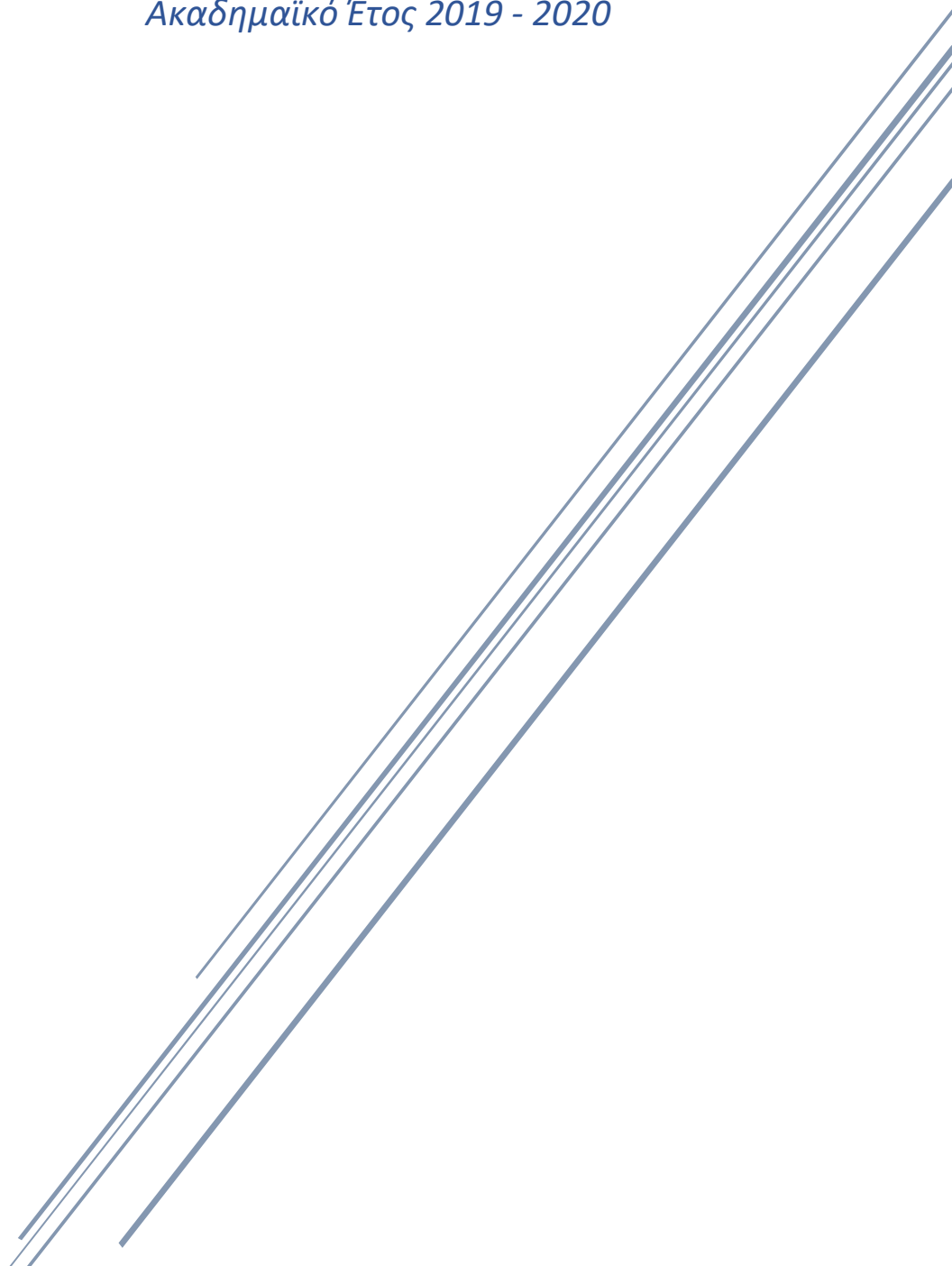


Επεξεργασία Φυσικής Γλώσσας

Απαλλακτικής Εργασία

Ακαδημαϊκό Έτος 2019 - 2020



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
UNIVERSITY OF PIRAEUS

Δημήτρης Ματσαγγάνης, Π17068

Περιεχόμενα

Εισαγωγή Εργασίας	4
Προσωπικοί Προβληματισμοί και Κίνητρα Χρήσης NPL μεθόδων	4
Θέμα 1	9
1. Εκφώνηση	9
2. Εισαγωγή	9
3. Παρουσίαση Ιδέας	10
3.1. Εισαγωγικό Πλαίσιο Ιδέας	10
3.2. Παρουσίαση Ιδέας Υλοποίησης	11
3.3. Εκπαίδευση Μοντέλου – Train Classifiers	11
3.4. Κωδικοποίηση Κειμένου – Κριτικής	12
3.5. Αναπαράσταση Λέξεων Μέσω Διανυσμάτων	13
3.6. Ομοιότητα Λέξεων – Διανυσμάτων	13
3.7. Στατιστικό γλωσσικό μοντέλο	14
3.8. Μοντέλο word2vec	15
3.9. Κατηγοριοποίηση Κειμένου – Κριτικής	16
3.10. Συμπέρασμα Υλοποίησης	16
Θέμα 2	17
Εκφώνηση	17
Ερώτημα 1 ^ο	18
Ερώτημα 2 ^ο	20
Θέμα 3	22
Εκφώνηση	22
Εισαγωγικό Πλαίσιο	23
Αρχικοποίηση Υλοποίησης	23
Θέμα 4	26
1. Εκφώνηση	26
2. Εισαγωγικό Πλαίσιο	26
3. Επεξήγηση Κώδικα	27

