ΠΑΝΑΓΙΩΤΗΣ ΑΠΟΣΤΟΛΟΠΟΥΛΟΣ (Π17007)

ΕΡΓΑΣΙΑ ΕΠΕΞΕΡΓΑΣΙΑΣ ΦΥΣΙΚΗΣ ΓΛΩΣΣΑΣ

ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2019-2020

Περιεχόμενα

[Θέμα 1ο 2](#_Toc46315112)

[1. Εισαγωγή 2](#_Toc46315113)

[2. Παρουσίαση κεντρικής ιδέας 2](#_Toc46315114)

[2.1 Κωδικοποίηση κειμένου 2](#_Toc46315115)

[2.2 Αναπαράσταση 3](#_Toc46315116)

[2.3 Ομοιότητα διανυσμάτων 4](#_Toc46315117)

[2.4 Στατιστικό γλωσσικό μοντέλο 4](#_Toc46315118)

[2.5 Το μοντέλο word2vec 5](#_Toc46315119)

[2.6 Κατηγοριοποίηση κειμένου 6](#_Toc46315120)

[2.7 Συμπέρασμα 6](#_Toc46315121)

[Θέμα 2ο 7](#_Toc46315122)

[Ερώτημα 1ο 7](#_Toc46315123)

[Ερώτημα 2ο 10](#_Toc46315124)

# Θέμα 1ο

# 1. Εισαγωγή

Ένα από τα σύγχρονα θέματα της Επεξεργασίας Φυσικής Γλώσσας (Natural Processing Language) είναι αυτό της κατηγοριοποίησης των διάφορων κειμένων ως θετικά ή αρνητικά. Η ταξινόμηση των κειμένων σε αυτές τις δύο κατηγορίες είναι πολύ σημαντική καθώς μπορούν μεγάλες πλατφόρμες ή μέσα κοινωνικής δικτύωσης να τα χρησιμοποιήσουν με κατάλληλο τρόπο, προκειμένου να παραχθούν χρήσιμα συμπεράσματα. Επιπρόσθετα, να σημειώσουμε ότι η χρήση αντίστοιχων εφαρμογών γίνεται από πλατφόρμες όπως το Facebook, το Twitter, αλλά και από το Imdb για την αποσαφήνιση των κριτικών από τους χρήστες για τις διάφορες ταινίες/σειρές. Για την αντιμετώπιση του συγκεκριμένου θέματος μπορούμε να φτιάξουμε ένα μοντέλο πρόβλεψης. Πιο συγκεκριμένα, ένας ενδεδειγμένος τρόπος προσέγγισης είναι αυτός των **νευρωνικών δικτύων** και **word embeddings**.

# 2. Παρουσίαση κεντρικής ιδέας

Η υλοποίηση του συγκεκριμένου μοντέλου θα μπορούσε να γίνει με την χρήση της γλώσσας προγραμματισμού **Python**, λόγω των έτοιμων βιβλιοθηκών που μας προσφέρει. Ωστόσο, στη συγκεκριμένη παρουσίαση θα δοθεί βάρος στην ανάλυση του θεωρητικού υποβάθρου και όχι του κώδικα, με την παρουσίαση αρκετών χρήσιμων μοντέλων που συμβάλλουν στην επίτευξη του στόχου.

# 2.1 Κωδικοποίηση κειμένου

Αρχικά, για την επίτευξη του στόχου μας θα πρέπει να έχουμε ένα σύνολο από κείμενα(κριτικές), τα οποία να έχουν κάποια ετικέτα, που να τα χαρακτηρίζει ως θετικά ή αρνητικά. Σε αυτό το σημείο δεν μας ενδιαφέρει το σημασιολογικό περιεχόμενο του κειμένου, δηλαδή δεν θα πραγματοποιήσουμε κάποια σημασιολογική ανάλυση. Επιπρόσθετα, δεν θα ληφθούν υπόψη τα οποία σημασιολογικά σχήματα, όπως οι μεταφορές ή οι παρομοιώσεις.

Μια προσέγγιση για τον «χαρακτηρισμό» ενός κειμένου σε θετικό ή αρνητικό θα μπορούσε να ήταν η δημιουργία ενός λεξικού με όλες τις πιθανές λέξεις, ώστε κάθε φορά να μετράμε το βάρος των λέξεων του κειμένου. Για παράδειγμα σε ένα κείμενο 200 λέξεων θα βρίσκαμε τις θετικές και αρνητικές λέξεις του κειμένου και ανάλογα το πρόσημο, δηλαδή την μεταξύ τους διαφορά, θα εξαγάγαμε το συμπέρασμά μας . Ωστόσο, αυτή η τεχνική εμπεριέχει τον κίνδυνο του καθορισμού του βάρους μιας λέξης με υποκειμενικά κριτήρια και όχι με αντικειμενικά ή ακόμα και την ύπαρξη σαρκασμών και ειρωνειών (σημασιολογική αμφισημία).

Έτσι κάθε κείμενο θα το διαχειριζόμαστε ως μια διαφορετική οντότητα, δίχως να ενδιαφερόμαστε εάν ένα κείμενο είναι θετικό ή αρνητικό, και να φτιάξουμε ένα σύστημα το οποίο να μας υπολογίζει την πιθανότητα να ανήκει σε μία από τις δύο κατηγορίες με βάση τις προηγούμενες προσημειώσεις.

