ότι αυτά τα δύο περιείχαν κατηγορήματα στα οποία δεν εφαρμόστηκαν κανόνες *lemmatization*), με τις 40000 πετύχαμε την εξαγωγή συμπερασμάτων κοινής λογικής, της ίδιας

κατηγορίας με τα υπόλοιπα σύνολα κατηγορημάτων. Η εφαρμογή κανόνων *lemmatization*, όταν έχουμε λίγα αρχεία παρέχει καλύτερα συμπεράσματα, κόβοντας όμως, χρήσιμη πληροφορία (π.χ. πληροφορία που αφορά τον πληθυντικό μιας λέξης και όχι τον ενικό). Με περισσότερα αρχεία όμως η ανάγκη για *lemmatization* παύει να υπάρχει και εξάγονται έτσι, καλύτερα συμπεράσματα κοινής λογικής.

Δεν υπάρχει άμεση σχέση μεταξύ τρόπου τροποποίησης της επερώτησης του χρήστη, μέσω της Έξυπνης Μηχανής (ρύθμιση Scene Constructor) και του τρόπου, με τον οποίο δημιουργήθηκαν τα κατηγορήματα μάθησης (ρύθμιση Scene Constructor).

- Όπως προαναφέραμε, μέσω της Έξυπνης Μηχανής, ο χρήστης μπορεί να τροποποιεί το ερώτημα πριν δοθεί στο Reasoner. Επίσης, υπενθυμίζουμε πως τα σύνολα κατηγορημάτων που δημιουργήσαμε (V2, V3 κτλ.), είχαν παραχθεί με συγκεκριμένες ρυθμίσεις στο Scene Constructor. Για παράδειγμα, θα περίμενε κανείς πως ο Reasoner θα εξήγαγε καλύτερους συλλογισμούς για το V7, αν τροποποιείτο το ερώτημα του χρήστη με την αντίστοιχη ρύθμιση της Έξυπνης Μηχανής (S11). Όμως, μέσα από τα πειράματα, δεν παρατηρήθηκε κάτι τέτοιο.

5

Συμπεράσματα και Μελλοντική Δουλειά

Τι μπορούμε να αλλάξουμε στο μέλλον;

Σε αυτό το κεφάλαιο παρουσιάζονται τα τελικά συμπεράσματα στα οποία καταλήξαμε, σε σχέση με την υλοποίηση, αλλά και τη χρησιμότητα της Έξυπνης Μηχανής. Επίσης προτείνονται τροποποιήσεις που μπορούν να γίνουν στο μέλλον και θα μπορούσαν να βελτιώσουν λογισμικά τα οποία αναπτύξαμε και έχουν σχέση με τη λειτουργικότητα της Έξυπνης Μηχανής.

Τα αποτελέσματα που διαφαίνονται μέσα από τα πειράματα, είναι ενθαρρυντικά και ίσως βοηθητικά για μελλοντικές ερευνητικές εργασίες. Δε θα ισχυριστούμε ότι δημιουργήσαμε την τέλεια μηχανή εξαγωγής συμπερασμάτων κοινής λογικής, αλλά μπορούμε να πούμε πως δημιουργήσαμε κάτι, το οποίο, αν και αρχικά έμοιαζε αβέβαιο, στο τέλος έδινε καλά αποτελέσματα.

Σε σχέση με τον αρχικό στόχο που θέσαμε ως προς την εξαγωγή συμπερασμάτων κοινής λογικής, λέμε πως επιτεύχθηκε σε αρκετό βαθμό. Επίσης, οι στόχοι που τέθηκαν γύρω από τον τρόπο αλληλεπίδρασης των χρηστών με την Έξυπνη Μηχανή, ικανοποιήθηκαν σε αρκετό βαθμό, αφού η μηχανή απαντούσε σε επερωτήσεις των χρηστών σε σύντομο χρονικό διάστημα, παρουσιάζοντας τις απαντήσεις σε μορφή κοντινή προς τη φυσική γλώσσα.

Αν αναλογισθεί κανείς τα δύο χρόνια που αφιερώσαμε για τη μελέτη, την ανάπτυξη, τη χρήση των προγραμμάτων επεξεργασίας των πληροφοριών και γενικότερα, όλη τη πιο πάνω διαδικασία να εφαρμόζεται και να υλοποιείται με σχεδόν συμβατικά μηχανήματα, τότε μπορεί να καταλάβει τι μπορεί να γίνει, αν ασχοληθούν μελλοντικοί ερευνητές περισσότερα χρόνια και με καλύτερα και γρηγορότερα μηχανήματα.

Μελλοντικά, σε γενικές γραμμές, μπορούν να γίνουν οι πιο κάτω βελτιώσεις-αναβαθμίσεις:

***Τροποποίηση-αναβάθμιση* Web Crawler**

Σε σχέση με την ταχύτητα**,** θα πρέπει να γίνει γρηγορότερο το πρόγραμμα του Web Crawler, ώστε να εντοπίζει τις πληροφορίες μέσα σε λιγότερο χρόνο (χρειάστηκαν σχεδόν πέντε ημέρες για εντοπισμό 40000 αρχείων).

- Μια αρχική ιδέα είναι να μη μετρά λέξεις για την αξιολόγηση μιας ιστοσελίδας, αλλά να βρεθεί άλλος τρόπος αξιολόγησης του περιεχομένου της, ο οποίος θα είναι πολύ πιο γρήγορος (π.χ. να εντοπίζει λέξεις με έντονα και πλάγια γράμματα, όπως κάνει και η μηχανή AltaVista).
- Μια άλλη εισήγηση είναι να βρεθεί τρόπος, έτσι ώστε να τρέχει ο αλγόριθμος A* παράλληλα.
- Σε περιπτώσεις που θα θέλαμε να εντοπίζονται πληροφορίες για διάφορα θέματα ταυτόχρονα, θα μπορούσε να τροποποιηθεί το πρόγραμμα, έτσι ώστε να τρέχουν παράλληλα ανεξάρτητα στιγμιότυπα που το καθένα με την ολοκλήρωσή του, θα οδηγούσε σε ξεχωριστή αλληλουχία των προγραμμάτων της αλυσίδας:
  (Downloader->Cleaner->Splitter->Stanford_Parser->Scene_Constructor).

Σε σχέση με την τεχνική crc-32, θα πρέπει να τροποποιηθεί, γιατί πρόκειται για πολύ ευαίσθητη τεχνική. Παρατηρήθηκαν περιπτώσεις που, ενώ το περιεχόμενο ιστοσελίδων ήταν το ίδιο, δινόταν σε κάθε μια διαφορετικός δεκαεξαδικός κωδικός, λόγω του ότι διέφεραν π.χ. κατά ένα γράμμα. Μπορεί να εφαρμοστεί μια καινούργια τεχνική, η οποία θα εντοπίζει και θα αξιολογεί τα κείμενα, ανάλογα με την ποιότητα και το νόημα του περιεχομένου τους και όχι μόνο με τα κοινά *byte* τους.

Stanford Parser

Η διαδικασία *parsing*, μέσα από το Stanford Parser, να γίνει γρηγορότερη, μέσα από τη χρήση μηχανημάτων με περισσότερη κύρια μνήμη, για να μπορούν να δίνονται στον *parser* απευθείας παράγραφοι και όχι μόνο προτάσεις. Σε αντίθετη περίπτωση, ίσως να χρειαστεί να γίνει και αναζήτηση άλλων *parsers* που να είναι λιγότερο απαιτητικοί σε κύρια μνήμη.

NLP Extractor

Η εξαγωγή των κανόνων να γίνεται σε πλήρως φυσική γλώσσα και όχι σε μερική, όπως συλλογίζεται ο μέσος άνθρωπος την ώρα που διαβάζει μια εφημερίδα, ένα περιοδικό ή ένα ηλεκτρονικό μέσο. Μια ιδέα είναι, πριν την παρουσίαση των αποτελεσμάτων του NLP Extractor, να φιλτράρεται μια βάση δεδομένων που θα περιέχει παραδείγματα φυσικών προτάσεων. Μέσα από τα παραδείγματα φυσικών προτάσεων θα τροποποιούνται τα αρχικά αποτελέσματα του NLP Extractor, για να μοιάζουν περισσότερο σε φυσικές προτάσεις, πριν την παρουσίασή τους προς τους χρήστες από την Έξυπνη Μηχανή.

6 ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία εκπονήθηκε υπό την επίβλεψη του καθηγητή Δρ. Λοΐζου Μιχαήλ. Θα ήθελα να τον ευχαριστήσω για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα, αλλά και για τη συμπαράσταση και καθοδήγησή του.

Επίσης, θα ήθελα να ευχαριστήσω τον τέως διευθυντή του Γυμνασίου Κοκκινοχωρίων Πάνου Ιωάννου κ. Ευάγγελο Κωνσταντίνου, αλλά και την τέως διευθύντρια του Εσπερινού Λυκείου Κοκκινοχωρίων κα Μαρία Ζάρου, για τη συγκατάθεσή τους στη διεξαγωγή έρευνας υπό μορφής ερωτηματολογίου στους καθηγητές και τους μαθητές των σχολείων τους.

7 ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Charniak, E. 2000. A Maximum Entropy Inspired Parser, Brown University, Providence, RI, In Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

[2] Etzioni, O.; Carafella, M.J.; Downey, D.; Popescu, A.M.; Shaked, T.; Soderland, S.; Weld, D.S.; Yates, A. 2005. Unsupervised Named-Entity Extraction From the Web: An Experimental Study, Artificial Intelligence 165(1):91-134.

[3] Fundel, K.; Kuffner, R.; Zimmer, R. 2006. RelEx—Relation Extraction Using Dependency Parse Trees, Advance Access Publication, Associate Editor: Satoru Miyano.

[4] Hart, P.E.; Nilsson, N.J.; Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics SSC4 4 (2), 100–107.

[5] Klein, D.; Manning, C. 2003. Accurate Unlexicalized Parsing, In Proceedings of the 41st Meeting of the Association for Computational Linguistics, 423-430.

[6] Lafferty, J.; Sleatro, D.; Temperley, D. 1992. Grammatical Trigrams: A Probabilistic Model of LInk Grammar, Carnegie Mellon University Pittsburgh, PA, USA.

[7] Lenat, D.B. 1995. CYC: A Large-Scale Investment in Knowledge Infrastructure, CACM38(11), 33-38.

[8] Littlestone, N. 1987. Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm, Machine Learning, 285-318.

[9] Malecha, G. 2009. The Role of Teaching Material in Knowledge Infusion, Manuscript.

[10] Marneffe, M.C.; Manning, C. 2008. Stanford Typed Dependencies Representation, In Cooling 2008 Workshop on Gross-framework and Cross-domain Parser Evaluation.

[11] McCarthy, J. 1960. Programs with Common Sense, Technical Report, Stanford University, Cambridge, MA, USA.

[12] Michael, L.; Valiant, L. 2008. A First Experimental Demonstration of Massive Knowledge Infusion, In Proceedings of KR' 2008, 378-389.

[13] Miller, G.; Beckwith, R.; Fellbaum, C.; Gross, D.; Miller, K. 1990. WordNet: An Online Lexical Database, International Journal of Lexicography, 3, 4, 235-244.

[14] Muggleton, S.H. 1991. Inductive Logic Programming, New Generation Computing 8(4), 295-318.

[15] Norvig P.; Russell, S. 2003. Artificial Intelligence: A Modern Aproach, Pearson Education, Second Edition.

[16] Peterson, W.W.; Brown, D.T. 1961. Cyclic Codes for Error Detection, Proceedings of the Institute of Radio Engineers 49, 228.

[17] Punyakanok, V.; Roth, D.; Yih, W. 2008. The Importance of Syntactic Parsing and Inference in Semantic Role Labeling, MIT Press Cambridge, MA, USA.

[18] Stork, D.G. 1999. The Open Mind Initiative, IEEE Expert Systems and their Applications 14(3), 16-20.

[19] Tsuruoka, Y.; Tsujii, J. 2005. Chunk Parsing Revisited**,** In Proceedings of the 9th International Workshop on Parsing Technologies, 133-140.

[20] Valiant, L. 2000. Robust Logics, Artificial Intelligence Journal, 117, 231-253.