3.1. Διαδικασία sentence_split()	27
3.2. Εκτέλεση Υλοποιήσιμου	28
4. Αποτελέσματα Εφαρμογής	28
5. Απαιτήσεις Συστήματος και Εκτέλεση	29
Θέμα 5	30
1. Εκφώνηση	30
2. Εισαγωγικό Πλαίσιο	30
3. Επεξήγηση Κώδικα	31
3.1. Συντακτικοί κανόνες (Grammar Rules)	31
3.2. Καθορισμός πρότασης	32
3.3. Ορισμός του Parser	32
3.4. Δημιουργία Συντακτικού Δέντρου	32
4. Αποτελέσματα Εφαρμογής	33
5. Απαιτήσεις Συστήματος και Εκτέλεση	33
Θέμα 6	34
1. Εκφώνηση	34
2. Εισαγωγικό Πλαίσιο	34
3. Επεξήγηση Κώδικα – Αποτελέσματα	34
Θέμα 7	38
1. Εκφώνηση	38
2. Κεντρική Ιδέα Υλοποίησης	38
2.1. Εισαγωγικό Πλαίσιο	38
2.2. Δημιουργία Ιστοριών	39
2.3. Ορολογία Υλοποίησης	40
2.4. Τύποι Γνώσης	43
2.5. Εισαγωγή Δεδομένων στη Βάση Γνώσης	46
2.6. Τύποι Ερωτήσεων	47
2.7. Αποτελέσματα Υποβολής Ερωτήσεων στη Βάση Γνώσης	50
Περιεχόμενα Απεσταλμένου Αρχείου	51

Εισαγωγή Εργασίας

Απαλλακτική εργασία Ιουνίου – Σεπτεμβρίου 2020

Τα θέματα που πραγματεύεται και θέτει προς υλοποίηση η παρούσα εργασία είναι τα ακόλουθα:

- Ανάπτυξη & Τεκμηρίωση
- Λεκτικού Αναλυτή
- Συντακτικού Αναλυτή
- Σημασιολογικού Αναλυτή
- Διαχείριση Βάσης Γνώσης

Προσωπικοί Προβληματισμοί και Κίνητρα Χρήσης NPL μεθόδων

Όπως όλοι γνωρίζουμε, η γλώσσα – καθολικά στην ανθρώπινη ιστορία - αποτελεί κομμάτι της επικοινωνίας, με άλλα λόγια είναι ένα καίριο συστατικό για την συνύπαρξη και συχνά για την επιβίωση των ανθρώπων.

Η επικοινωνία βασίζεται σε άτυπους κανόνες τους οποίους χρησιμοποιούμε στην καθημερινότητά μας, χωρίς να έχουμε συνείδηση της πράξης αυτής.

Θα ήταν εξάλλου φοβερά κουραστικό να σκεφτόμασταν αδιάπτωτα και κάθε φορά τον κατάλληλο τρόπο δημιουργίας μιας και μόνο πρότασης, πόσο μάλλον την κατασκευή ενός διαλόγου.

Σπάνιες είναι οι φορές που οι περιστάσεις υποδεικνύουν αυστηρά προσεγμένη ομιλία ή γραπτή επικοινωνία.

Δεν είναι τυχαίο το γεγονός ότι από πολύ νωρίς, στα πρώτα κιόλας στάδια της ζωής, ο άνθρωπος κατακτά ασυνείδητα τις δομικές λειτουργίες του λόγου, κάτι που αργότερα το σχολείο συμπληρώνει και τελειοποιεί – κινούμενος

πάντα ασυμπτωτικά, ως προς τη τελειότητα του λόγου, καθώς αυτή δεν υφίσταται.

Η τυπική εκπαίδευση συμπεριλαμβάνει, εκτός των άλλων, την άριστη γνώση γραμματικών κανόνων και συντακτικών εννοιών.

Από τις πρώτες κιόλας τάξεις του Δημοτικού οι μαθητές καλούνται να κατανοήσουν βασικές χρήσεις της γλώσσας τους και να αποδώσουν τις νεοαποκτηθείσες γνώσεις στον γραπτό και προφορικό λόγο.

Στόχος της συγκεκριμένης πρακτικής είναι η κατανόηση από την πλευρά των μαθητών της πολυπλοκότητας που κρύβει η επικοινωνία.

Γλωσσολόγοι και καθηγητές Πανεπιστημίου εργάζονται ακατάπαυστα με μοναδικό σκοπό την ανάδειξη της ορθής χρήσης της γλώσσας και την εύρεση του καταλληλότερου τρόπου διδασκαλίας αυτής.

Ωστόσο, παρά την προσπάθεια ειδικών και εκπαιδευτικών, στη σύγχρονη βιβλιογραφία παρατηρείται μία απαξίωση από μέρους των νέων ως προς τις δεξιότητες της επικοινωνίας. Σήμερα, είναι ευρέως αποδεκτό το γεγονός ότι τα άτομα νέων ηλικιών απομακρύνονται όλο και περισσότερο από τις τυπικές μεθόδους μάθησης και εκπαίδευσης.

Εργαλεία, που παλαιότερα ήταν χρήσιμα για τον δάσκαλο και τον μαθητή, όπως για παράδειγμα η γραμματική και το συντακτικό, στις μέρες μας τείνουν ολοένα και συνεχώς να υποβαθμίζονται.

Η αιτία των παραπάνω φαινομένων δε φαίνεται να εστιάζεται σε μία μόνο πηγή.

Αρχικά, οι απαιτητικοί ρυθμοί της σύγχρονης αστικής κοινωνίας προωθούν τη σύντομή επικοινωνία, καθιστώντας την, μάλιστα, απαραίτητη.

Αποτέλεσμα όλων αυτών είναι τα νεαρά άτομα να ξεχνούν, ή πολλές φορές να μην αφομοιώνουν καν, τους συντακτικούς και γραμματικούς κανόνες που υπαγορεύει η εκάστοτε γλώσσα.

Η χρήση γραμματικο-συντακτικών κανόνων περιορίζεται αυστηρά στα πλαίσια του σχολείου.

Πρακτικά, αυτό σημαίνει, ότι οι άνθρωποι που καλούνται να απαρτίσουν τις κοινωνίες του μέλλοντος δεν γνωρίζουν πώς να γράφουν και ποιος είναι ο επίσημος ενδεδειγμένος τρόπος επικοινωνίας.

Η επικοινωνία τους και ο τρόπος τους να μοιραστούν ιδέες ακολουθεί μια νέα τάση, που θα μπορούσαμε κωμικά να περιγράψουμε με την φράση “όσο πιο απλό, τόσο πιο κατανοητό”.

Αποτέλεσμα όλων αυτών είναι ο πλούτος των γλωσσών, που αποτελούν πολιτιστική κληρονομιά των λαών, να θυσιάζεται σταδιακά στο όνομα της ευκολίας και της οικονομίας του χρόνου.

Επιπλέον, ο νέος τρόπος ζωής έχει απομακρύνει κυρίως τον νεανικό πληθυσμό από την κλασική έννοια του διαβάσματος.