# 2.2 Αναπαράσταση

Για να μπορέσουμε να κατασκευάσουμε ένα μοντέλο πρόβλεψης θα πρέπει να αναπαραστήσουμε τις λέξεις σε μορφή διανύσματος, ώστε να αυτά να δίδονται σαν είσοδο σε κάποιον αλγόριθμο μηχανικής μάθησης, όπως για παράδειγμα σε ένα νευρωνικό δίκτυο. Πιο συγκεκριμένα, αφού σκανάρουμε όλα τα διαθέσιμα κείμενα, βρίσκουμε τις μοναδικές εμφανίσεις της κάθε λέξης και τις αποθηκεύουμε σε μια λίστα-λεξικό (dictionary), με διαστάσεις **v**, όσες και οι λέξεις. Για κάθε λέξη σε αυτό το λεξικό δημιουργούμε ένα διάνυσμα **v** θέσεων, στο οποίο στη θέση στην οποία αντιστοιχεί η θέση της λέξης, βάζουμε την τιμή **1** και σε όλες τις υπόλοιπες. Για παράδειγμα εάν ο λεξικό μας έχει τις λέξεις [“the”, “big”, “dog”], το διάνυσμα που αντιστοιχεί στη λέξη “the” είναι το [1,0,0], στη λέξη “big” είναι το [0,1,0] και στη λέξη “dog” είναι το [0,0,1]. Συνήθως, κατά την διαδικασία «χτισίματος του λεξικού» για να κρατήσουμε τον αριθμό των λέξεων μικρό μετατρέπουμε τα κείμενα σε lower case και παραλείπουμε τα στοιχεία στίξης, αλλά και αφαιρούμε τις λέξεις με λιγότερες από **k** εμφανίσεις, όπου **k** είναι μια παράμετρος που την θέτουμε εμείς.

# 2.3 Ομοιότητα διανυσμάτων

Σημαντικό βήμα είναι η αξιοποίηση των διανυσμάτων που δημιουργήσαμε, δηλαδή να είμαστε σε θέση να συγκρίνουμε πόσο δύο διανύσματα στον χώρο είναι «κοντά», δηλαδή σημασιολογικά. Επειδή δύναται να έχουμε διανύσματα με τελείως διαφορετικά μήκη δεν θα χρησιμοποιήσουμε το εσωτερικό γινόμενο, αλλά η γωνία συνημίτονου (Εικόνα 1).

# 2.4 Στατιστικό γλωσσικό μοντέλο

Με τον όρο αυτό αναφερόμαστε σε μια κατανομή, είναι δηλαδή μια ακολουθία κατανομής λέξεων. Αυτό που μας ενδιαφέρει, είναι η εύρεση της κατανομής της λέξης η οποία «ακολουθεί» μια συγκεκριμένη ακολουθία λέξεων. Η συγκεκριμένη μοντελοποίηση είναι ευρέως γνωστή σε εφαρμογές όπως το Word (spell checking), η αναγνώριση φωνής ακόμα και το autocomplete. Aαναλυτικότερα για την πρώτη εφαρμογή, για μια λέξη που δεν είναι στο λεξικό, αναζητούνται κάποια μέτρα απόστασης, ώστε να βρεθούν οι λέξεις που ταιριάζουν, ταξινομημένες πάντα σύμφωνα με τις πιθανότητές τους. Ωστόσο, στη φυσική γλώσσα υπάρχει η πιθανότητα να δημιουργηθεί μια καινούργια πρόταση, η οποία να μην υπάρχει σε καμία συλλογή οπότε η αντιμετώπιση είναι πιο περίπλοκη.

# 2.5 Το μοντέλο word2vec

Για την αντιμετώπιση του παραπάνω προβλήματος υιοθετήθηκε η πρόταση του Mikolov, γνωστό και ως το μοντέλο **word2vec**. Κάθε φορά που εκπαιδεύουμε ένα τέτοιο μοντέλο, επιθυμούμε είτε να του δώσουμε k λέξεις προσπαθώντας να βρούμε την επόμενη, είτε να του δίνουμε μια λέξη και θα προσπαθούμε να βρούμε σε ένα «παράθυρο», έστω k λέξεων ποιες είναι οι πιο πιθανές λέξεις που το συνοδεύουν. Η πρώτη αρχιτεκτονική ονομάζεται **CBOW** και η δεύτερη **Skip-gram** (Εικόνα)**.** Το μοντέλο αυτό για να είναι λειτουργικό και αξιόπιστο θα πρέπει αφού το «χτίσουμε», να το εκπαιδεύσουμε. Για την επίτευξη του στόχου μας η Python προσφέρει μια σειρά από βιβλιοθήκες και συναρτήσεις, οι οποίες συμβάλλουν σε αυτό.