[21] Valiant, L. Knowledge Infusion, In Proceedings of the National Conference on Artificial Intelligence, Volume 21, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

8 Παράρτημα

Υπάρχουν βοηθητικά στοιχεία στα οποία μπορεί να ανατρέξει οποιοσδήποτε χρειάζεται επιπλέον πληροφορίες;

Στο παράρτημα παρουσιάζουμε το ερωτηματολόγιο που χρησιμοποιήσαμε, καθώς και την ανάλυσή του, όπως και απόσπασμα από τις δυαδικές συσχετίσεις από το εγχειρίδιο του Stanford Parser. Επίσης, παρουσιάζουμε πληροφορίες σε σχέση με τεχνικά ζητήματα που αφορούν τους Η.Υ. που χρησιμοποιήσαμε για τη διεξαγωγή των πειραμάτων. Τέλος, παρουσιάζουμε το εγχειρίδιο χρήσης της Έξυπνης Μηχανής.

☐☐☐

ΕΡΩΤΗΜΑΤΟΛΟΓΙΟ

Αγαπητοί συνάδελφοι, μαθητές & μαθήτριες,

Η παρούσα έρευνα θα προσπαθήσει να ανιχνεύσει τις απαιτήσεις σας γύρω από την αναζήτηση πληροφοριών στο Διαδίκτυο. Πρόκειται για έναν έλεγχο των γνώσεων και των απαιτήσεών σας γύρω από την αναζήτηση πληροφοριών, με απώτερο σκοπό τη δημιουργία μιας Έξυπνης Μηχανής αναζήτησης, η οποία θα σας παρέχει πλεονεκτήματα που μέχρι τώρα δεν είχατε.

Στο ερωτηματολόγιο υπάρχουν ερωτήσεις κλειστού και ανοικτού τύπου με διευκρινιστικά σχόλια. Υπάρχει συγκεκριμένος χώρος για τις απαντήσεις σας και καλείστε να απαντάτε βάζοντας √ μέσα στο σωστό τετραγωνάκι και όχι έξω από αυτό. ☐

Α΄ Μέρος-Γενικές Πληροφορίες

1. Σημειώστε το φύλο σας:
   - Άρρεν ☐
   - Θήλυ ☐

2. Σημειώστε την ηλικία σας:
   - 10 – 19 ☐
   - 20 – 29 ☐
   - 30 – 39 ☐
   - 40 – 49 ☐
   - 50 – 60 ☐

3. Σημειώστε το μορφωτικό σας επίπεδο:
   - Απόφοιτος Δημοτικής Εκπαίδευσης ☐
   - Απόφοιτος Γυμνασίου ☐
   - Απόφοιτος Λυκείου ☐
   - Απόφοιτος Πανεπιστημίου ☐
   - Κάτοχος μεταπτυχιακού Master ☐
   - Κάτοχος Διδακτορικού τίτλου ☐

4. Πόσο συχνά χρησιμοποιείτε το Διαδίκτυο στους εξής χώρους;

	Καθημερινά	4-6 φορές την εβδομάδα	1-3 φορές την εβδομάδα	2-3 φορές το μήνα	Λιγότερο
Στο σχολείο	☐	☐	☐	☐	☐
Στο σπίτι	☐	☐	☐	☐	☐
Στη δουλειά	☐	☐	☐	☐	☐
Κάπου αλλού	☐	☐	☐	☐	☐

5. Πόσο συχνά χρησιμοποιείτε τις μηχανές αναζήτησης (Search Engines);
   - 1 με 5 ώρες/εβδομάδα ☐
   - 6 με 10 ώρες/εβδομάδα ☐
   - 11 με 20 ώρες/εβδομάδα ☐
   - Περισσότερο από 20 ώρες/εβδομάδα ☐

6. Βαθμολογήστε τις μηχανές αναζήτησης τοποθετώντας τον αριθμό 6 σε αυτήν που χρησιμοποιείτε περισσότερο και τον αριθμό 1 σε αυτήν που χρησιμοποιείτε λιγότερο. Εάν δε γνωρίζετε μια μηχανή Αναζήτησης, επιλέξτε τον αριθμό 0.

   (Σημείωση: Οι μηχανές αναζήτησης να βαθμολογηθούν μοναδικά, δηλαδή η κάθε μια θα πρέπει να έχει διαφορετική βαθμολογία.)

	Βαθμολογία
• Google	0 [1 2 3 4 5 6]
• Yahoo	0 [1 2 3 4 5 6]
• Bing	0 [1 2 3 4 5 6]
• Ask	0 [1 2 3 4 5 6]
• AltaVista	0 [1 2 3 4 5 6]
• Lycos	0 [1 2 3 4 5 6]

7. Γιατί προτιμάτε τη μηχανή αναζήτησης που έχετε βαθμολογήσει με τον αριθμό 6; (Σημειώστε όσα κουτάκια θέλετε).

   - Είναι πολύ φιλική. ☐
   - Εντοπίζει τις πληροφορίες που θέλω. ☐
   - Είναι πολύ γρήγορη. ☐
   - Δεν περιέχει περιττά στοιχεία. ☐
   - Ανοίγει πιο γρήγορα στον Η.Υ. σε σχέση με τις άλλες. ☐
   - Δεν περιέχει διαφημίσεις. ☐
   - Επειδή τη θυμάμαι πιο εύκολα λόγω του ονόματός της. ☐

8. Κυρίως χρησιμοποιείτε τις μηχανές αναζήτησης για:

   - Αναζήτηση διευθύνσεων άλλων ιστοσελίδων. ☐
   - Αναζήτηση γενικότερων πληροφοριών. ☐
   - Για απάντηση σε συγκεκριμένα ερωτήματα. ☐
   - Άλλο:…… ☐

9. Πόσο ευχαριστημένοι είστε από τις απαντήσεις που παίρνετε;

   - Πάρα πολύ ☐
   - Πολύ ☐
   - Λίγο ☐
   - Καθόλου ☐

Αν στο προηγούμενο βήμα επιλέξατε *πάρα πολύ* ή *πολύ* πηγαίνετε στο Β’ Μέρος.

10. Οι απαντήσεις που παίρνετε: (μπορείτε να επιλέξετε όσες επιλογές θέλετε)

- Είναι λανθασμένες τις περισσότερες φορές. ☐
- Δεν απαντούν ευθέως σε αυτό που θα θέλατε. ☐
- Είναι πολύ γενικές. ☐
- Σας σπαταλάνε χρόνο στο να εντοπίσετε αυτό που πραγματικά σας ενδιαφέρει. ☐
- Άλλο:…… ☐

Β΄ Μέρος-Τρόποι αλληλεπίδρασης

Θα μπορούσαμε να πούμε ότι στις υπάρχουσες κλασσικές μηχανές αναζήτησης δίνουμε μια ερώτηση ή μια λέξη κλειδί και παίρνουμε σαν αποτέλεσμα τα "βιβλία" τα οποία περιέχουν την ερώτηση ή τη λέξη κλειδί και τα οποία ψάχνουμε με την ***ελπίδα*** να βρούμε την απάντηση στην ερώτησή μας (απάντηση η οποία τελικά μπορεί να μην περιέχεται καν στα "βιβλία" αυτά).

Στόχος είναι η δημιουργία μιας Έξυπνης Μηχανής αναζήτησης στην οποία δίνουμε μια ερώτηση ή μια λέξη κλειδί και η οποία εντοπίζει και επιστρέφει τα "βιβλία" που περιέχουν την απάντηση, ή και απευθείας την ίδια την απάντηση.

11. Σε ποιο ποσοστό της συνολικής χρήσης μηχανών αναζήτησης που κάνετε πιστεύετε ότι θα χρησιμοποιούσατε μια Έξυπνη Μηχανή αναζήτησης; Γιατί επιλέξατε το πιο κάτω ποσοστό;

- Δηλώστε το ποσοστό (από 0% μέχρι 100%):.....%
- Γιατί:.....
.....
.....
.....

12. Τι είδους αλληλεπίδραση (ερώτησης και απάντησης) θα θέλατε να έχετε με μια Έξυπνη Μηχανή αναζήτησης; (Πρέπει να επιλέξετε μόνο **μία** απάντηση).

- Κοινής Λογικής π.χ. ☐

Πού θα οδηγήσει η συνεχόμενη ρύπανση του πλανήτη;

Αποτελέσματα: 1. Η ρύπανση του πλανήτη είναι η κύρια αιτία για τις κλιματικές αλλαγές. 2. Αν συνεχιστεί η ρύπανση του πλανήτη, θα λιώσουν οι πάγοι και θα ανέβει η στάθμη της θάλασσας.

- Υπολογιστικής-Μαθηματικής φύσεως π.χ. ☐

Ποιο είναι το εμβαδό επιφάνειας και ο όγκος ενός κώνου όταν: Ύψος=20 Ακτίνα βάσης=10

Αποτέλεσμα: Εμβαδό επιφάνειας=1016.640738463052 Όγκος=2094.3951023931954

- Κάποιο άλλο είδος; ☐
  Παρακαλώ εξηγήστε ή δώστε ένα παράδειγμα:

……
……
.....

13. Δεδομένης της δυσκολίας απάντησης σε φυσική γλώσσα από ένα μηχάνημα, πόσο αποδεκτή βρίσκετε την κάθε μία από τις πιο κάτω μορφές απαντήσεων; (Τοποθετήστε σε κάθε κουτάκι την επιλογή σας).

Επιλογές:
0=καθόλου
1= λίγο
2= πολύ
3= πάρα πολύ

Ερώτηση: "Τι γίνεται σε περιπτώσεις απαγωγής;"

-Πρόταση-φράση ☐

Ο δολοφόνος απαγάγει το θύμα και ζητά λύτρα.

-Σύνολο λέξεων ☐

δολοφόνος απαγάγει θύμα ζητά λύτρα

-Κατηγορήματα ☐

δολοφόνος 1 θύμα 2 λύτρα 3 1 απαγάγει 2 1 ζητά 3

-Επιστροφή μίας σελίδας η οποία περιέχει την απάντηση ☐
(ίσως με κάποιες highlighted λέξεις)

Το θύμα πήγε εκδρομή την περασμένη εβδομάδα και ενώ περπατούσε ανέμελο στο πεζοδρόμιο ο δολοφόνος, ερχόμενος με αυτοκίνητο, το απήγαγε. Στη συνέχεια ζήτησε λύτρα.

14. Για κάθε έναν από τους τρόπους της προηγούμενης ερώτησης, πόσο χρόνο θα είστε διατεθειμένοι να περιμένετε για να πάρετε την απάντηση στην ερώτησή σας; Λάβετε υπόψη ότι στις υπάρχουσες κλασσικές μηχανές αναζήτησης, μετά την επιστροφή των αποτελεσμάτων, χρειάζεται να ξοδέψετε επιπλέον χρόνο για να εντοπίσετε την απάντηση στα "βιβλία" που σας έχουν επιστραφεί.

- Πρόταση-φράση [Θα περίμενα ☐ λεπτά και ☐ δευτερόλεπτα]
- Σύνολο λέξεων [Θα περίμενα ☐ λεπτά και ☐ δευτερόλεπτα]
- Κατηγορήματα [Θα περίμενα ☐ λεπτά και ☐ δευτερόλεπτα]
- Επιστροφή σελίδας [Θα περίμενα ☐ λεπτά και ☐ δευτερόλεπτα]

ΤΕΛΟΣ ΕΡΩΤΗΜΑΤΟΛΟΓΙΟΥ
(Σας Ευχαριστούμε)

Ανάλυση Ερωτηματολογίου

Για καλύτερα αποτελέσματα, πριν από την προσπάθεια δημιουργίας της Έξυπνης Μηχανής, δημιουργήθηκε ένα ερωτηματολόγιο που ως στόχο είχε τον προσδιορισμό των αναγκών των χρηστών γύρω από την αναζήτηση των πληροφοριών στον Παγκόσμιο Ιστό (www). Έπρεπε να εντοπιστεί, αν υπήρχε ανάγκη δημιουργίας μιας Έξυπνης Μηχανής αναζήτησης συμπερασμάτων κοινής λογικής και, γενικά, έπρεπε να εντοπιστούν χρήσιμες πληροφορίες σε σχέση με τις εμπειρίες των χρηστών όσον αφορά τη χρήση υπαρχουσών μηχανών αναζήτησης.