Ένα καλογραμμένο βιβλίο έχει σήμερα σημαντικά μικρότερη αξία σε σύγκριση με το παρελθόν.

Τα παιδιά προτιμούν πολύ περισσότερο να ασχοληθούν με τεχνολογικά προϊόντα.

Η εικόνα και η ψηφιοποίησι των μέσων τραβούν επιτακτικά την προσοχή του ανθρώπου οποιασδήποτε ηλικίας. Ιδιαίτερα στον δυτικό πολιτισμό, η εξοικείωση με τα τεχνολογικά μέσα είναι μεγάλη και θεωρείται πια δεδομένη και απαραίτητη.

Όπως είναι φυσικό, η παραδοσιακή έννοια της μελέτης ανήκει στο παρελθόν. Νέες εκπαιδευτικές τάσεις που επωφελούνται από την απήχηση της τεχνολογίας ξεπροβάλλουν και στοχεύουν στην πρόκληση του ενδιαφέροντος των μαθητών.

Οι παρουσιάσεις με τη μορφή Power Point που χρησιμοποιούνται πλέον ευρέως σε όλες τις βαθμίδες της εκπαίδευσης είναι χαρακτηριστικό παράδειγμα του γεγονότος αυτού.

Ακολουθώντας, ο μαθητής χάνει την έννοια της συνοχής του κειμένου και συνηθίζει σε ένα πιο απλό και σχηματικό μοτίβο μελέτης, με αποτέλεσμα φυσικά να εξαλείφονται και όλες οι συντακτικές ιδιότητες της γλώσσας. Ωστόσο, η καλπάζουσα πρόοδος του είδους μας και οι αλλαγές που αυτή συνεπάγεται δεν είναι εφικτό να σταματήσουν, ακόμα και αν υπάρχουν τέτοιου είδους απώλειες.

Ο σοφός άνθρωπος, εξάλλου, δεν στέκεται ποτέ αντίθετα στις εξελίξεις.

Η τεχνολογία έχει εισβάλλει δυναμικά στις ζωές όλων μας και η παρουσία της είναι πολλά υποσχόμενη.

Κάθε στάδιο της ανθρώπινης ζωής έχει επηρεαστεί από τις τεχνολογικές προόδους. Βρέφη και παιδιά νηπιακής και προσχολικής ηλικίας εξιτάρονται στη θέα μιας οθόνης με κινούμενη εικόνα, ενώ από πολύ νωρίς παιδιά σχολικής ηλικίας γνωρίζουν να χειρίζονται άψογα τον υπολογιστή, το κινητό και άλλα τεχνολογικά εργαλεία.

Όπως αναφέρθηκε και νωρίτερα η μάθηση σε οποιαδήποτε μορφή και ηλικία αξιοποιεί τα νέα μέσα.

Το διαδίκτυο βρίθει από εφαρμογές και παιχνίδια που αναφέρονται σε άτομα σε ποικίλα στάδια ανάπτυξης.

Τα παιδιά απολαμβάνουν μέσω παιχνιδιού, μιας εύκολης, διασκεδαστικής και χαλαρωτικής διαδικασίας να έρχονται σε επαφή με τα γράμματα και τις νέες γνώσεις.

Η μάθηση παύει να νοηματοδοτείται με αρνητικούς όρους, μπορεί να λαμβάνει χώρα σε οποιοδήποτε μέρος, οποιαδήποτε στιγμή της ημέρας και να βοηθάει τα νεαρά άτομα να περνούν την ώρα τους εποικοδομητικά.

Κάποιες εφαρμογές, επιπλέον, παρέχουν την επιλογή της συνεργατικής ενασχόλησης με αντικείμενα μάθησης, μέσω της οποίας προωθείται το συνεργατικό πνεύμα και αναπτύσσονται οι αλληλοδιδασκτικές ικανότητες. Εξάλλου, είναι χαρακτηριστικό το παράδειγμα των Βόρειων Ευρωπαϊκών χωρών, που έχουν εντάξει ήδη στο εκπαιδευτικό τους σύστημα τη χρήση Ηλεκτρονικών Υπολογιστών και άλλων ψηφιακών μέσων.

Τα μέσα αυτά βοηθούν το δάσκαλο να οπτικοποιεί τα δεδομένα, να διευκολύνει τη διδασκαλία και να διατηρεί το ενδιαφέρον των μαθητών αμείωτο.

Ωστόσο, δεν θα ήταν σωστό να παραλείψουμε να τονίσουμε ότι, όπως κάθε θετική κατάσταση κρύβει και κάποια αρνητική χροιά, έτσι και εδώ η χρήση των τεχνολογικών επιτευγμάτων ενέχει τον κίνδυνο του εθισμού.

Πάντα με την κατάλληλη προστασία και επιτήρηση τα παιδιά, ιδιαίτερα αυτά των μικρότερων ηλικιών, είναι απαραίτητο να αναπτύξουν σταδιακά ενός είδους αυτοκυριαρχία και να κατανοήσουν ότι τα τεχνολογικά μέσα δεν είναι άλλο παρά βοηθητικά εργαλείο στη διαδικασία.

Η ιδέα ότι η μάθηση περνά απαραίτητα μέσα από το σχολείο, ο δάσκαλος είναι ο μόνος που καταφέρνει να μεταδώσει πέρα από γνώσεις και μια σειρά από απαραίτητες αρχές και ότι το βιβλίο παραμένει ένας πιστός σύντροφος είναι κεντρικής σημασίας.

Ως εκ τούτου, καθίσταται προφανές το γεγονός ότι η γραφική παράσταση του κλάδου της Επεξεργασίας Φυσικής Γλώσσας και των εφαρμογών αυτής στη ζωή μας όχι μόνο θα βρίσκεται πάνω από τον άξονα x' , αλλά θα έχει και γνησίως αύξουσα μονοτονία. Σκοπός του προαναφερθέντα μαθηματικού παραλληλισμού είναι η ανάδειξη του γεγονότος, ότι η Επεξεργασίας Φυσικής Γλώσσας έχει να προσφέρει συντριπτικά περισσότερα θετικά από ότι αρνητικά στο σύγχρονο ψηφιακό κόσμο.

Θέμα 1

Στη παρούσα ενότητα θα ασχοληθούμε με το 1^ο από τα 7 – ζητούμενα – θέματα της εργασίας.