**Εικόνα που περιέχει χάρτης

Περιγραφή που δημιουργήθηκε αυτόματα**

Εικόνα 1- Μοντέλα word2vec

# 2.6 Κατηγοριοποίηση κειμένου

Έχοντας στην φαρέτρα μας όλα τα παραπάνω εργαλεία είμαστε σε θέση να τα συνδυάσουμε, ώστε να κατηγοριοποιήσουμε ένα νέο-άγνωστο κείμενο σε μία ευρύτερη κατηγορία. Αρχικά, κάθε κείμενο θα αναπαρίσταται ως ο μέσος όρος των διανυσμάτων των λέξεων του συγκεκριμένου κειμένου[[1]](#footnote-1). Για να είμαστε σίγουροι ότι τα αποτελέσματα της εκπαίδευσης του μοντέλου μας θα είναι τέτοια, ώστε να μπορεί να ανταπεξέλθει σε ένα νέο κείμενο, θα πρέπει τα δεδομένα μας, δηλαδή τα κείμενα τα οποία είναι ήδη κατηγοριοποιημένα να τα χωρίσουμε σε δύο κατηγορίες: αυτά που θα χρησιμοποιηθούν για την εκπαίδευση και αυτά για τον έλεγχο. Η αναλογία συνήθως είναι 75% για την πρώτη κατηγορία και 25% για τη δεύτερη. Επιπρόσθετα, για τον αλγόριθμο μηχανικής μάθησης χρησιμοποιούμε **Λογιστική Παλινδρόμηση,** δηλαδή το μοντέλο θα μάθει την πιθανότητα η ετικέτα να είναι ένα (θετική) δεδομένου ενός κειμένου. Συνεπώς, ο αλγόριθμος προσπαθεί να βρει τα βάρη, ώστε να ελαχιστοποιήσει το σφάλμα στα δεδομένα εκπαίδευσης. Με αυτή τη τεχνική είμαστε σε θέση να γνωρίζουμε σε τι ποσοστό ταξινομεί ορθά τα κείμενα το μοντέλο.

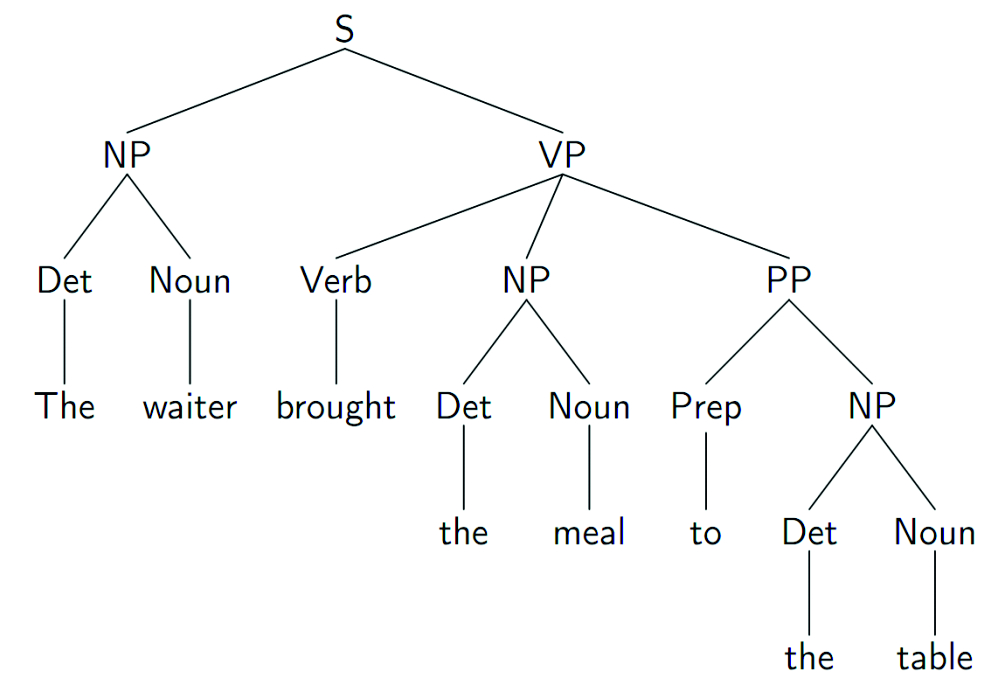
# 2.7 Συμπέρασμα

Σύμφωνα με όλα τα παραπάνω, αντιλαμβανόμαστε ότι αυτή η διαδικασία προϋποθέτει την σωστή εκπαίδευση του μοντέλου μας, εισάγοντάς του δεδομένα-κείμενα τα οποία είναι ήδη προσημειωμένα. Επίσης, όσο πιο μεγάλα είναι τα κείμενά μας τόσο μεγαλύτερη ακρίβεια θα υπάρχει σε νέα κείμενα. Το θέμα αυτό είναι από τα πιο σύγχρονα στην Επεξεργασία Φυσικής Γλώσσας και αδιαμφισβήτητα έχει πολλές προεκτάσεις.

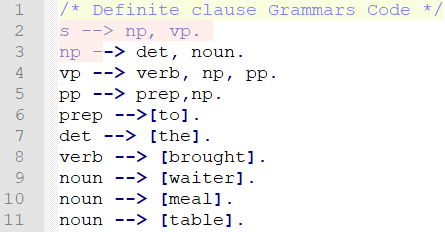
# Θέμα 2ο

# Ερώτημα 1ο

Προκειμένου να αναγνωρίσουμε την πρόταση : [the, waiter, brought, the, meal, to, the, table], σύμφωνα με το παρακάτω σχήμα (**Εικόνα** ), θα πρέπει να χρησιμοποιήσουμε την ακόλουθη γραμματική , η οποία είναι σε μορφή DCG (Defining Clause Grammars)[[2]](#footnote-2).