Γενικότερα, μέσα από το ερωτηματολόγιο, έπρεπε να ανακαλύψουμε και να προσδιορίσουμε τα εξής:

- Πόσο ευχαριστημένοι είναι οι χρήστες γύρω από την αναζήτηση πληροφοριών, χρησιμοποιώντας υπάρχουσες μηχανές αναζήτησης.
- Αν θεωρούν τις σημερινές συνθήκες αναζήτησης ιδανικές και πλήρεις ή αν θα ήθελαν μια πιο στοχευόμενη αναζήτηση, μέσα από ένα νέο τρόπο αλληλεπίδρασης ερώτησης/απάντησης.

Όλα τα πιο πάνω, μαζί με αρκετά άλλα επιμέρους ζητήματα, τέθηκαν υπό μορφή ερωτηματολογίου σε 231 άτομα ηλικίας 12 μέχρι 60 χρόνων. Το ερωτηματολόγιο συνολικά αποτελείται από δεκατέσσερις ερωτήσεις (κλειστού και ανοικτού τύπου) και τέθηκε, συγκεκριμένα, στις εξής ομάδες πληθυσμού:

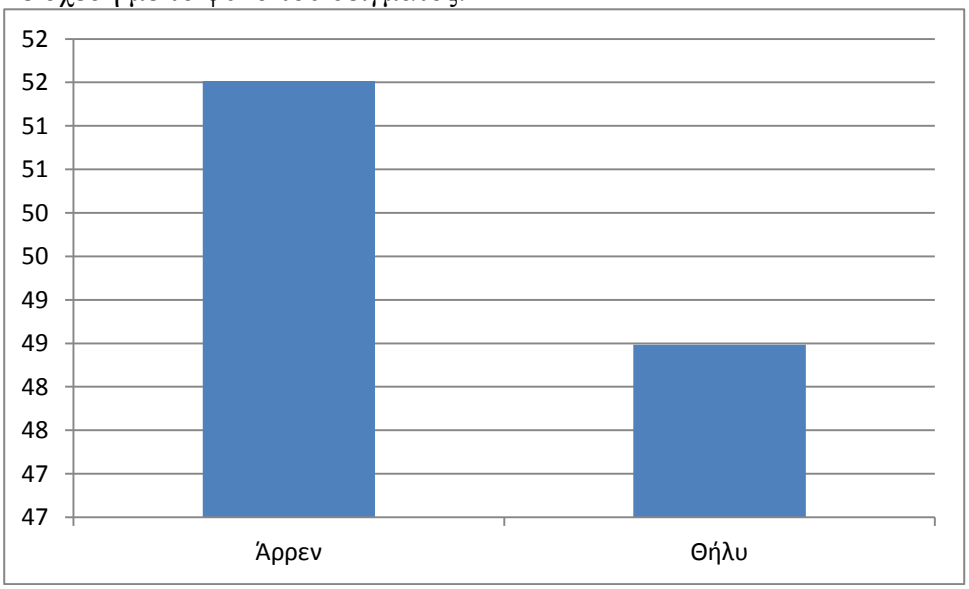
- Σε καθηγητές και μαθητές του Γυμνασίου Κοκκινοχωρίων Πάνου Ιωάννου.
- Σε καθηγητές και μαθητές του Εσπερινού Λυκείου Κοκκινοχωρίων.

Πιο κάτω φαίνονται αναλυτικά τα αποτελέσματα της έρευνας:

Αποτελέσματα Ανάλυσης Ερωτηματολογίων (%)

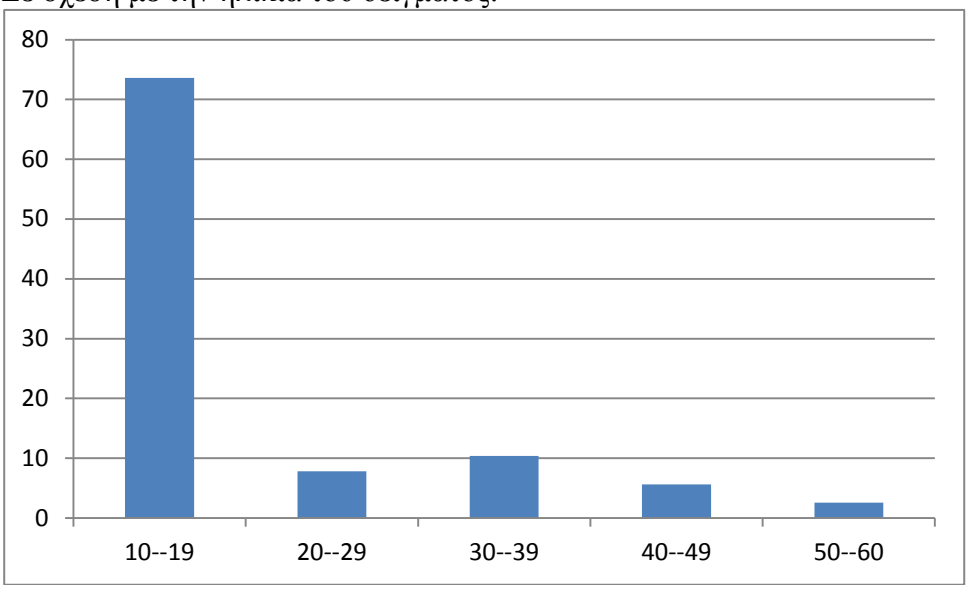
Δείγμα: 231 άτομα ηλικίας 12-60 χρόνων.
Περίοδος έρευνας: 10/2009-12/2009.

Γραφική Παράσταση 01
Σε σχέση με το φύλο του δείγματος:



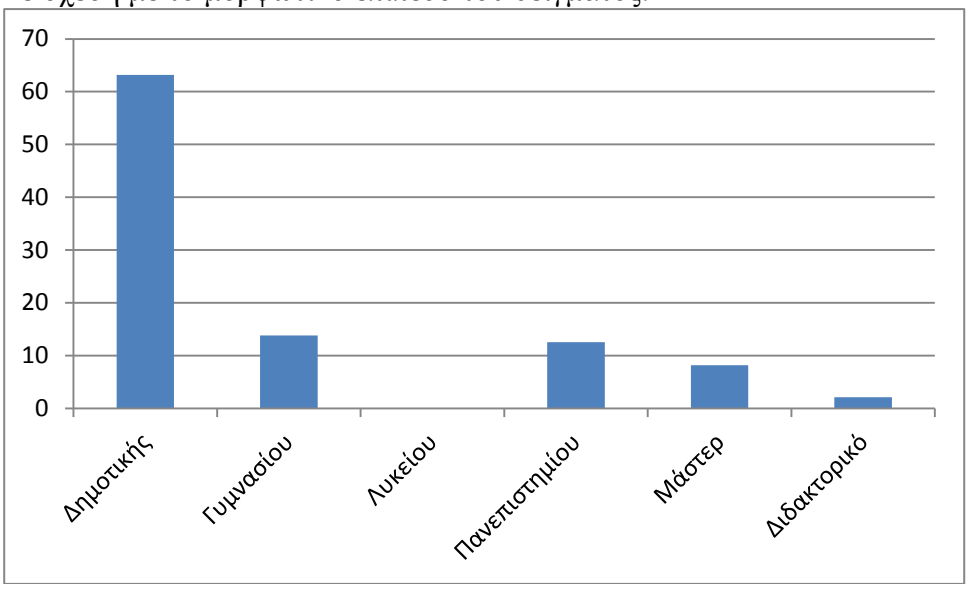
Το ερωτηματολόγιο τέθηκε σε ένα δείγμα συνόλου 231 ατόμων, εκ των οποίων οι 119 ήταν άντρες και οι 112 γυναίκες.

Γραφική Παράσταση 02
Σε σχέση με την ηλικία του δείγματος:



Βλέποντας τη γραφική παράσταση, καταλαβαίνουμε ότι η πλειοψηφία του δείγματος αφορά άτομα ηλικίας 10-19 χρόνων και αυτό συμβαίνει, γιατί το ερωτηματολόγιο, όπως προαναφέραμε, τέθηκε κυρίως σε μαθητές Γυμνασίου και σε μαθητές Εσπερινού σχολείου. Οι μαθητές του Εσπερινού (ηλικίες από 17 μέχρι 60 χρόνων) είναι πολύ λιγότεροι από τους μαθητές του Γυμνασίου και, για αυτόν το λόγο, φαίνεται η συντριπτική διαφορά στην πιο πάνω γραφική παράσταση.

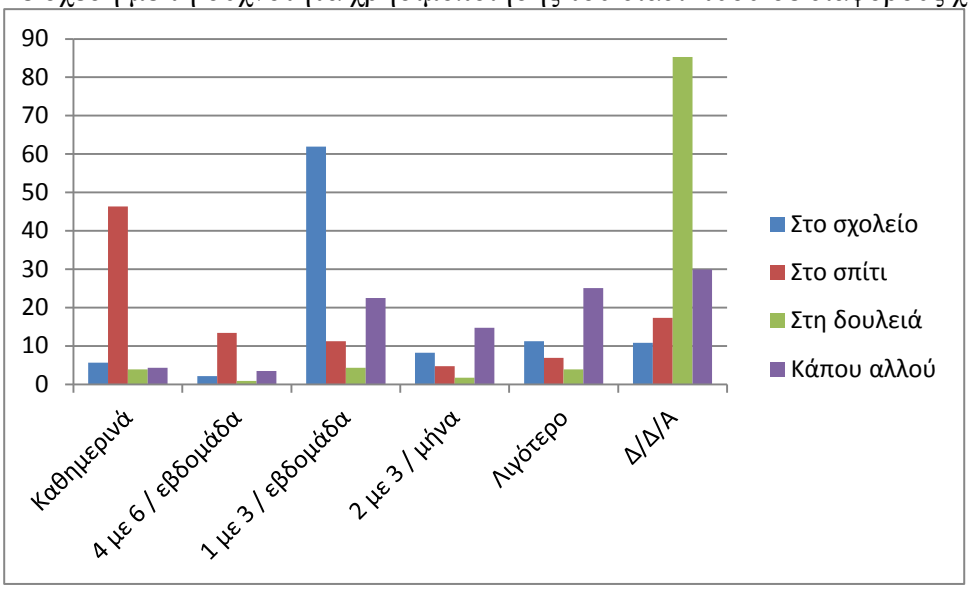
Γραφική Παράσταση 03
Σε σχέση με το μορφωτικό επίπεδο του δείγματος:



Σε σχέση με το μορφωτικό επίπεδο, βλέπουμε ότι η ομάδα που κυριαρχεί είναι αυτή της Δημοτικής εκπαίδευσης. Αυτό συμβαίνει, γιατί ο κύριος όγκος του δείγματος ήταν μαθητές μεταξύ 10-19 χρόνων και οι οποίοι δεν είχαν αποφοιτήσει ακόμα από το Γυμνάσιο (συμπεριλαμβάνονται και μαθητές του Εσπερινού σχολείου). Επίσης, βλέπουμε ότι οι απόφοιτοι Γυμνασίου (αυτό το δείγμα βρέθηκε στο Εσπερινό σχολείο) έχουν, περίπου, ίσο αριθμό με τους απόφοιτους Πανεπιστημίου

(συναδέλφους που εργάζονται στα πιο πάνω σχολεία). Αξίζει να αναφέρουμε ότι σχεδόν οι μισοί από τους συναδέλφους είναι κάτοχοι μεταπτυχιακού διπλώματος Master, με ένα χαμηλότερο ποσοστό να είναι κάτοχοι διδακτορικού διπλώματος.