1. Εκφώνηση

Θέμα 1ο (20 μονάδες):

Αναζητείστε στο διαδίκτυο ένα σύγχρονο θέμα επεξεργασίας φυσικής γλώσσας που σας έκανε εντύπωση, ή σας κίνησε το ενδιαφέρον από τις διαλέξεις του προσκεκλημένου ομιλητή κατά την διάρκεια του μαθήματος. Αναπτύξτε το θέμα ή την εφαρμογή σε 10 το πολύ σελίδες.

2. Εισαγωγή

Τα σύγχρονα συστήματα Επεξεργασίας Φυσικής Γλώσσας πρέπει να διαθέτουν σημαντική γνώση για την δομή της ίδιας της γλώσσας, τις λέξεις της, του συνδυασμού των λέξεων σε προτάσεις, των εννοιών των λέξεων, το πώς συμμετέχουν οι λέξεις στη σημασία της πρότασης, κοκ.

Τα συστήματα αυτά απαιτούν την ύπαρξη μεθόδων κωδικοποίησης και χρήσης της γνώσης που να παράγουν τις κατάλληλες συμπεριφορές.

Επίσης, θα πρέπει να λαμβάνεται υπόψη το πλαίσιο συμφραζομένων της Γλώσσας πάνω στο οποίο γίνεται η Επεξεργασία.

Ιδανικά, τα εργαλεία Επεξεργασίας Φυσικής Γλώσσας θα πρέπει να είναι σε θέση να επεξεργάζονται δομημένες και αδόμητες προτάσεις με απλά ερωτήματα, παράγοντας μία ενιαία, συνδυαστική απάντηση, ουδέτερη προς τα δεδομένα.

Η ανάγκη δημιουργίας καινοτόμων διαδικτυακών εφαρμογών που συνδυάζουν την Επεξεργασία Φυσικής Γλώσσας με διαδικτυακές υπηρεσίες, προκειμένου να εξυπηρετείται ευκολότερα ο χρήστης, έχει οδηγήσει την ερευνητική κοινότητα στην μελέτη και παραγωγή πρότυπων μεθόδων και εργαλείων στον τομέα Ανάλυσης και Επεξεργασίας Φυσικής Γλώσσας.

Η μελέτη, ανάλυση, και εξαγωγή συμπερασμάτων από τις σύγχρονες αυτές μεθόδους Επεξεργασίας της Φυσικής Γλώσσας ως διεπαφές επικοινωνίας χρήστη-υπολογιστή είναι ένας από τους μεγαλύτερους πόλους έλξης για κάθε επιστήμονα των υπολογιστών (Computer Scientist) ως προς αυτή.

3. Παρουσίαση Ιδέας

Σε αυτή την υπό ενότητα θα γίνει παρουσίαση του σύγχρονου θέματος Επεξεργασίας Φυσικής Γλώσσας που προσωπικά μου έκανε εντύπωση, ιδιαίτερα μετά τις παρουσιάσεις του κ. Περήφανου.

Μετά από πειραματισμό και έρευνα επι του θέματος, σας παραθέτω στα αποσταλμένα αρχεία, ένα ενδεικτικό που υλοποιεί το σύγχρονο θέμα που θα περιγράφει παρακάτω (*Το αρχείο θα βρίσκεται στο φάκελο με τίτλο Subject 1 και θα είναι συμπιεσμένο με όνομα NLP-Sentiment-Analysis.zip*).

3.1. Εισαγωγικό Πλαίσιο Ιδέας

Ένα από τα πλέον σύγχρονα θέματα της Επεξεργασίας Φυσικής Γλώσσας (Natural Processing Language) είναι αυτό της κατηγοριοποίησης των διάφορων κειμένων – συνήθως κριτικών και αξιολογήσεων - ως θετικά ή αρνητικά.

Η ταξινόμηση των κειμένων σε αυτές τις δύο κατηγορίες είναι πολύ σημαντική καθώς, με αυτό το τρόπο μπορούν μεγάλες πλατφόρμες ή μέσα κοινωνικής δικτύωσης να τα χρησιμοποιήσουν με κατάλληλο τρόπο, προκειμένου να παραχθούν χρήσιμα συμπεράσματα (ακόμα και ανάλυση των δεδομένων).

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.

Όπως μας είχε παρουσιαστεί από τον κ. Περήφανο, η ανάλυση των κριτικών των ταινιών, τις οποίες παραθέτουν οι χρήστες που έχουν δει την ταινία, είναι καίριας σημασίας για ένα σαιτ όπως το IMDb.

Για την αντιμετώπιση του συγκεκριμένου θέματος μπορούμε να φτιάξουμε ένα μοντέλο πρόβλεψης.

Πιο συγκεκριμένα, ο πλέον ενδεδειγμένος τρόπος προσέγγισης είναι αυτός μέσω των νευρωνικών δικτύων και word embeddings.

3.2. Παρουσίαση Ιδέας Υλοποίησης

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.

Για την υλοποίηση του κώδικα μπορείτε να δείτε το ενδεικτικό αρχείο που αναφέρθηκε παραπάνω και βρίσκεται στο φάκελο του θέματος 1 (Το αρχείο θα βρίσκεται στο φάκελο με τίτλο *Subject 1* και θα είναι συμπιεσμένο με όνομα *NLP-Sentiment-Analysis.zip*).

3.3. Εκπαίδευση Μοντέλου – Train Classifiers

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.

Το πρόγραμμα που υλοποιήσαμε (όπως είχε υποδείξει και ο κ. Περήφανος), εμπεριέχε τη διαδικασία εκπαίδευσης του μοντέλου μέσα από κριτικές (Train Model's Classifiers).

3.4. Κωδικοποίηση Κειμένου – Κριτικής

Σε αυτό το σημείο της διαδικασίας, δεν μας ενδιαφέρει το σημασιολογικό περιεχόμενο του κειμένου, δηλαδή δεν θα πραγματοποιήσουμε κάποια σημασιολογική ανάλυση.

Ακόμη, 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 500 λέξεων θα βρίσκαμε τις θετικές και αρνητικές λέξεις του κειμένου και ανάλογα το πρόσημο, δηλαδή την μεταξύ τους διαφορά, θα εξαγάγαμε το συμπέρασμά μας – από το ήδη εκπαιδευμένο μοντέλο 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 (δηλαδή 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.