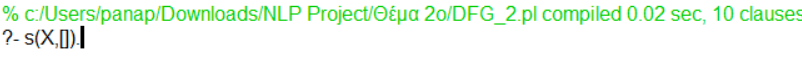


Εικόνα 2- Συντακτικό δέντρο



Εικόνα 3- Γραμματική για την αναγνώριση της πρότασης

Συνεπώς, παρατηρούμε ότι μια πρόταση χωρίζεται σε μία *noun\_phrase* και μια *verb\_phrase*, οι οποίες στη συνέχεια θα αναλυθούν περαιτέρω. Αυτό που φαίνεται παραπάνω είναι η ανάλυση μιας *noun\_phrase* σε *det* και *noun*, ενώ μια *verb\_phrase* σε *verb, noun\_phrase* και *preposition\_phrase.* H τελευταία με τη σειρά της αναλύεται σε *prep (preposition)* και *noun\_phrase.* Επόμενο βήμα είναι η ανάλυση των *prep,det,verb* και *noun*, δηλαδή ο καθορισμός του συνόλου των δυνατών τιμών που μπορούν να λάβουν. Αυτό θα έχει και ως αποτέλεσμα την αναγνώριση της ζητούμενης πρότασης. Για την επιβεβαίωση της ορθότητας της γραμματικής μας έχουμε δύο δυνατούς τρόπους: ο πρώτος είναι να δημιουργήσουμε όλες τις δυνατές προτάσεις και να βρούμε την ζητούμενη και ο δεύτερος να «ρωτήσουμε» αν η συγκεκριμένη πρόταση είναι true or false. Παρακάτω παρατίθενται και οι δύο τρόποι.



Εικόνα που περιέχει κείμενο, εφημερίδα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα 4- Εμφάνιση όλων των δυνατών προτάσεων



Εικόνα 5- 2ος τρόπος αναγνώρισης

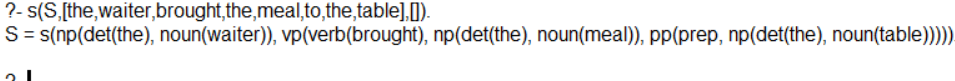
# Ερώτημα 2ο

Στη Prolog μπορούμε να παράγουμε συντακτικά δέντρα που είναι ισοδύναμα με αυτό της εικόνας 1, στην μορφή **functor**. Προκειμένου, να επιτευχθεί αυτό θα πρέπει να αλλάξουμε την παραπάνω γραμματική, όπως φαίνεται παρακάτω[[3]](#footnote-3).

Εικόνα που περιέχει οθόνη, τηλέφωνο, άνδρας, ιδιοκτησία

Περιγραφή που δημιουργήθηκε αυτόματα

Ουσιαστικά, αυτό που παρατηρούμε είναι ότι η φιλοσοφία της ανάλυσης της κάθε φράσης είναι ίδια. Οι αλλαγές που έγιναν αφορούν μόνο το κομμάτι της εμφάνισης του συντακτικού δέντρου σε μορφή **functor.**



Εικόνα 6- Εμφάνιση συντακτικού δέντρου σε μορφή **functor**

Στην παραπάνω εικόνα φαίνεται το συντακτικό δέντρο σε μορφή **functor**, για την αναγνώριση της πρότασης [the, waiter, brought, the, meal, to, the, table].

# Θέμα 4ο

# 1. Παρουσίαση κεντρικής ιδέας

Σκοπός αυτής της άσκησης είναι η ανάπτυξη ενός Λεκτικού Αναλυτή σε άλλη γλώσσα προγραμματισμού, η οποία θα διαβάζει μια μικρή ιστορία και θα είναι σε θέση να παράγει μια λίστα από προτάσεις, κάθε μία από τις οποίες περιέχει μια λίστα από λέξεις. Για την υλοποίηση της συγκεκριμένης άσκησης χρησιμοποιήθηκε η γλώσσα προγραμματισμού **Python version 3.8.**  Αξίζει να σημειωθεί ότι χρησιμοποιήθηκε η βιβλιοθήκη **nltk**, η οποία αφορά την Επεξεργασία Φυσικής Γλώσσας. Έτσι στη συγκεκριμένη εφαρμογή θα ζητάμε από τον χρήστη να εισάγει το όνομα του αρχείου όπου επιθυμεί να προσπελαστεί.

**Θέμα 7**

Στη παρούσα ενότητα θα ασχοληθούμε με το 7ο και τελευταίο θέμα της εργασίας.

1. Εκφώνηση

Θέμα 7ο (40 μονάδες):

Χρησιμοποιείστε το σύνολο του κώδικα που σας δόθηκε σε Prolog για την «κατανόηση» μιας μικρής ιστορίας. Πρέπει να παράγετε τα περιεχόμενα της βάσης γνώσης και να μπορείτε να εισάγετε νέες πληροφορίες από το πληκτρολόγιο, να κάνετε ερωτήσεις στην βάση γνώσης, κ.λπ., παρόμοιες με αυτές της πρότυπης λύσης. Εναλλακτικά μπορείτε να χρησιμοποιήσετε άλλες γλώσσες προγραμματισμού ή libraries τον κώδικα και την λειτουργικότητα των οποίων πρέπει να τεκμηριώσετε πειστικά και κατανοητά.