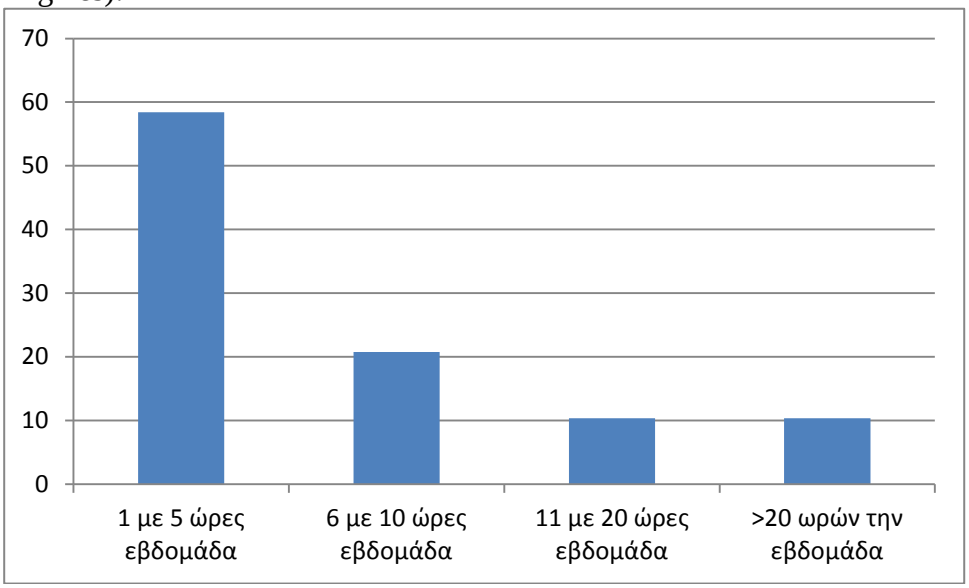
Γραφική Παράσταση 04
Σε σχέση με τη συχνότητα χρησιμοποίησης του διαδικτύου σε διάφορους χώρους:



Σε αυτήν την ερώτηση, βλέπουμε ότι στο σχολείο, οι περισσότεροι χρησιμοποιούν το διαδίκτυο μία με τρεις φορές την εβδομάδα και, στο σπίτι, οι περισσότεροι χρησιμοποιούν το διαδίκτυο καθημερινά. Στη στήλη Δ/Δ/Α* το ποσοστό των ατόμων που χρησιμοποιεί το διαδίκτυο στη δουλειά είναι το μεγαλύτερο, γιατί η πλειοψηφία του δείγματός μας αφορά άτομα, τα οποία δεν εργάζονται.

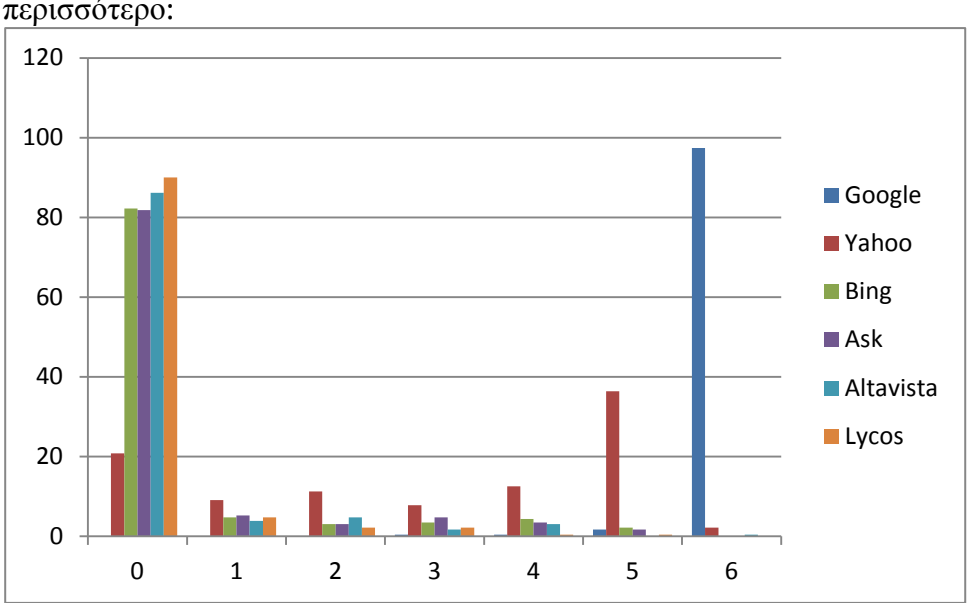
*Η στήλη Δ/Δ/Α είναι μια στήλη την οποία εμείς δημιουργήσαμε, για να δείξουμε το ποσοστό των ατόμων που δεν επέλεξε τις συγκεκριμένες επιλογές.

Γραφική Παράσταση 05
Σε σχέση με τη συχνότητα χρησιμοποίησης των μηχανών αναζήτησης (Search Engines):



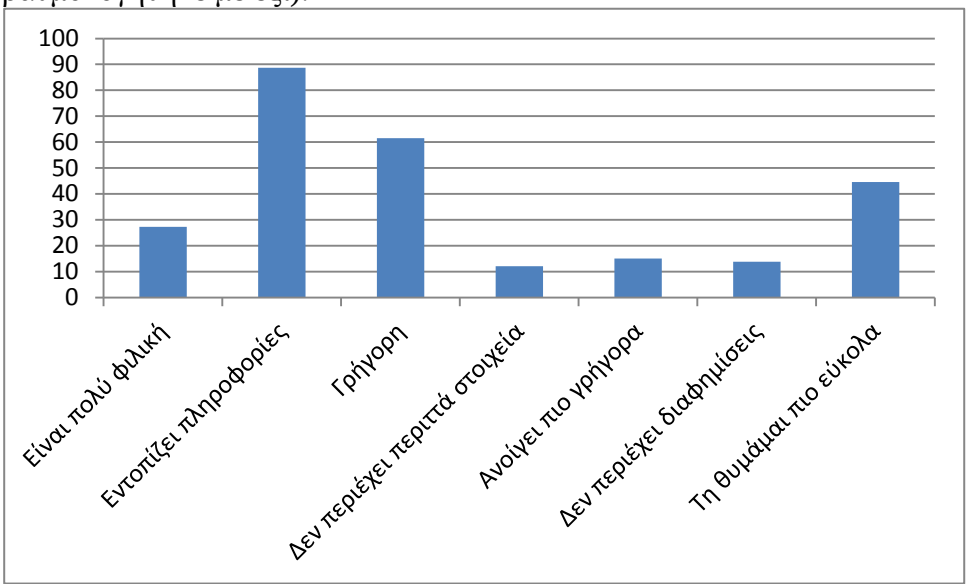
Σε σχέση με τη συχνότητα χρησιμοποίησης των μηχανών αναζήτησης, βλέπουμε ότι οι περισσότεροι χρησιμοποιούν τις μηχανές αναζήτησης μια με πέντε φορές την εβδομάδα, με ένα μικρότερο μέρος του δείγματος να χρησιμοποιεί τις μηχανές έξι με δέκα ώρες την εβδομάδα.

Γραφική Παράσταση 06
Βαθμολόγηση μηχανών αναζήτησης με βάση αυτήν που χρησιμοποιείται περισσότερο:



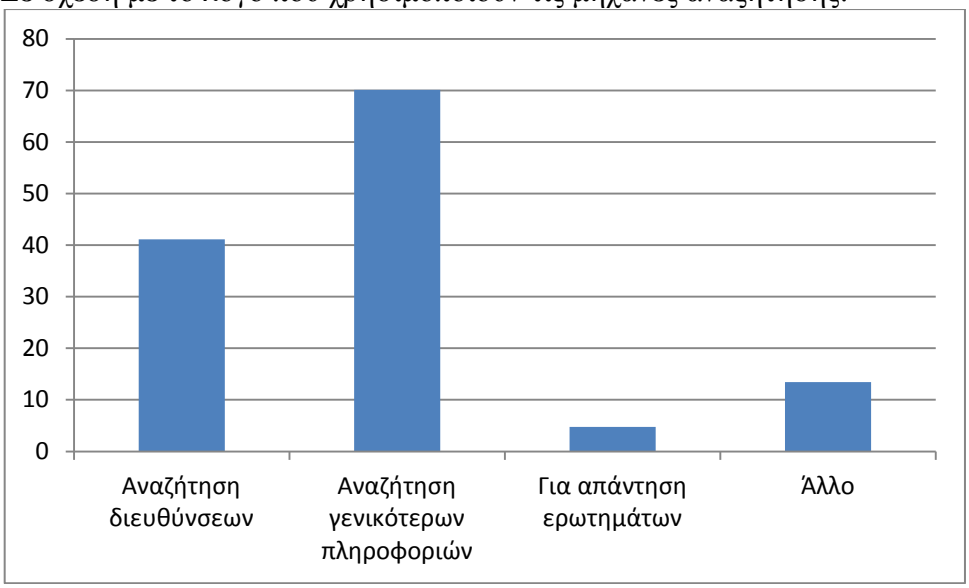
Σε αυτήν την ερώτηση, οι μαθητές έπρεπε να βαθμολογήσουν μοναδικά τις μηχανές αναζήτησης, ξεκινώντας με το βαθμό έξι (σε αυτήν που χρησιμοποιούν περισσότερο) και καταλήγοντας κλιμακωτά στο βαθμό ένα (σε αυτήν που χρησιμοποιούν λιγότερο). Αν δεν γνώριζαν μια μηχανή αναζήτησης, θα έπρεπε να επιλέγουν τον αριθμό μηδέν. Βλέπουμε ότι οι περισσότεροι χρησιμοποιούν τη μηχανή αναζήτησης Google (σχεδόν όλο το δείγμα) και, ακολούθως, τη μηχανή αναζήτησης Yahoo. Επίσης, βλέπουμε ότι τις άλλες μηχανές αναζήτησης, οι περισσότεροι δεν τις γνωρίζουν.

Γραφική Παράσταση 07
Για το λόγο προτίμησης της προηγούμενης μηχανής αναζήτησης (που βαθμολογήθηκε με έξι):



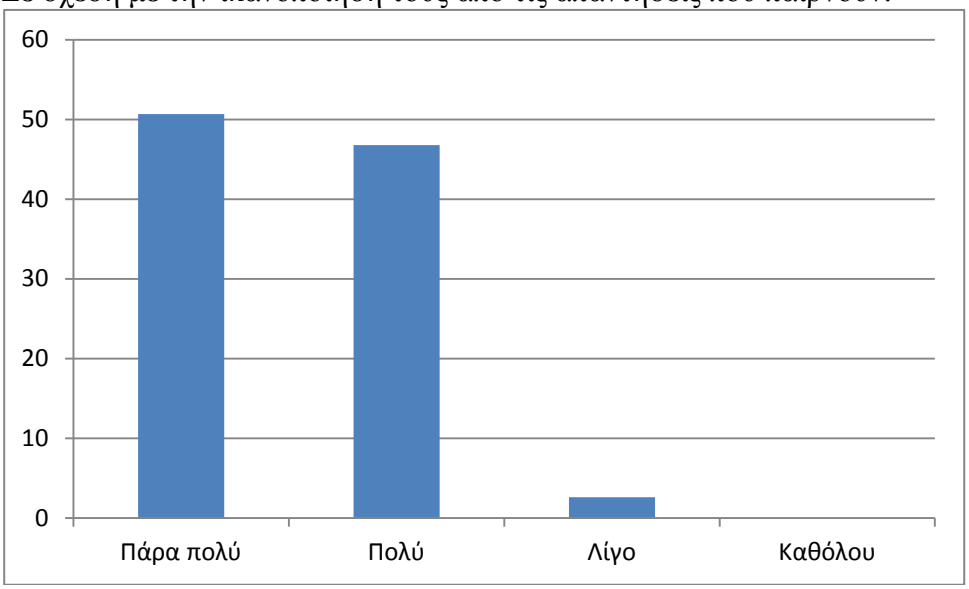
Φαίνεται ότι οι χρήστες που βαθμολόγησαν με βαθμό έξι την Google την προτιμάνε, γιατί, κυρίως, εντοπίζει τις πληροφορίες που ζητάνε και γιατί είναι γρήγορη στην ανταπόκριση. Ένα ψηλό ποσοστό έδωσε ως λόγο το εύκολο όνομα (Google), γιατί το θυμάται πιο εύκολα.