3.5. Αναπαράσταση Λέξεων Μέσω Διανυσμάτων

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.

Ειδικότερα, αφού σκανάρουμε όλα τα διαθέσιμα κείμενα, βρίσκουμε τις μοναδικές εμφανίσεις της κάθε λέξης και τις αποθηκεύουμε σε μια λίστα - λεξικό (dictionary), με διαστάσεις n , 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", "cute", "cat"], το διάνυσμα που αντιστοιχεί:

- στη λέξη "the" είναι το [1,0,0],
- στη λέξη "cute" είναι το [0,1,0] και
- στη λέξη "cat" 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 οι κατασκευαστές.

3.6. Ομοιότητα Λέξεων – Διανυσμάτων

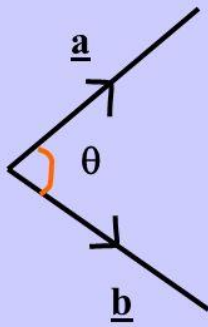
Σημαντικό βήμα είναι η αξιοποίηση των διανυσμάτων που ορίσαμε και δημιουργήσαμε παραπάνω, είναι να είμαστε σε θέση να συγκρίνουμε πόσο δύο διανύσματα στον χώρο είναι σχετικά κοντά, δηλαδή σημασιολογικά συγγενικά.

Επειδή, δύναται να έχουμε διανύσματα με τελείως διαφορετικά μήκη δεν θα χρησιμοποιήσουμε το εσωτερικό γινόμενο όπως συνηθίζεται, αλλά τη γωνία συνημίτονου.

Ακολουθεί σχετική εικόνα:

Angle Between Vectors

Given that $\underline{a} \cdot \underline{b} = |\underline{a}| |\underline{b}| \cos\theta$



then it must follow that

$$\cos\theta = \frac{\underline{a} \cdot \underline{b}}{|\underline{a}| |\underline{b}|}$$

... and this allows us to find the angle between \underline{a} & \underline{b} .

Εικόνα 1 : Υπολογισμός γωνίας συνημίτονου.

3.7. Στατιστικό γλωσσικό μοντέλο

Με τον όρο αυτό αναφερόμαστε σε μια κατανομή, είναι δηλαδή μια ακολουθία κατανομής λέξεων.

Αυτό που μας ενδιαφέρει εμάς, 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. Αναλυτικότερα, για την πρώτη εφαρμογή, για μια λέξη που δεν είναι στο λεξικό, αναζητούνται κάποια μέτρα απόστασης (διανυσμάτων), 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 (high complexity level).

3.8. Μοντέλο word2vec

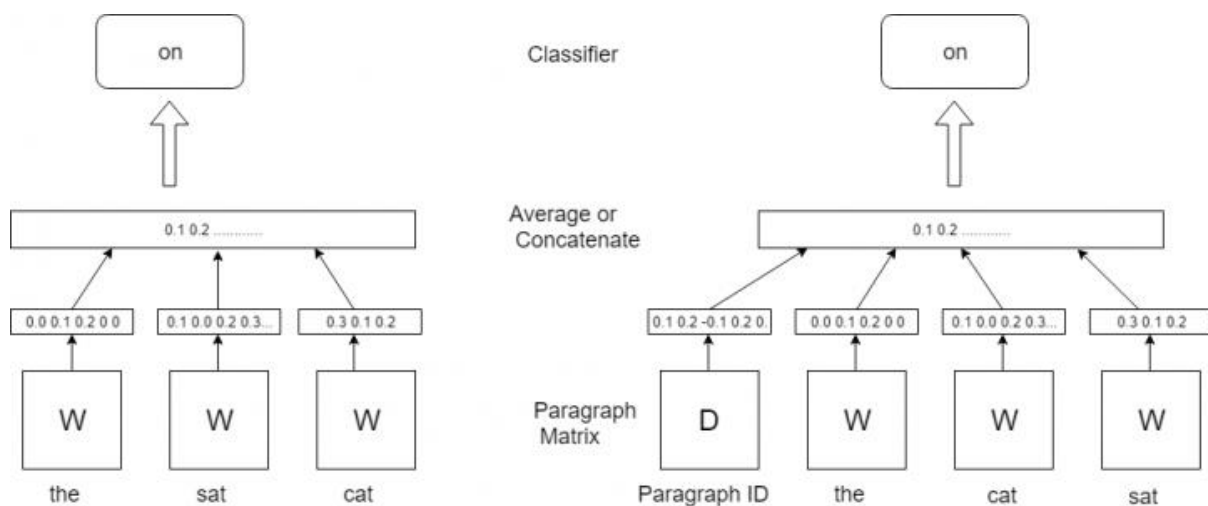
Για την αντιμετώπιση του παραπάνω προβλήματος υιοθετήθηκε η πρόταση του Mikolov, γνωστό και ως το μοντέλο word2vec.

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.

Το μοντέλο αυτό για να είναι λειτουργικό και αξιόπιστο θα πρέπει αφού το δημιουργήσουμε, να το εκπαιδεύσουμε (Train Classifiers).

Για την επίτευξη του στόχου μας η Python προσφέρει μια σειρά από βιβλιοθήκες και συναρτήσεις, οι οποίες συμβάλλουν σε αυτό.

Ακολουθεί ενδεικτική επεξηγηματική εικόνα:



Εικόνα 2 : Επεξηγηματική εικόνα.

3.9. Κατηγοριοποίηση Κειμένου – Κριτικής

Έχοντας στην φαρέτρα μας όλα τα παραπάνω εργαλεία που μας προσφέρει απλόχερα η γλώσσα της Python, είμαστε στη πλεονεκτική θέση να τα συνδυάσουμε, ώστε να κατηγοριοποιήσουμε ένα νέο-άγνωστο κείμενο (μια κριτική) σε μία ευρύτερη κατηγορία.

Αρχικά, κάθε κείμενο θα αναπαρίσταται ως ο μέσος όρος των διανυσμάτων των λέξεων του συγκεκριμένου κειμένου.

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 (εκπαίδευση μοντέλου) και
- 25% για τον έλεγχο.

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 μηχανικό μοντέλο.

3.10. Συμπέρασμα Υλοποίησης

Σύμφωνα με όλα τα παραπάνω, αντιλαμβανόμαστε ότι αυτή η διαδικασία προϋποθέτει την σωστή εκπαίδευση του μοντέλου μας, εισάγοντάς του δεδομένα – κριτικές, τα οποία είναι ήδη προσημειωμένα.