2. Κεντρική Ιδέα Υλοποίησης

Σε αυτή την ενότητα θα αναφερθούμε εκτενώς στην υλοποίηση μας για το 7ο ερώτημα της εργασίας.

2.1. Εισαγωγικό Πλαίσιο

Σκοπός της παρούσας άσκησης είναι με βάση το αρχείο που μας δόθηκε *(Themis\_understand\_NLP\_v1.pl)* να το τροποποιήσουμε καταλλήλως, δημιουργώντας νέες ιστορίες και μια νέα βάση γνώσης, στην οποία ο χρήστης θα μπορεί να εισάγει νέες πληροφορίες από το πληκτρολόγιο και να κάνετε ερωτήσεις.

Τα παραπάνω στην ολότητά τους θα επεξηγηθούν στις παρακάτω ενότητες.

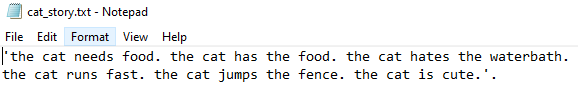
2.2. Δημιουργία Ιστοριών

Πρωταρχικό μέλημά μας, για τη διαδικασία επίλυσης της άσκησης είναι η δημιουργία νέων ιστοριών (αρχείων κειμένων).

Κατ’ απόλυτη αντιστοιχία με τα προδοθέντα αρχεία, υλοποιήθηκαν δύο αρχεία κειμένου – ιστορίες.

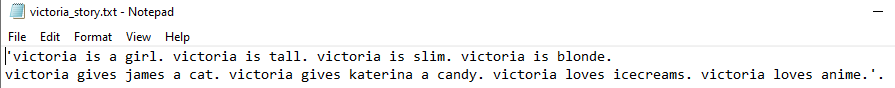
Τα προαναφερθέντα αρχεία ιστοριών βρίσκονται στον φάκελο Subject 7 με ονόματα cat\_story.txt και victoria\_story.txt και απεικονίζονται παρακάτω:

* cat\_story.txt



*Εικόνα 9 : 1η Ιστορία cat\_story.txt*

* victoria\_story.txt



*Εικόνα 10 : 2η Ιστορία victoria\_story.txt*

Τα αρχεία αυτά περιέχουν σύντομες ιστορίες στα πρότυπα που μας υποδείξατε καθώς περιλαμβάνουν μικρές περιόδους και αρκετά διαφορετικά μέρη του λόγου (υποκείμενα, ρήματα, αντικείμενα κλπ.).

2.3. Ορολογία Υλοποίησης

Σε αυτή την ενότητα θα αναφερθούμε στην ορολογία που χρησιμοποιείτε στο αρχείο μας.

Ακολουθούν οι επεξηγήσεις των εννοιών:

* Proper Nouns (pn):Αναφέρεται σε ονόματα και τοποθεσίες. Μπορεί να είναι υποκείμενο αλλά και αντικείμενο σε μια πρόταση και δεν παίρνει άρθρο. Παράδειγμα δήλωσης proper noun στο πρόγραμμα:

*pn(victoria)-->[victoria].*

* Intransitive Verbs (iv):Ρήμα που δεν δέχεται αντικείμενο. Αναφέρει πράξη του υποκειμένου και μπορεί από μόνο του να αποτελεί φράση (verbal phrase πιο συγκεκριμένα). Παράδειγμα:

*iv(runs,s)-->[runs].*

*iv(runs,q)-->[running].*

Το ίδιο ρήμα (iv(runs,s)) το βρίσκουμε σε δυο διαφορετικούς τύπους (iv(runs,s)). Οι τύποι συνδέονται μεταξύ τους με το κοινό όνομα και είναι τα αρχικά για τα sentence και query όπου και χρησιμοποιούνται πιο συχνά. (π.χ.: Cat runs / Who is running?)

* Auxiliary Verbs (av): Auxiliary verbs ή αλλιώς helping verbs ονομάζονται τα ρήματα που λειτουργούν ως βοηθητικά στα ρήματα που τα ακολουθούν, δίνοντας τους ή συμπληρώνοντας το νόημα τους. Δύο ρήματα τέτοιου τύπου χρησιμοποιούνται στην εργασία τα οποία δίνονται παρακάτω:

av(is)-->[is].

av(does)-->[does].

* Verbs (v): Ο όρος χρησιμοποιείτε για όλα τα ρήματα τα οποία δέχονται αντικείμενο. Μεταξύ τους υπάρχουν πολλοί τύποι ρημάτων (emotion verbs, action verbs, helping verbs).