Γραφική Παράσταση 08
Σε σχέση με το λόγο που χρησιμοποιούν τις μηχανές αναζήτησης:



Σε αυτή τη γραφική παράσταση, βλέπουμε ότι χρησιμοποιούν τις μηχανές αναζήτησης για την αναζήτηση γενικότερων πληροφοριών και για την αναζήτηση άλλων διευθύνσεων ιστοσελίδων. Στις απαντήσεις τους, αυτοί που επέλεξαν την επιλογή Άλλο, ανέφεραν λόγους όπως η ψυχαγωγία, οι εργασίες και η εύρεση και αγορά προϊόντων.
(Βλέπουμε ότι, σχεδόν όλο το δείγμα μας δεν επέλεξε την επιλογή για απάντηση ερωτημάτων).

Γραφική Παράσταση 09
Σε σχέση με την ικανοποίησή τους από τις απαντήσεις που παίρνουν:

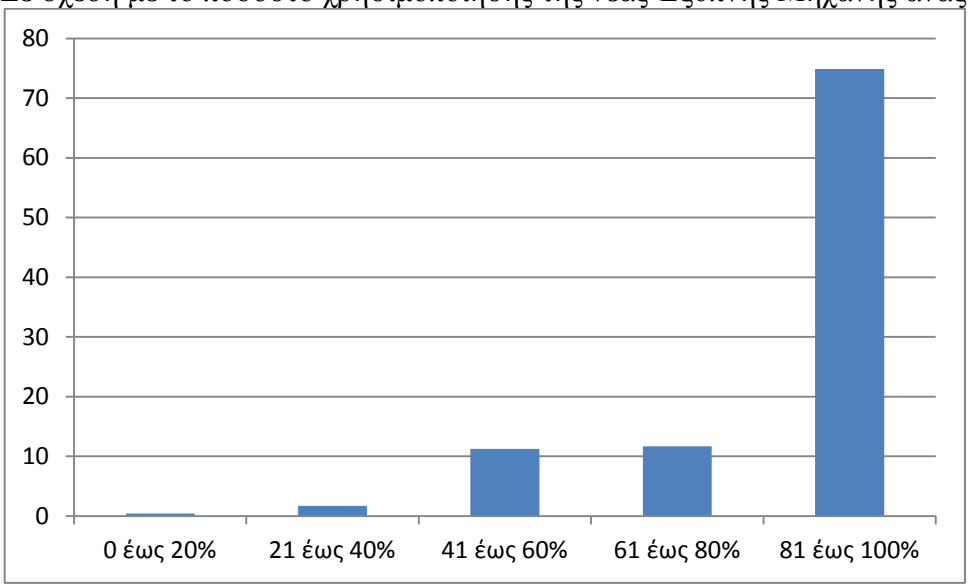


Γενικά, βλέπουμε ότι το δείγμα μας είναι πολύ ευχαριστημένο, σε σχέση με τις απαντήσεις που παίρνει από τις μηχανές αναζήτησης, σε ποσοστό σχεδόν 100%.

Γραφική Παράσταση 10
Από αυτούς που απάντησαν Λίγο ή Καθόλου στην προηγούμενη ερώτηση:

Ένα πολύ χαμηλό ποσοστό του δείγματος που επέλεξε στην προηγούμενη ερώτηση Λίγο, έδωσε ως λόγο προτίμησης το γεγονός ότι τους παίρνει αρκετό χρόνο ο εντοπισμός της απάντησης που ψάχνουν.

Γραφική Παράσταση 11
Σε σχέση με το ποσοστό χρησιμοποίησης της νέας Έξυπνης Μηχανής αναζήτησης:



Σε ερώτησή μας, αν υπήρχε μια Έξυπνη Μηχανή που θα τους απαντούσε ευθέως σε αυτό που έψαχναν, πόσο χρόνο θα της αφιέρωναν από το συνολικό χρόνο που αφιέρωναν, όσον αφορά τη χρήση των άλλων μηχανών αναζήτησης, απάντησαν τα εξής: Ότι 75% θα χρησιμοποιούσε την Έξυπνη Μηχανή, σε συνολικό ποσοστό 81% με 100% του συνολικού τους χρόνου.

Σε σημείωσή τους, ανά ηλικία, ανέφεραν τους εξής λόγους:

10-19

- Εξοικονομούμε χρόνο και κόπο.
- Για ψάξιμο συγκεκριμένων πληροφοριών.
- Θα ήταν πιο εξυπηρετική.
- Βρίσκουμε κάτι που θέλουμε πολύ γρήγορα.
- Είναι γρήγορη και σου δίνει ακριβείς απαντήσεις.
- Θα ήταν πιο εύκολη στην αναζήτηση.
- Εξοικονομεί ενέργεια και κάνει καλό στον πλανήτη.
- Θα έχει πολλές πληροφορίες.
- Δε θα καθόμαστε να διαβάζουμε άσχετες πληροφορίες, αλλά θα μπαίνουμε κατευθείαν στο θέμα.
- Δε θα εκνευριζόμαστε και δε θα απογοητευόμαστε μέχρι να βρούμε τη σωστή απάντηση.
- Αν και θα είναι πιο εύκολη και αποτελεσματική η αναζήτηση πληροφοριών, οι περισσότεροι έχουμε ήδη συνηθίσει στις παλιές μεθόδους και θα είναι δύσκολο να προσαρμοστούμε γρήγορα στις καινούργιες.
- Δε θα θέλουμε να ανοίγουμε εγκυκλοπαίδεια για να βρούμε την απάντηση.
- Είναι ακριβώς αυτό που ζητούσα.
- Επειδή είναι η πρώτη μηχανή που σου προσφέρει αυτή τη δυνατότητα, να σου βγάζει, δηλαδή, γρήγορα και απευθείας τις απαντήσεις που θέλεις.

20-29

- Γατί έτσι θα κέρδιζα χρόνο.
- Για ένα 30% δε θα την χρησιμοποιούσαν, γιατί θα περιορίζονταν οι δυνατότητες ανεύρεσης μιας πολύπλευρης απάντησης.

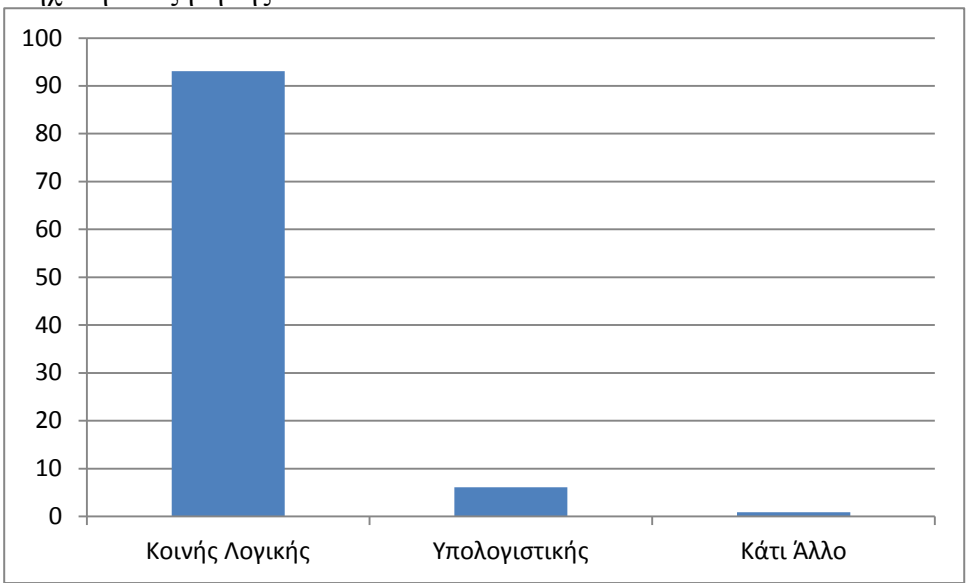
30-39

- Αν με πείσει ότι συμπεριφέρεται και με εξυπηρετεί καλύτερα από τις άλλες, δε χρειάζεται να χρησιμοποιώ τις άλλες.
- Θα είναι πολύ βοηθητική στην απάντηση που αναζητώ κάθε φορά. Θα σπαταλώ λιγότερο χρόνο για αναζήτηση. Πιστεύω ότι το ψάξιμο θα είναι πιο ποιοτικό και αποτελεσματικό.
- Για εξοικονόμηση χρόνου και κόπου.
- Δεν πιστεύω να υπάρχει Έξυπνη Μηχανή 100%.

50-60

- Θα εξοικονομούσα χρόνο, αφού θα έβρισκα ευκολότερα ό,τι αναζητούσα.
- Θα είναι ακριβής και ουσιαστική.

Γραφική Παράσταση 12
Σε σχέση με το είδος της αλληλεπίδρασης που θα ήθελαν να έχουν με την Έξυπνη Μηχανή αναζήτησης:

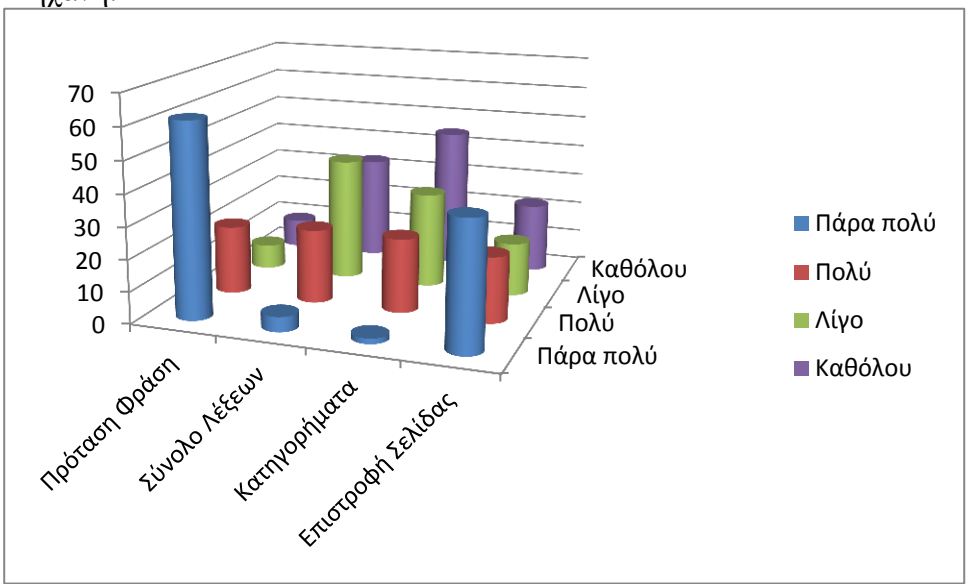


Σε ερώτηση σχετικά με το είδος της αλληλεπίδρασης που θα ήθελαν να έχουν με τη νέα μηχανή, οι περισσότεροι (ποσοστό 93%) δήλωσαν ότι προτιμούν μια Έξυπνη Μηχανή κοινής λογικής.

Αυτοί που επέλεξαν Κάτι άλλο έθεσαν την εξής απαίτηση:

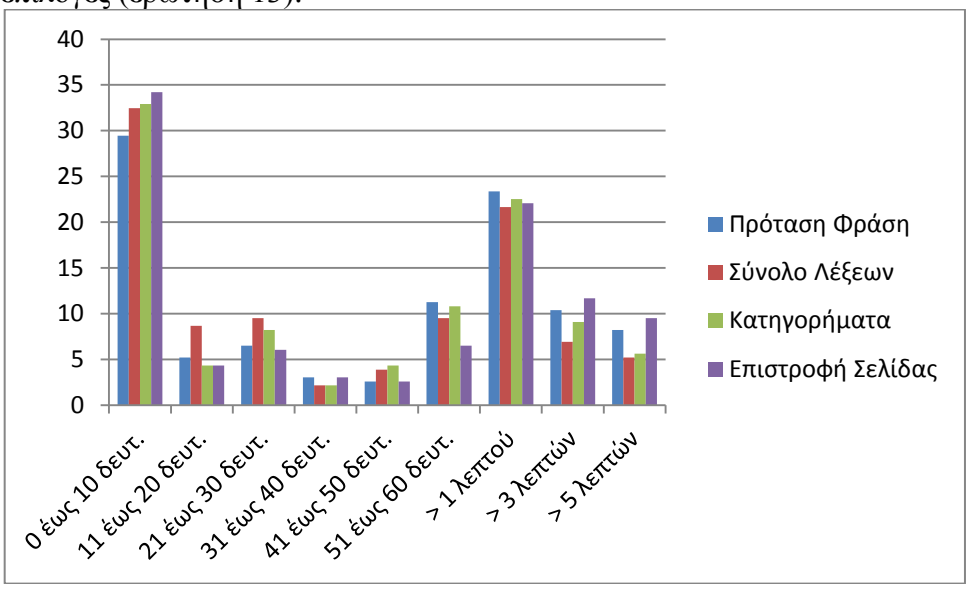
- Να απαντά σε όλες τις ερωτήσεις των μαθημάτων μας.