Επιπλέον, 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 και μεγάλο όραμα (potential) για πιθανές υλοποιήσεις στο μέλλον.

Θέμα 2

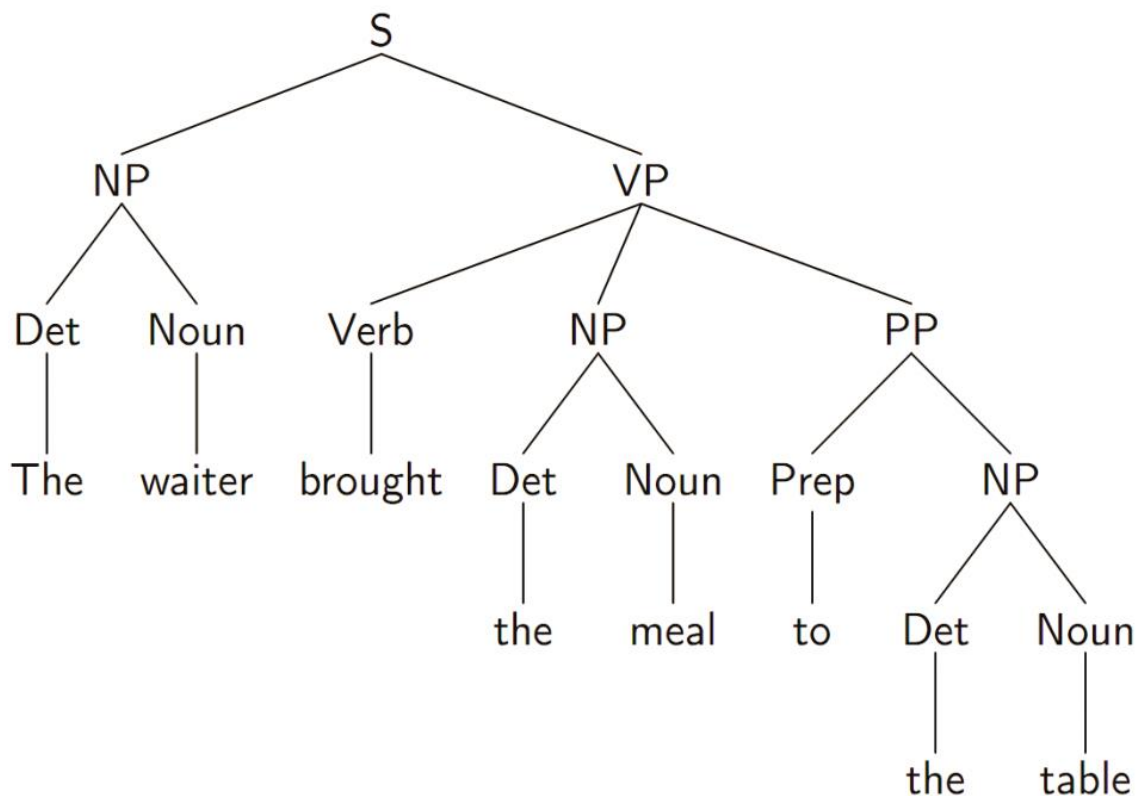
Στη παρούσα ενότητα θα ασχοληθούμε με το 2^ο θέμα της εργασίας.

Εκφώνηση

Θέμα 2ο (20 μονάδες):

(α) Με ποια γραμματική σε μορφή DCG μπορούμε να αναγνωρίσουμε την πρόταση : [the, waiter, brought, the, meal, to, the, table], σύμφωνα με το παρακάτω σχήμα;

(β) Με ποια γραμματική σε μορφή DCG μπορούμε να παράγουμε σε μορφή **functor** το συντακτικό δένδρο για την αναγνώριση της πρότασης : [the, waiter, brought, the, meal, to, the, table], σύμφωνα με το παρακάτω σχήμα ;



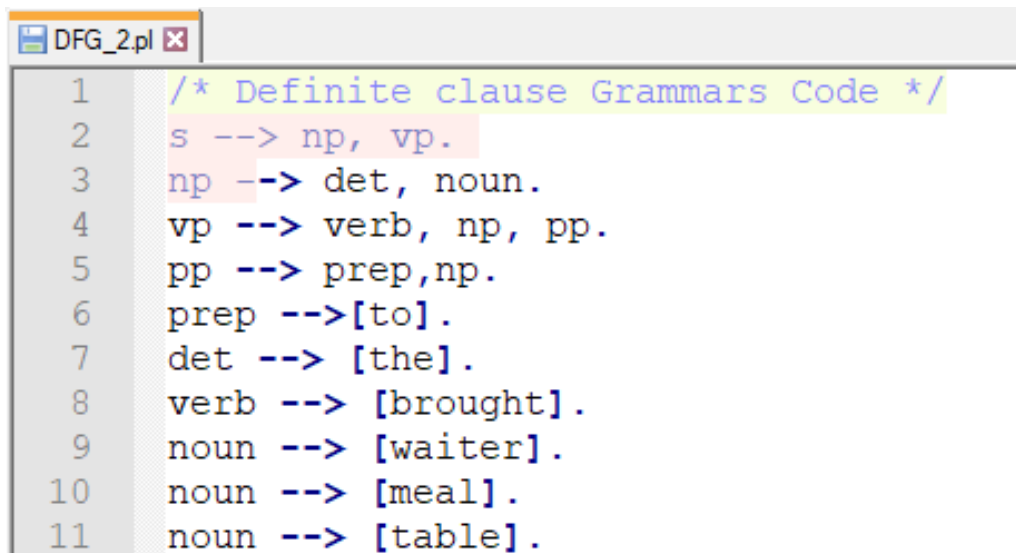
Εικόνα 3 : Συντακτικό Δέντρο Εκφώνησης.

Ερώτημα 1^ο

Προκειμένου να αναγνωρίσουμε την πρόταση της εκφώνησης :
[the, waiter, brought, the, meal, to, the, table],

σύμφωνα με το παραπάνω σχήμα της εκφώνησης (Εικόνα 3), θα πρέπει να χρησιμοποιήσουμε την ακόλουθη γραμματική, η οποία είναι σε μορφή DCG (Defining Clause Grammars).