Δίνονται πάλι όπως στα iv δύο τύποι συνδεόμενοι για χρήση σε διαφορετικούς τύπους προτάσεων. Στην παραγωγή γνώσης παίρνουν δυο ορίσματα, το υποκείμενο και το αντικείμενο, σε αυτήν την σειρά. Παραδείγματα βρίσκονται παρακάτω:

*v(loves,s)-->[loves]. v(loves,q)-->[love].*

*v(hates,s)-->[hates]. v(hates,q)-->[hate].*

*v(has,s) -->[has]. v(has,q) -->[have].*

* Transitive Verbs (tv):Μεταβατικά ρήματα, δέχονται υποκείμενο, και πάνω από ένα αντικείμενο. Ένα άμεσο και ένα απλό. Στο παράδειγμα «Ο Κώστας δίνει ένα μήλο στην Μαρία» το μήλο είναι το αντικείμενο και η Μαρία το άμεσο αντικείμενο. Χρησιμοποιείτε ξεχωριστά καθώς στην βάση γνώσης έχει τρία γνωρίσματα (1 υποκείμενο, 2 αντικείμενα) και χρησιμοποιούνται τρείς τύποι του. Στην τωρινή έκδοση του προγράμματος υπάρχει μόνο ένα ρήμα το οποίο φαίνεται και παρακάτω:

*tv(gives,s)-->[gives]. tv(gives,q)-->[give]. tv(gives,q2)-->[giving].*

* Adverb (adv):Μέρος του λόγου που περιγράφει ένα ρήμα, χρησιμοποιείτε μετά από αυτόκαι αποτελείτε μόνο από μια λέξη. Παράδειγμα:

*adj(quickly)-->[quickly].*

* Adjectives (adj):Στην ελληνική γλώσσα τα επίθετα, χρησιμοποιούνται ως χαρακτηρισμός προσώπου στο συγκεκριμένο πρόγραμμα. Παρακάτω λίγα παραδείγματα:

*adj(tall)-->[tall].*

*adj(short)-->[short].*

* Determiner (det):Ονομάζονται οι λέξεις που είτε χρησιμοποιούνται για την κλήση ενός ουσιαστικού (άρθρα όπως a, the) είτε λέξεις που χρησιμοποιούνται για να δηλώσουν κτίση (my, his). Σε κάθε περίπτωση χρησιμοποιούνται πριν το ουσιαστικό σε φράση. Στο συγκεκριμένο πρόγραμμα χρησιμοποιούμε τα άρθρα (a, the) και τις αντωνυμίες (who, which) αλλά μόνο τα πρώτα δηλώνονται ως det για πρακτικούς λόγους. Στο πρόγραμμα, λοιπόν determiners ορίζονται τα άρθρα. Επίσης δηλώνετε και η κενή λίστα, καθώς δεν χρειάζεται πάντα ένα άρθρο πριν ένα ουσιαστικό. Η πλήρης λίστα βρίσκεται παρακάτω:

*det(a) -->[a].*

*det(the) -->[the].*

* Noun (n):Τα ουσιαστικά στο πρόγραμμα χρησιμοποιούνται ως αντικείμενα και μόνο. Τον ρόλο του υποκειμένου παίρνουν τα proper nouns, ένα υποείδος ουσιαστικών που εξηγείτε παραπάνω. Μερικά παραδείγματα τους:

*n(book)-->[book].*

*n(cat)-->[cat].*

* Noun Phrase (np):Φράση που δηλώνει το υποκείμενο σε μια πρόταση. Κανονικά μπορεί να αποτελείτε και από ένα proper noun αλλά αυτή η δήλωση (np(N):- pn(N)) δεν γίνετε για πρακτικούς λόγους και όχι επιστημονικής ακρίβειας. Ένας άλλος τύπος φράσης είναι o εξής και ο μόνος που χρησιμοποιείτε στο συγκεκριμένο πρόγραμμα, πέρα από το proper noun:

np(N) --> det(\_),n(N).

* Verbal Phrase: Φράσεις που περιέχουν το ρήμα και το αντικείμενο (αν αυτό υπάρχει):

*vp(1,V,N) --> v(V,s),n(N).*

*vp(2,is,A) --> av(is),adj(A).*

*vp(3,V,N) --> v(V,s),pn(N).*

2.4. Τύποι Γνώσης

Στην υλοποίησή μας υπάρχουν πέντε τύποι γνώσης στους οποίους βασίζεται το πρόγραμμα. Αυτοί αντιστοιχούν άμεσα σε πέντε τύπους προτάσεων.

Σε αυτήν την ενότητα παρουσιάζονται ένας προς ένας οι τύποι γνώσης:

* Τύπος 1 (υποκείμενο – ρήμα – αντικείμενο):

Αυτός ο τύπος γνώσης αφορά μια πράξη που πράττει ένα φυσικό πρόσωπο (sem\_np) σε ένα αντικείμενο (sem\_vp). Η ακολουθία φράσεων είναι η εξής:

sem(1,Sem) 🡪 sem\_np(N), sem\_vp(1,V,N1)

Ο τύπος γνώσης που παράγεται είναι:

V(N,N1) ή αλλιώς ρήμα(υποκείμενο, αντικείμενο)

Η γνώση περνάει στο Sem, το δεύτερο όρισμα του sem. Η μετατροπή γίνετε με τον παρακάτω τρόπο:

Sem=..[V,N,N1]

*Παράδειγμα:*

1 ?- sem(1,A,B,[]).

A = loves(victoria, icecreams),

B = [victoria, loves, icecreams] .

* Τύπος 2 (υποκείμενο – επίθετο):

Αυτός ο τύπος γνώσης αφορά στον προσδιορισμό χαρακτηριστικού σε ένα πρόσωπο. Η ακολουθία φράσεων είναι η εξής:

sem(2,Sem) --> sem\_np(N), sem\_vp(2,\_,A),

Ο τύπος γνώσης που παράγεται είναι:

A(N) ή αλλιώς επίθετο(πρόσωπο)

Η μετατροπή γίνετε με τον παρακάτω τρόπο:

Sem=..[A,N]

*Παράδειγμα*:

2 ?- sem(2,A,B,[]).