Γραφική Παράσταση 13
Σε σχέση με τις μορφές απαντήσεων που θα ήθελαν να παίρνουν από τη νέα Έξυπνη Μηχανή:



Σε ερώτησή μας σχετικά με τις μορφές απάντησης που θα ήθελαν από τη νέα μηχανή, οι περισσότεροι επέλεξαν την πρόταση-φράση.

Γραφική Παράσταση 14
Σε σχέση με το χρόνο αναμονής για την απάντηση όσον αφορά τις προηγούμενες επιλογές (ερώτηση 13):



Όπως βλέπουμε, οι περισσότεροι επέλεξαν χρόνο από μηδέν έως δέκα δευτερόλεπτα, για όλες τις μορφές απάντησης.

Γενικό σχόλιο
Από τα πιο πάνω υπήρχε επιβεβαίωση της ανάγκης για δημιουργία μιας Έξυπνης Μηχανής συμπερασμάτων κοινής λογικής, η οποία θα απαντούσε σε επερωτήσεις των χρηστών, με μια μορφή κοντινή προς τη φυσική γλώσσα, σε σύντομο χρονικό διάστημα.

Εγχειρίδιο επεξήγησης δυαδικών συσχετίσεων Stanford Parser

Πιο κάτω παρουσιάζουμε ένα μικρό μέρος από το εγχειρίδιο του *parser*, μέσα από το οποίο φαίνεται ο τρόπος δημιουργίας δυαδικών συσχετίσεων.

abbrev	abbrev(Corporation, ABC)	«The Australian Broadcasting Corporation (ABC)».
acomp	acomp(looks, beautiful)	«She looks very beautiful».
advcl: adverbial clause modifer	advcl(tell, know)	An adverbial clause modifier of a VP is a clause modifying the verb (temporal clause, consequence, conditional clause, etc.). «If you know who did it, you should tell the teacher».
advmod: adverbial modifier	advmod(modified, genetically)	Modify the meaning of the word. «Genetically modified food».
agent: agent	agent(killed, police)	An agent is the complement of a passive verb which is introduced by the preposition «by» and does the action. «The man has been killed by the police».
amod: adjectival modifer	amod(meat, red)	Serves to modify the meaning of the NP. «Sam eats red meat».
appos: appositional modifer	appos(Sam, brother) appos(Bill, cousin)	An appositional modifier of an NP is an NP immediately to the right of the first NP that serves to define or modify that NP. «Bill (John's cousin)». «Sam, my brother».
attr: attributive (γνώρισμα)	attr (is, What)	Copular verb such as «to be», «to seem», «to appear». «What is that?».
aux: auxiliary	aux(died, has) aux(died, has) aux(leave, should)	Modal auxiliary, «be» and «have» in a composed tense. «Reagan has died». «He should leave».
auxpass: passive auxiliary	auxpass(killed, been)	Non-main verb of the clause which contains the passive information. «Kennedy has been killed».
cc: coordination	cc(big, and)	An element of a conjunct and the coordinating conjunction word of the conjunct. «Bill is big and honest».
ccomp: clausal complement	ccomp(says, like) ccomp(certain, did) ccomp(says, like) ccomp(certain, did)	A clausal complement of a VP or an ADJP is a clause with internal subject which functions like an object of the verb or of the adjective; a clausal complement of a clause is the clausal complement of the VP or of the ADJP which is the predicate of that clause. «He says that you like to swim». «I am certain that he did it».
complm: complementizer	complm(like, that)	Is the word introducing it. It will be the subordinating conjunction «that» or «whether». «He says that you like to swim».
conj : conjunct	conj(big, honest)	A conjunct is the relation between two elements connected by a coordinating conjunction, such as «and», «or», etc. «Bill is big and honest».
csubj: clausal subject	csubj (makes, said) csubj (true, said)	i.e. The subject is itself a clause. «What she said makes sense». «What she said is not true».
csubjpass: clausal passive subject	Csubjpass(suspected, lied)	A clausal passive subject is a clausal syntactic subject of a passive clause. «That she lied was suspected by everyone».
det: determiner	det(man, the) det(book, which)	A determiner is the relation between the head of an NP and its determiner. «The man is here». «Which book do you prefer?».
dobj : direct object	dobj (gave, raise) dobj (win, lottery)	The direct object of a VP is the noun phrase which is the (accusative) object of the verb. «She gave me a raise». «They win the lottery».
expl: expletive	expl(is, There)	This relation captures an existential «there». The main verb of the clause is

		the governor. «There is a ghost in the room».
infmod:	infmod(points, establish) infmod(anything, say)	An infnitive that serves to modify the meaning of the NP. «Points to establish are . . . .» «I don't have anything to say».
iobj : indirect object	iobj (gave, me)	The indirect object of a VP is the noun phrase which is the (dative) object of the verb. «She gave me a raise».
mark: marker	mark(attacked, after)	It will be a subordinating conjunction different from «that» or «whether»: e.g. «because», «when», «although», etc. «Forces engaged in fighting after insurgents attacked».
measure: measure-phrase modifier	measure(old, years) measure(long, feet)	The measure-phrase modifier is the relation between the head of an ADJP/ADVP and the head of a measure-phrase modifying the ADJP/ADVP. «The director is 65 years old». «6 feet long».
neg: negation modifier	neg(scientist, not) neg(drive, n't)	The negation modifier is the relation between a negation word and the word it modifies. «Bill is not a scientist». «Bill doesn't drive».
nn: noun compound modifier	nn(futures, oil) nn(futures, price)	A noun compound modifier of an NP is any noun that serves to modify the head noun. «Oil price futures».
nsubj : nominal subject	nsubj (defeated, Clinton) nsubj (cute, baby)	A nominal subject is a noun phrase which is the syntactic subject of a clause. «Clinton defeated Dole». «The baby is cute».
nsubjpass: passive nominal subject	nsubjpass(defeated, Dole)	A passive nominal subject is a noun phrase which is the syntactic subject of a passive clause. «Dole was defeated by Clinton».
numeric modifier	num(sheep, 3)	Modify the meaning of the NP. «Sam eats 3 sheep».
number: element of compound number	number($, billion)	An element of compound number is a part of a number phrase or currency amount. «I lost $ 3.2 billion».
parataxis: parataxis	parataxis(left, said)	Is a relation between the main verb of a clause and other sentential elements, such as a sentential parenthetical, a clause after a «:» or a «;». «The guy, John said, left early in the morning».
partmod: participial modifier	partmod(trufles, picked) partmod(shoot, demonstrating)	A participial modifier of an NP or VP is a participial verb form that serves to modify the meaning of the NP or VP. «Trufles picked during the spring are tasty». «Bill tried to shoot demonstrating his incompetence».
pcomp: prepositional complement	pcomp(on, are) pcomp(about, missing)	The prepositional complement of a preposition is the head of a clause following the preposition. «We have no information on whether users are at risk». «They heard about you missing classes».
pobj : object of a preposition	pobj (on, chair)	The object of a preposition is the head of a noun phrase following the preposition. «I sat on the chair».
poss: possession modifier	poss(clothes, Bill)	The possession modifier relation holds between the head of an NP and its possessive determiner, or a genitive 's complement. «Bill's clothes».
possessive: possessive modifier	possessive(John, 's)	The possessive modifier relation appears between the head of an NP and the genitive 's. «John’s clothes».
preconj : preconjunct	preconj (boys, both)	A preconjunct is the relation between the head of an NP and a word that is part of a conjunction,an puts emphasis on it (e.g., «either», «both», «neither»).

		«Both the boys and the girls are here».
predet: predeterminer	predet(boys, all)	A predeterminer is the relation between the head of an NP and a word that precedes and clarifies the use of the NP determiner. «All the boys are here».
prep/prepc: prepositional modifier	prep(cat, in) prep(saw, with) prep(responsible, for)	A prepositional modifier of a verb, adjective, or noun is any prepositional phrase that serves to modify the meaning of the verb, adjective, or noun. «I saw a cat in a hat». «I saw a cat with a telescope». «He is responsible for meals».
prt: phrasal verb particle	prt(shut, down)	The phrasal verb particle relation identifies a phrasal verb, and holds between the verb and its particle. «They shut down the station».
punct: punctuation	punct(Go, !)	This is used for any piece of punctuation in a clause. «Go home!».
purpcl : purpose clause modifier	purpcl(talked, secure)	A purpose clause modifier of a VP is a clause headed by «(in order) to» specifying a purpose. At present the system only recognizes ones that have «in order to». «He talked to him in order to secure the account».
quantmod: quantifier phrase modifier	quantmod(200, About)	A quantifier modifier is an element modifying the head of a QP constituent. «About 200 people came to the party».
rcmod: relative clause modifier	rcmod(man, love) rcmod(book,bought)	A relative clause modifier of an NP is a relative clause modifying the NP. The relation points from the head noun of the NP to the head of the relative clause, normally a verb. «I saw the man you love». «I saw the book which you bought».
ref : referent	ref (book, which)	A referent of the head of an NP is the relative word introducing the relative clause modifying the NP. «I saw the book which you bought».
rel : relative	rel (love, who) rel (love, wife)	relative of a relative clause is the head word of the WH-phrase introducing it. «I saw the man who you love». «I saw the man whose wife you love».
tmod: temporal modifier	tmod(swam, night)	A temporal modifier of a VP or an ADJP is any constituent that serves to modify the meaning of the VP or the ADJP by specifying a time; a temporal modifier of a clause is a temporal modifier of the VP which is the predicate of that clause. «Last night, I swam in the pool».
xcomp: open clausal complement	xcomp(like, swim) xcomp(ready, leave)	An open clausal complement (xcomp) of a VP or an ADJP is a clausal complement without its own subject, whose reference is determined by an external subject. The name xcomp is borrowed from Lexical-Functional Grammar. «He says that you like to swim». «I am ready to leave».
xsubj : controlling subject	xsubj (eat, Tom)	A controlling subject is the relation between the head of a open clausal complement (xcomp) and the external subject of that clause. «Tom likes to eat fish».

Χρήσιμες τεχνικές πληροφορίες για μελλοντική δουλειά

Σε αυτό το σημείο, αναφέρουμε χρήσιμα σχόλια, τα οποία έχουν σχέση με σημαντικά σημεία και τα οποία καλό είναι να προσέξουν μελλοντικοί ερευνητές, οι οποίοι θα ασχοληθούν με την ίδια ή με παρόμοια έρευνα. Συγκεκριμένα, θα αναφέρουμε πληροφορίες για ταχύτητες εκτέλεσης, αλλά και χρήσης πόρων, που έχουν σχέση με τα προγράμματα που χρησιμοποιήσαμε, αλλά και με τα μηχανήματα που είχαμε στη διάθεσή μας.

Διαθέσιμος εξοπλισμός

Για τις ανάγκες των πειραμάτων (40000 αρχεία) χρησιμοποιήσαμε δύο H.Y. με τα εξής χαρακτηριστικά:

H.Y. A	H.Y. B
-intel xeon® cpu e5507 2.27 GHZ (2 Quad processors) -8 GB RAM -64 bit Windows 7 professional	-Quad core amd opteron™ processor 2376 2.30GHz -4 GB Ram -Windows server 2003 (32 bit)

Μέσα από τα πειράματα, αντιμετωπίσαμε αρκετά προβλήματα, που είχαν σχέση με τους διαθέσιμους πόρους του συστήματος, γιατί, τις περισσότερες φορές, φτάναμε σε χρήση 100% *CPU & RAM*.