A = tall(victoria),

B = [victoria, is, tall] .

* Τύπος 3 (υποκείμενο – ρήμα):

Αυτός ο τύπος γνώσης αφορά μια πράξη που πράττει ένα φυσικό πρόσωπο (sem\_np(N)) και δεν μεταφέρεται, αφορά δηλαδή το ίδιο. Η ακολουθία φράσεων είναι η εξής:

sem(3,Sem) --> sem\_np(N), sem\_iv(V,s),

Ο τύπος γνώσης που παράγεται είναι:

V(N) ή αλλιώς ρήμα(υποκείμενο)

Η μετατροπή γίνετε με τον παρακάτω τρόπο:

Sem=..[V,N]

*Παράδειγμα:*

3 ?- sem(4,A,B,[]).

A = runs(cat),

B = [cat, runs] .

* Τύπος 4 (υποκείμενο – ρήμα – επίρρημα):

Αυτός ο τύπος γνώσης αφορά μια πράξη που πράττει ένα φυσικό πρόσωπο (sem\_np(N)) και δεν μεταφέρεται, αφορά δηλαδή το ίδιο αλλά ακολουθείτε από ένα όρισμα-επίρρημα. Η ακολουθία φράσεων είναι η εξής:

sem(4,Sem) --> sem\_np(N), sem\_iv(V,s), sem\_adv(A),

Ο τύπος γνώσης που παράγεται είναι:

V(N,Α) ή αλλιώς ρήμα(υποκείμενο, επίρρημα)

Η μετατροπή γίνετε με τον παρακάτω τρόπο:

Sem=..[V,N,Α]

*Παράδειγμα:*

4 ?- sem(5,A,B,[]).

A = runs(cat, quickly),

B = [cat, runs, quickly] .

* Τύπος 5 (υποκείμενο – ρήμα – άμεσο αντικείμενο – αντικείμενο):

Αυτός ο τύπος γνώσης αφορά μια πράξη που πράττει ένα φυσικό πρόσωπο (sem\_np(N)) σε ένα άλλο.

Η γνώση περιέχει και ένα ακόμη αντικείμενο με το οποίο ενεργεί το πρώτο πρόσωπο στο δεύτερο.

Ενναλακτικά η πράξη ορίζεται με το ρήμα και το αντικείμενο (π.χ. ρίχνει πέτρες) και ενεργείτε από το υποκείμενο στο άμεσο αντικείμενο (Ο Πέτρος ρίχνει πέτρες στον Γιάννη). Η ακολουθία φράσεων είναι η εξής:

sem(5,Sem) -->sem\_np(N), sem\_tv(V,s), sem\_np(N1), sem\_np(N2),

Ο τύπος γνώσης που παράγεται είναι:

V(N,N1,N2) ή αλλιώς ρήμα(υποκείμενο, άμεσο\_αντικείμενο, αντικείμενο)

Η μετατροπή γίνετε με τον παρακάτω τρόπο:

Sem=..[V,N,N1,N2]

*Παράδειγμα:*

5 ?- sem(6,A,B,[]).

A = gives(victoria, katerina, candy),

B = [victoria, gives, katerina, a, candy] .

2.5. Εισαγωγή Δεδομένων στη Βάση Γνώσης

Me βάση την προηγούμενη ενότητα, υπάρχουν πέντε τύποι γνώσης που αφορούν σε σχέσεις μεταξύ ατόμων, αλλά και προσωπικές πράξεις και χαρακτηριστικά προσώπων.

Ο κώδικας της υλοποίησης που έχει παρουσιαστεί ως τώρα, έχει την δυνατότητα με την παρακάτω εντολή να διαβάσει προτάσεις με τις παραπάνω δομές, αλλά και να αποθηκεύσει την πληροφορία σε μια βάση γνώσης (knowledge base), στην οποία μπορούν να τεθούν ερωτήματα.

Ο κώδικας που χρησιμοποιείτε για την εισαγωγή πληροφορίας, αν αυτή είναι έγκυρη, είναι ο παρακάτω:

---------------------------------------------------------------------------------------------------------

tell(Sentence):-

sem(\_, Sem, Sentence, []),

assert(kb\_fact(Sem)),

nl,write(kb\_fact(Sem)), nl, write(' added to knowledge base.'),nl, !.

---------------------------------------------------------------------------------------------------------

Ο χρήστης εισάγει μόνο το κείμενο (Sentence) και αυτό δοκιμάζεται από την Prolog σε ένα ερώτημα της μορφής:

sem(\_, Sem, Sentence, []),

Με αυτό τον τρόπο η Prolog εκτελεί το assert(kb\_fact(Sem)) και περνάει την γνώση στο πρόγραμμα. Οι εντολές που ακολουθούν απλά δηλώνουν τον τύπο πρότασης που διαβάστηκε και δίνουν μια κενή γραμμή.

//pic tell

2.5. Τύποι Ερωτήσεων

Ενώ οι τύποι γνώσης είναι περιορισμένοι, οι τύποι ερωτήσεων που εξετάζονται είναι όλοι οι πιθανοί πάνω στην βάση γνώσης που έχουμε.