Το τρέξιμο των προγραμμάτων, μέσα από τη χρήση του διαχειριστή (Manager), χρειάζεται πολλούς πόρους μνήμης, αλλά και επεξεργαστή. Συγκεκριμένα, αναφέρουμε ενδεικτικά αυτό που παρατηρήσαμε μέσα από τη διαδικασία των πειραμάτων (η όλη διαδικασία ξεκίνησε στον H.Y. A).

Από την ημέρα που ολοκληρώθηκε ο εντοπισμός των αρχείων, μέσα από το Web Crawler (104 ώρες για 40000 αρχεία), ξεκίνησε η εκτέλεση των παράλληλων διαδικασιών. Μέσα από το Manager, δημιουργήσαμε 100 *buckets* (*folders*) που, στην ουσία, σήμαινε ότι θα έτρεχαν κάθε φορά 100 συνδυασμοί Downloader->Cleaner->Splitter->Stanford_Parser->Scene_Constructor για κάθε *bucket*. Η διαδικασία αυτή ξεκίνησε στις 8/4/11 μέχρι και τις 12/4/11. Λόγω της χρήσης πόρων που έφτανε στο 100% (*CPU & RAM*), δεν μπορούσαμε να ξέρουμε μέχρι πού βρισκόταν η όλη διαδικασία (χαρακτηριστικά, δεν μπορούσαμε να μετακινήσουμε ούτε το ποντίκι του υπολογιστή μας) και για αυτό σταματήσαμε την εκτέλεση των προγραμμάτων.

***Τα αποτελέσματα, μέχρι εκείνη τη στιγμή ήταν τα εξής*:**

-Ολοκληρώθηκαν μόνο 10 *buckets* από τα 100, όπου, μέσα από έλεγχο των αρχείων που έπρεπε να κατέβουν και να επεξεργαστούν, εντοπίσαμε ότι αφορούσαν μικρά σε μέγεθος αρχεία (μέγιστου μεγέθους 45 *KB* το καθένα).

-Σε όλα τα άλλα *buckets* είχε ξεκινήσει η διαδικασία του *parsing* (δηλαδή, είχαν ολοκληρωθεί οι εκτελέσεις των προγραμμάτων Downloader, Cleaner, Splitter). Συγκεκριμένα, σε όλα τα *buckets* είχαν γίνει *parsed* περίπου μέχρι και 200 αρχεία, που, στην τελική, έπρεπε να φτάσουν περίπου μέχρι και τα 30000. Συνεπώς, μπορεί κανείς να αναλογιστεί τον τελικό χρόνο ολοκλήρωσης, αν αφήναμε τη διαδικασία και δεν τη σταματούσαμε.

Λόγω των πιο πάνω προβλημάτων αποφασίσαμε να τρέχουμε τη διαδικασία *parsing* και στους δύο Η.Υ. (Α και Β). Παρατηρήσαμε ότι για 20 *buckets* χρειάζονταν, περίπου, 36-40 ώρες εκτέλεσης (χρήση 100% *CPU* και περίπου 50% *Ram*/4*GB*).

Ακολούθως, μετά την ολοκλήρωση του *parsing*, έπρεπε να τρέχουμε παράλληλα μέχρι και 100 στιγμιότυπα του Scene Constructor (για 100 buckets), με αποτέλεσμα χρήσης 100% *CPU*. Στη συνέχεια, προχωρήσαμε προς την παράλληλη εκτέλεση, από 20 μέχρι και 50 *buckets* κάθε φορά ανάλογα με το αν είχαμε ρυθμίσεις (Scene Constructor) για υπερώνυμα ή όχι (για υπερώνυμα είχαμε χρήση για κάθε στιγμιότυπο Scene Constructor μέχρι και 66 MB Ram).

Ενδεικτικά, αναφέρουμε κάποια αποτελέσματα (χρήσης πόρων στον Η.Υ. Α):

[υπερώνυμα, επιλεγμένοι όλοι οι σημασιολογικοί κανόνες δημιουργίας κατηγορημάτων, επιλογή όλων των υπερωνύμων]

start	20/4/11 8:20
finish	21/4/11 7:00
way	31-61-90
cpu	98%
ram	71%

[υπερώνυμα, μόνο ο κανόνας nsubj-dobj δημιουργίας κατηγορημάτων, επιλογή όλων των υπερωνύμων]

start	21/4/11 12:40
finish	21/4/11 22:00
way	50-50
cpu	85%
ram	60%

[υπερώνυμα, μόνο δυαδικοί κανόνες δημιουργίας κατηγορημάτων, επιλογή της πιο σπάνιας λέξης των υπερωνύμων]

start	22/4/11 23:30
finish	23/4/11 8:00
way	50-50
cpu	90%
ram	50%

[μόνο lemmatization, επιλεγμένοι όλοι οι σημασιολογικοί κανόνες δημιουργίας κατηγορημάτων]

start	18/4/11 09:00
finish	18/4/11 17:00
way	50-50
cpu	35%
ram	50%

[χωρίς lemmatization, επιλεγμένοι όλοι οι σημασιολογικοί κανόνες δημιουργίας κατηγορημάτων]

start	22/4/11 15:23
finish	22/4/11 23:20
way	50-50
cpu	25%

ram 50%

Πληροφορίες για το λογισμικό Learner

Μέσα από πειράματα με το Learner εντοπίσαμε ότι, για ένα *folder* με περίπου 600000 αρχεία, χρειάζονταν γύρω στις 33 ώρες, για να ολοκληρωθεί η διαδικασία μάθησης και η δημιουργία του *knowledge file*. Επειδή, κατά τα πειράματα, εντοπίστηκε πως ο Learner αξιοποιούσε (στο τρέξιμο ενός στιγμιότυπου) γύρω στο 10% *CPU* και γύρω στο 10% κύρια μνήμη (των 4 *GB*), αποφασίσαμε να τρέξουμε παράλληλα τη διαδικασία από εκεί και πέρα.

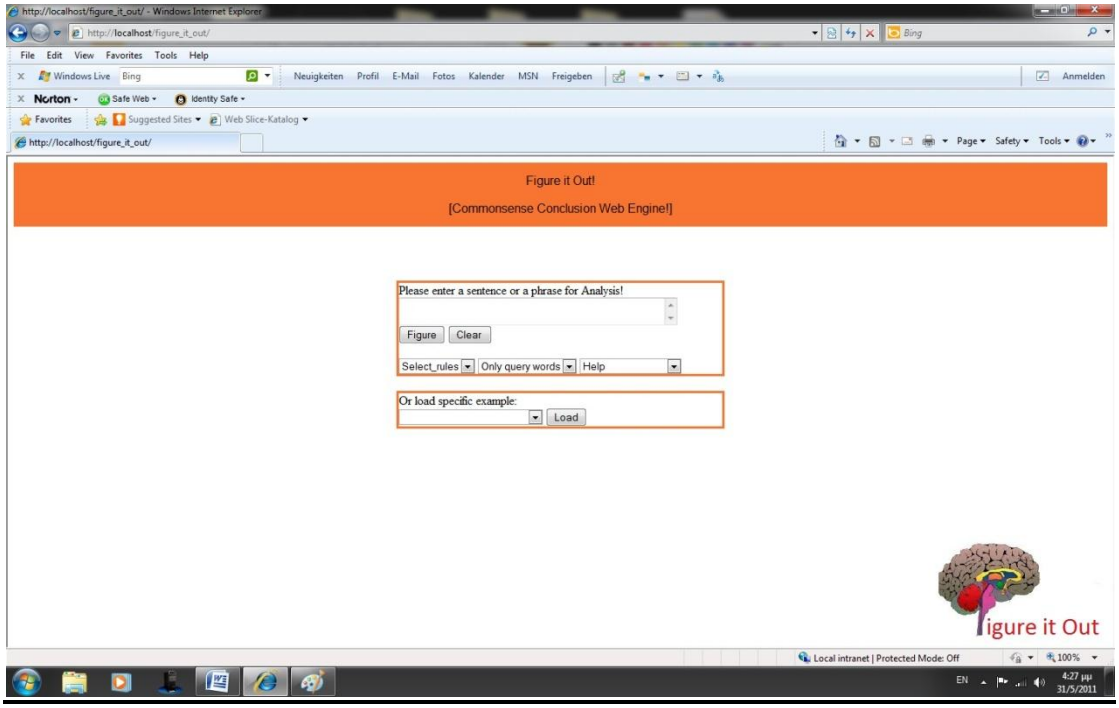
Ενδεικτικά αναφέρονται τα πιο κάτω:

- Στον Η.Υ. Β τρέξαμε 4 στιγμιότυπα του Learner για τα V2, V5, V9, V10 (υπενθυμίζουμε ότι το σύμβολο V υποδηλώνει σύνολα κατηγορημάτων) και, συγκεκριμένα, η διαδικασία ξεκίνησε στις 23/4/11, ώρα 18:24 και ολοκληρώθηκε ως εξής:
  - Το V10 (με 411656 αρχεία) στις 24/4/11, ώρα 22:00.
  - Τα V5 (με 673881 αρχεία) και το V9 (με 674063) στις 25/4/11, ώρα 20:00.
  - Το V2 (με 673881 αρχεία) στις 25/4/11, ώρα 23:00.
- Στον Η.Υ. Α τρέξαμε 5 στιγμιότυπα για τα V3, V7, V13, V12, V8 και η διαδικασία ξεκίνησε στις 24/4/11, ώρα 09:50 και ολοκληρώθηκε την ίδια ημέρα, στις 22:00.

Πιο πάνω, φαίνεται ο σημαντικός ρόλος που παίζει η ποιότητα του εξοπλισμού που χρησιμοποιείται. Εξαιτίας αυτού, η διαδικασία ολοκλήρωσης 5 συνόλων κατηγορημάτων (V) στον Η.Υ. Α ολοκληρώθηκε σε 12 ώρες, ενώ στον Η.Υ. Β η διαδικασία ολοκλήρωσης 5 συνόλων κατηγορημάτων χρειάστηκε περίπου 48 ώρες.

Εγχειρίδιο χρήσης Έξυπνης Μηχανής

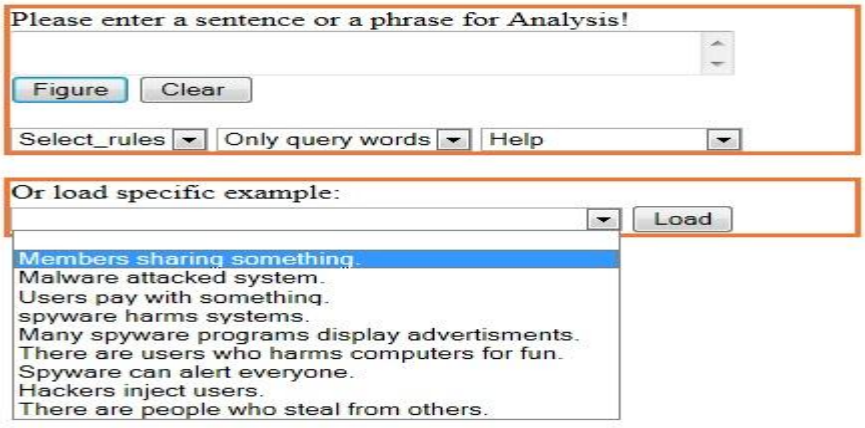
Την Έξυπνη Μηχανή μπορεί κανείς να τη βρει στη διεύθυνση http://82.116.211.98/figure_it_out και μπορεί να αποκτήσει πρόσβαση εισάγοντας για *user name* το **admin** και για *password* το **ouc**.