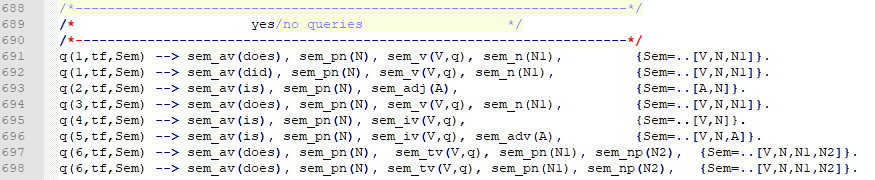
Είναι δε δυνατόν να χωριστούν σε δύο υποκατηγορίες. Στην εξακρίβωση της ορθότητας μιας πληροφορίας και στην αναζήτηση πληροφορίας.

2.5.1. Ερωτήσεις Εξακρίβωση Πληροφορίας

Στα σχόλια του προγράμματος ονομάζονται yes/no queries καθώς απαιτούν μια καταφατική ή μια αρνητική απάντηση.

Οι ερωτήσεις παράγουν μια σημασία / γνώση με παρόμοιο τρόπο, όπως γίνετε και στις προτάσεις, και στην συνέχεια αντί να προστεθεί αυτή η γνώση, εξετάζεται αν υπάρχει στην βάση γνώσης.

Οι τύποι που ορίζουν τις έγκυρες ερωτήσεις και είναι οι ακόλουθοι:



*Εικόνα 11 : Οι Τύποι που ορίζουν τις έγκυρες ερωτήσεις.*

Ο κανόνας που χρησιμοποιείτε για να τεθούν οι ερωτήσεις αυτού του τύπου είναι ο εξής:

---------------------------------------------------------------------------------------------------------

ask(X):- q(\_, tf, Sem, X, []),

if\_then\_else(kb\_fact(Sem), write('Yes.'), write('No.') ), !.

---------------------------------------------------------------------------------------------------------

Εξετάζει την φυσική γλώσσα σε έναν τύπο ερώτησης tf και αν είναι επιτυχής γυρίζει το Sem. Στην συνεχεία εξετάζεται αυτό και αν ισχύει, η Prolog τυπώνει Yes. Ειδάλλως τυπώνει No.

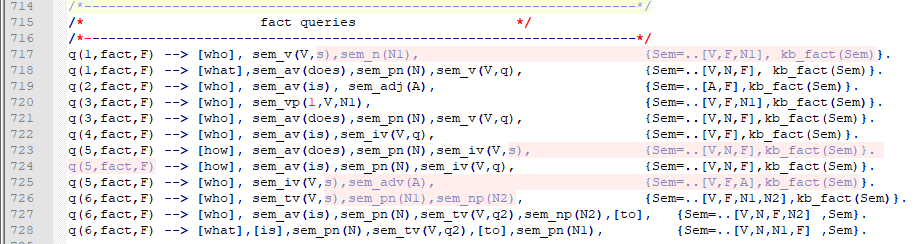
2.5.2. Ερωτήσεις Γεγονότων

Στα σχόλια του προγράμματος ονομάζονται fact queries καθώς απαιτούν ένα γεγονός σαν απάντηση.

Ένα γεγονός, μια καταχώρηση στην βάση γνώσης αποτελείτε από δύο έως τέσσερα μέλη.

Σε κάθε ερώτηση μπορεί να λείπει ένα μέλος της σχέσης και να ελέγχετε εάν υπάρχει καταχώρηση που ταιριάζει.

Ακολουθούν οι ερωτήσεις γεγονότων :



*Εικόνα 12 : Τύποι Ερωτήσεων Γεγονότων.*

Όπως στις προτάσεις και στον άλλο τύπο ερωτημάτων, χρησιμοποιείτε ένας κανόνας για την εξαγωγή απάντησης. Παρακάτω η διατύπωση της:

---------------------------------------------------------------------------------------------------------

ask(X):- q(\_,fact, Fact, X, []), write(Fact), !.

---------------------------------------------------------------------------------------------------------

Όπως αναφέρθηκε παραπάνω, το τρίτο όρισμα του q αποτελεί τον άγνωστο. Ωστόσο δίνετε η πλήρης διαδικασία:

* Το q() τροφοδοτείτε με το Χ, την φυσική γλώσσα.
* Αυτή παρέχει όλους τους όρους που χρειάζονται, και παράγεται η μορφή γνώσης με έναν άγνωστο, το Fact.
* Όταν δημιουργηθεί το Sem, θέτετε ως στόχος γεμίζοντας το fact με πιθανή απάντηση, αν αυτή υπάρχει.
* Γυρνώντας στην ask γράφεται η πιθανή απάντηση.

1. Να σημειωθεί ότι η αναπαράσταση της κάθε λέξης έχει περιεγράφηκε στην ενότητα 2.2. [↑](#footnote-ref-1)
2. H συγκεκριμένη γραμματική υπάρχει και στο αρχείο ‘DFG\_2.pl’ στον φάκελο **Θέμα 2ο.** [↑](#footnote-ref-2)
3. Η συγκεκριμένη γραμματική υπάρχει και στο αρχείο ‘syntax\_generator’ στον φάκελο **Θέμα 2ο.** [↑](#footnote-ref-3)