Σε σχέση με τον τρόπο αλληλεπίδρασης με την Έξυπνη Μηχανή, οι χρήστες έχουν δύο επιλογές:

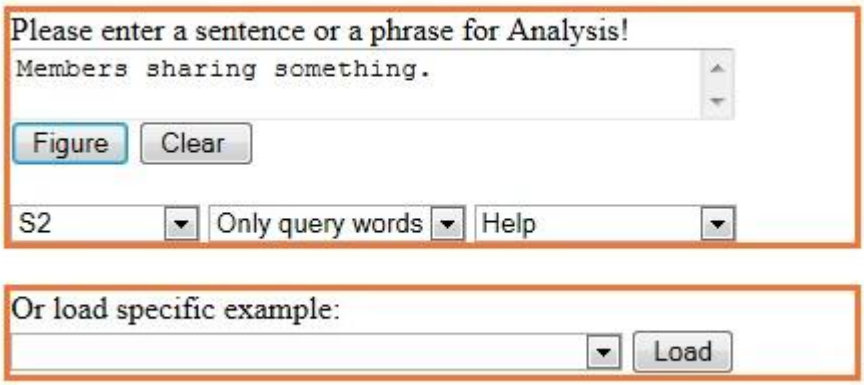
Επιλογή πρώτη [Γρήγορη εκκίνηση μέσω επιλογής παραδειγμάτων]
Μέσω αυτής της επιλογής, η Έξυπνη Μηχανή μας προσφέρει κάποια παραδείγματα, τα οποία μπορούμε να φορτώσουμε και να τρέξουμε εκείνη τη στιγμή, για να πάρουμε πίσω συμπεράσματα κοινής λογικής.

Βήμα 1
Επιλέγοντας το βέλος εμφανίζονται σε μας κάποιες προτάσεις και επιλέγουμε αυτήν της αρεσκείας μας (στην περίπτωση μας επιλέγουμε τυχαία την πρώτη).



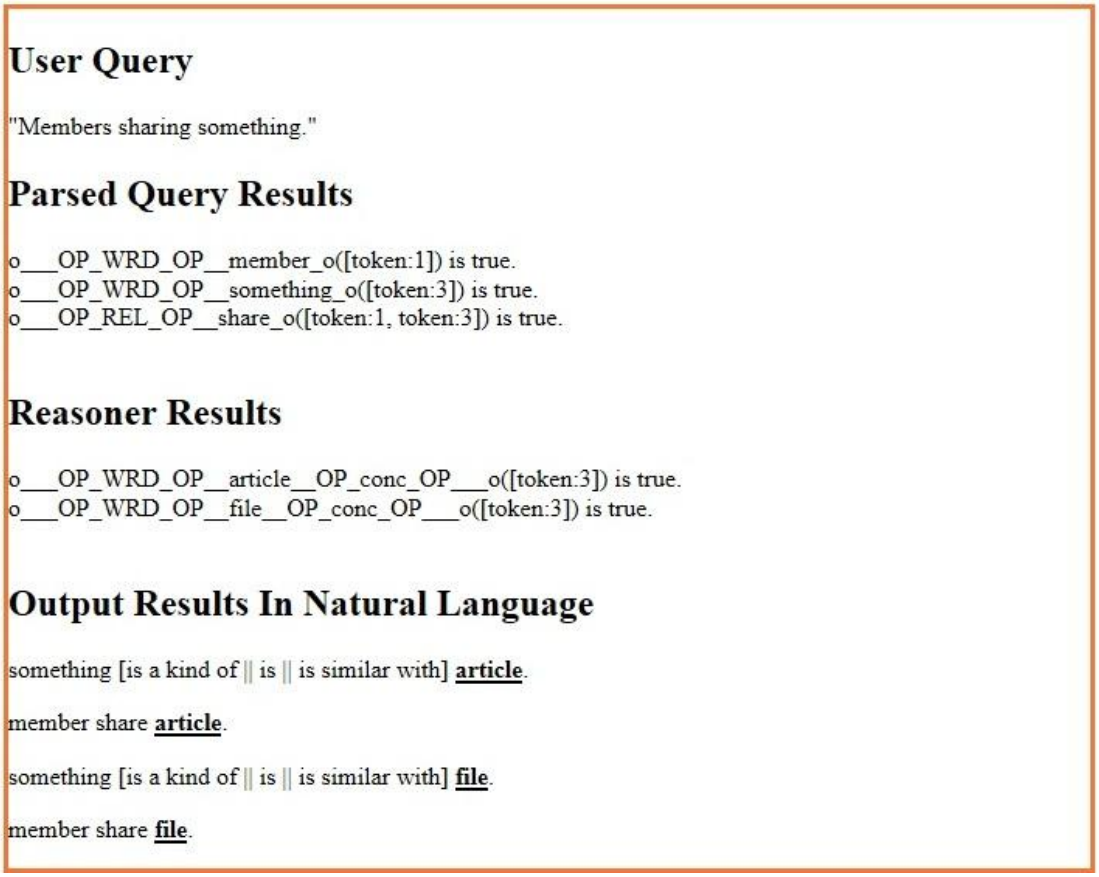
Βήμα 2

Επιλέγοντας το κουμπί *Load* φορτώνεται το παράδειγμα (αυτόματα επιλέγεται και κανόνας->, συγκεκριμένα στο *select_rules* επιλέγεται το *S2)*. Ο κάθε κανόνας έχει σχέση με τον τρόπο δημιουργίας των κατηγορημάτων και ειδικότερα με τον τρόπο εισαγωγής της επερώτησης του χρήστη στο Reasoner.



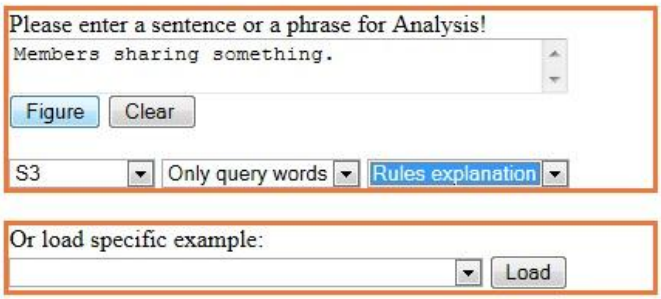
Βήμα 3

Σε αυτό το σημείο είμαστε έτοιμοι να ζητήσουμε από τη μηχανή να μας εξαγάγει συλλογισμούς για τη συγκεκριμένη πρόταση και έτσι επιλέγουμε το κουμπί *Figure*. Πιο κάτω βλέπουμε την ανταπόκριση της μηχανής:



Επιλέγοντας άλλο κανόνα (S_) ή και τροποποιώντας την αρχική πρόταση, μπορούμε να τρέξουμε και άλλα παραδείγματα.

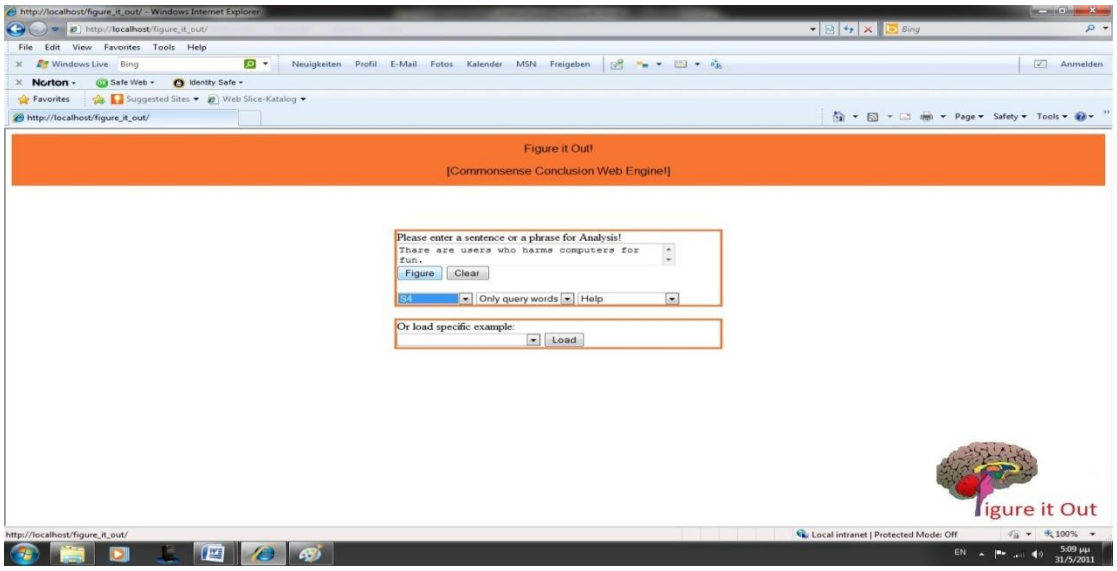
Επιλογή δεύτερη [Μέσω εισαγωγής δικών μας προτάσεων]
Σε αυτή την περίπτωση, ο χρήστης εισάγει την πρόταση της επιλογής του και ακολούθως, επιλέγει τον κανόνα στην επιλογή *select rules*, με τον οποίο θα τροποποιηθεί το ερώτημα, για να δοθεί με τη σειρά του στη μηχανή για εξαγωγή συμπερασμάτων. Μπορεί να επιλέξει ανάμεσα σε 12 κανόνες και, σε περίπτωση που θέλει να δει τη σημασία του καθενός, μπορεί να επιλέξει στη μηχανή την επιλογή *Rules explanation* και ακολούθως το κουμπί *figure* (όπως φαίνεται στην επόμενη εικόνα):



Ας δούμε τώρα τον τρόπο εισαγωγής πρότασης για εξαγωγή συλλογισμών:

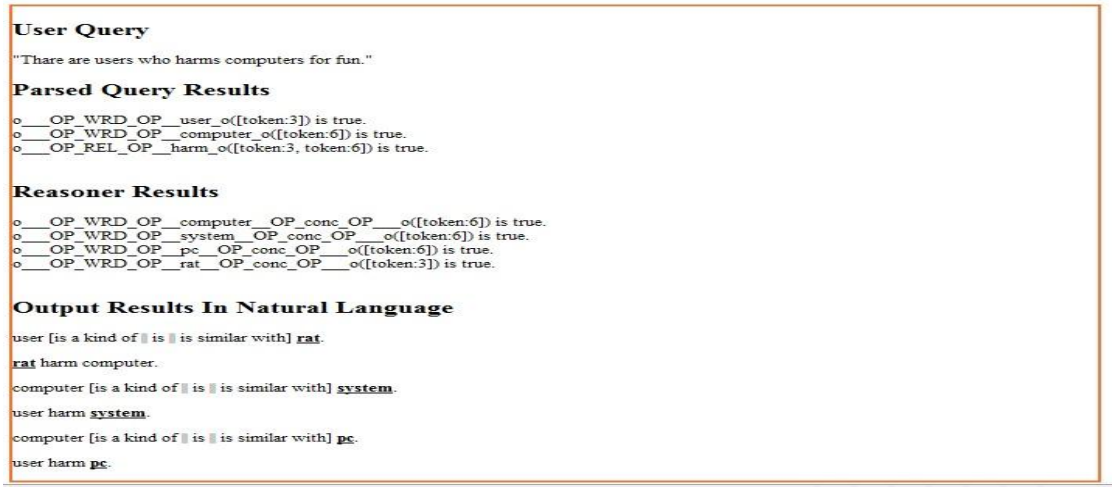
Βήμα 1
Αποφασίζουμε να εισάγουμε για εξαγωγή συμπερασμάτων κοινής λογικής την ακόλουθη πρόταση:
There are users who harms computers for fun.

Βήμα 2
Επίσης αποφασίζουμε να δοκιμάσουμε τον κανόνα *S4*.



Βήμα 3

Και αφού επιλέξουμε το κουμπί *Figure*, παίρνουμε τα εξής συμπεράσματα (σε περίπτωση που η μηχανή δε μπορεί να μας επιστρέψει κάτι, μας ενημερώνει με ανάλογο μήνυμα):



User Query

"Thare are users who harms computers for fun."

Parsed Query Results

o___OP_WRD_OP__user_o([token:3]) is true.
o___OP_WRD_OP__computer_o([token:6]) is true.
o___OP_REL_OP__harm_o([token:3, token:6]) is true.

Reasoner Results

o___OP_WRD_OP__computer__OP_conc_OP___o([token:6]) is true.
o___OP_WRD_OP__system__OP_conc_OP___o([token:6]) is true.
o___OP_WRD_OP__pc__OP_conc_OP___o([token:6]) is true.
o___OP_WRD_OP__rat__OP_conc_OP___o([token:3]) is true.

Output Results In Natural Language

user [is a kind of ▯ is ▯ is similar with] **rat**.

rat harm computer.

computer [is a kind of ▯ is ▯ is similar with] **system**.

user harm **system**.

computer [is a kind of ▯ is ▯ is similar with] **pc**.

user harm **pc**.

Τέλος Παραρτήματος