Η συγκεκριμένη γραμματική αναγνώριση της ζητούμενης πρότασης υπάρχει και στο αρχείο 'DFG_2.pl' στον φάκελο Subject 2.



```
1  /* Definite clause Grammars Code */
2  s --> np, vp.
3  np --> det, noun.
4  vp --> verb, np, pp.
5  pp --> prep, np.
6  prep --> [to].
7  det --> [the].
8  verb --> [brought].
9  noun --> [waiter].
10 noun --> [meal].
11 noun --> [table].
```

Εικόνα 4 : Κώδικας της γραμματικής αναγνώρισης της πρότασης.

Συνεπώς, παρατηρούμε ότι μια πρόταση χωρίζεται σε μία noun_phrase και μια verb_phrase, οι οποίες στη συνέχεια θα αναλυθούν.

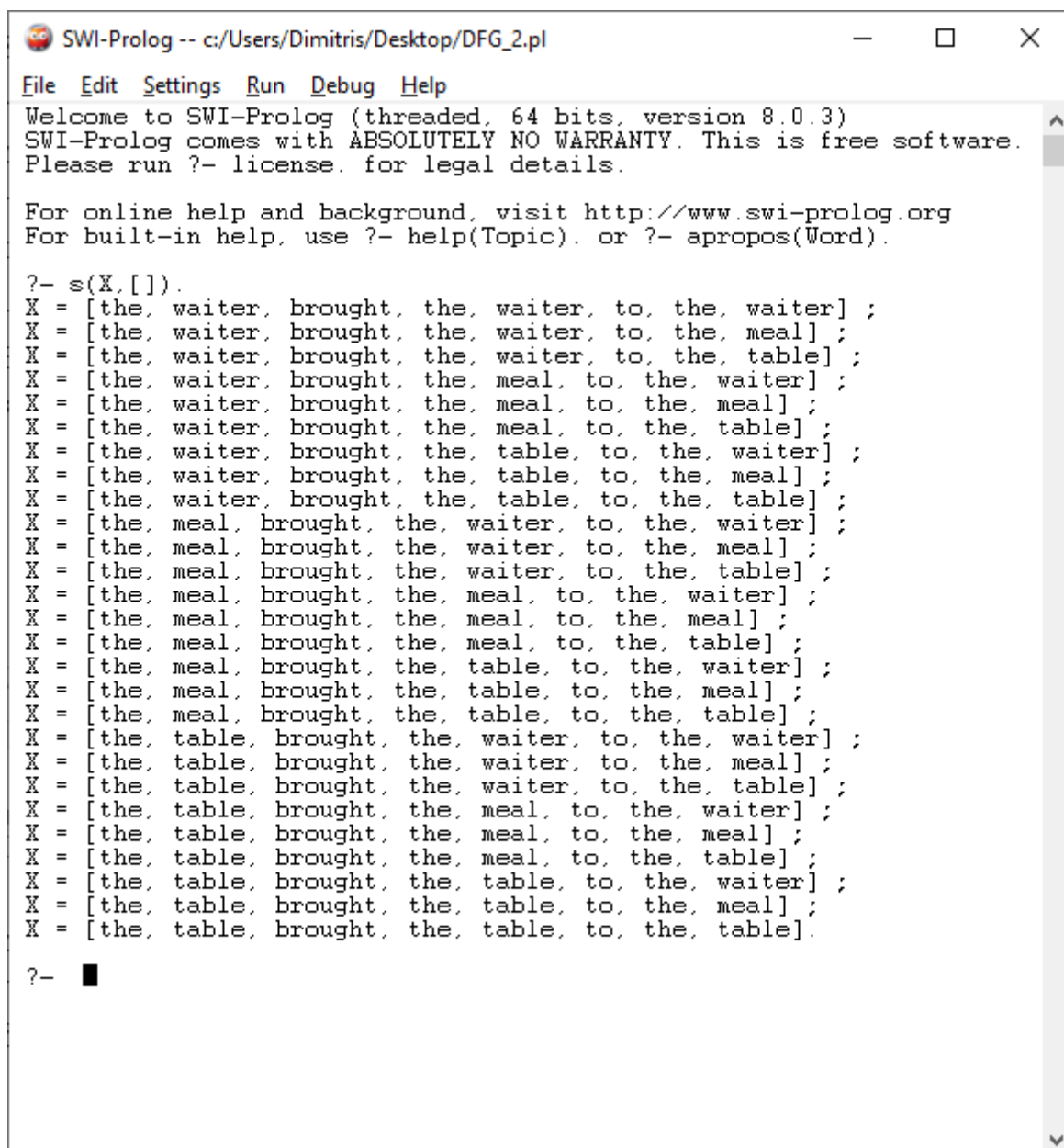
Αυτό που φαίνεται στον παραπάνω κώδικα είναι η ανάλυση μιας noun_phrase σε det και noun, ενώ μια verb_phrase σε verb, noun_phrase και preposition_phrase (κανόνες της γραμματικής).

Η 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 ή false.

Παρακάτω παρατίθενται ενδεικτικές εικόνες για amφότερους τους τρόπους:



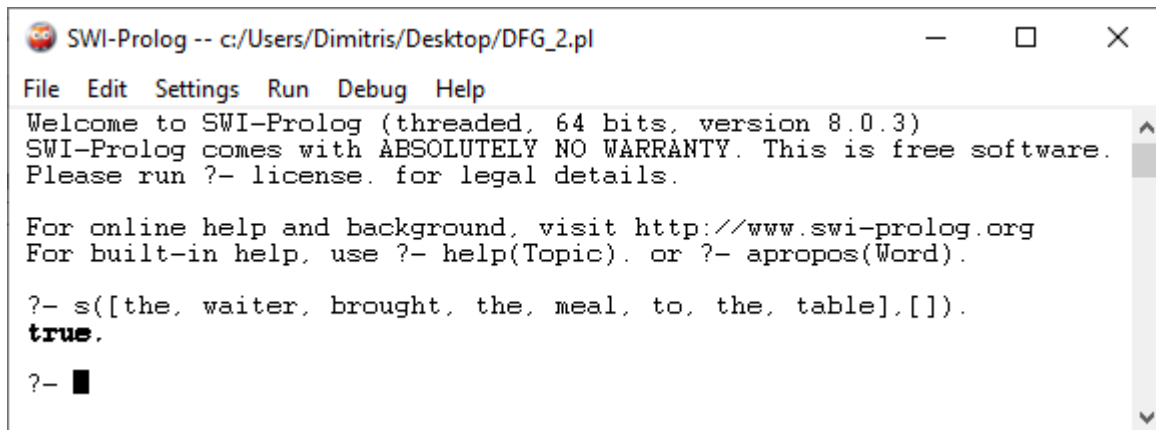
```
SWI-Prolog -- c:/Users/Dimitris/Desktop/DFG_2.pl
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.0.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- s(X, []).
X = [the, waiter, brought, the, waiter, to, the, waiter] ;
X = [the, waiter, brought, the, waiter, to, the, meal] ;
X = [the, waiter, brought, the, waiter, to, the, table] ;
X = [the, waiter, brought, the, meal, to, the, waiter] ;
X = [the, waiter, brought, the, meal, to, the, meal] ;
X = [the, waiter, brought, the, meal, to, the, table] ;
X = [the, waiter, brought, the, table, to, the, waiter] ;
X = [the, waiter, brought, the, table, to, the, meal] ;
X = [the, waiter, brought, the, table, to, the, table] ;
X = [the, meal, brought, the, waiter, to, the, waiter] ;
X = [the, meal, brought, the, waiter, to, the, meal] ;
X = [the, meal, brought, the, waiter, to, the, table] ;
X = [the, meal, brought, the, meal, to, the, waiter] ;
X = [the, meal, brought, the, meal, to, the, meal] ;
X = [the, meal, brought, the, meal, to, the, table] ;
X = [the, meal, brought, the, table, to, the, waiter] ;
X = [the, meal, brought, the, table, to, the, meal] ;
X = [the, meal, brought, the, table, to, the, table] ;
X = [the, table, brought, the, waiter, to, the, waiter] ;
X = [the, table, brought, the, waiter, to, the, meal] ;
X = [the, table, brought, the, waiter, to, the, table] ;
X = [the, table, brought, the, meal, to, the, waiter] ;
X = [the, table, brought, the, meal, to, the, meal] ;
X = [the, table, brought, the, meal, to, the, table] ;
X = [the, table, brought, the, table, to, the, waiter] ;
X = [the, table, brought, the, table, to, the, meal] ;
X = [the, table, brought, the, table, to, the, table].

?-
```

Εικόνα 5 : Εμφάνιση όλων των δυνατών συνδυασμών προτάσεων.



```
SWI-Prolog -- c:/Users/Dimitris/Desktop/DFG_2.pl
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.0.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- s([the, waiter, brought, the, meal, to, the, table], []).
true.
?- 
```

Εικόνα 6 : Ερώτηση True ή False.

Σχετικά με τους τρόπους αναζήτησης στη γλώσσα της Prolog, θα ήταν δόκιμο να αναφερθεί το γεγονός ότι:

Ο μηχανισμός αναζήτησης της Prolog ελέγχει αν ένα γεγονός είναι αληθές ή ψευδές, ή παράγει όλες τις λύσεις που το κάνουν αληθές.

Στην περίπτωση της DCG το πρόγραμμα είτε παράγει όλες τις προτάσεις που γίνονται αποδεκτές από το γραμματική είτε, αν δοθεί πρόταση, αποφασίζει αν είναι συντακτικά αποδεκτή.

Ερώτημα 2^ο

Στη Prolog έχουμε τη δυνατότητα να παράγουμε συντακτικά δέντρα που είναι ισοδύναμα με αυτό της εικόνας της εκφώνησης της άσκησης, στην μορφή **functor**.

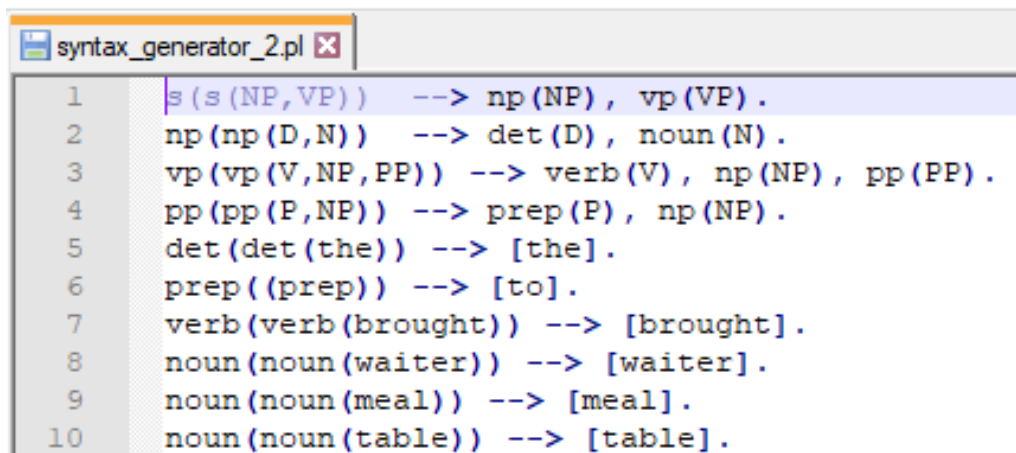
Προκειμένου, να επιτευχθεί αυτό θα πρέπει να αλλάξουμε την παραπάνω γραμματική με τρόπο κατάλληλο ώστε να εμφανίζεται ορθώς το συντακτικό δέντρο σε μορφή functor.

Η συγκεκριμένη γραμματική υπάρχει και στο αρχείο 'syntax_generator_2.pl' στον φάκελο Subject 2.

Ειδικότερα, η φιλοσοφία της ανάλυσης της κάθε φράσης θα είναι ίδια με πριν.

Οι αλλαγές που έγιναν σχετίζονται μόνο το κομμάτι της εμφάνισης του συντακτικού δέντρου σε μορφή functor.

Ακολουθεί σχετική εικόνα από τον κώδικα και τα αποτελέσματα αυτού:



```
1 s(s(NP,VP)) --> np(NP), vp(VP).
2 np(np(D,N)) --> det(D), noun(N).
3 vp(vp(V,NP,PP)) --> verb(V), np(NP), pp(PP).
4 pp(pp(P,NP)) --> prep(P), np(NP).
5 det(det(the)) --> [the].
6 prep(prepare) --> [to].
7 verb(verb(brought)) --> [brought].
8 noun(noun(waiter)) --> [waiter].
9 noun(noun(meal)) --> [meal].
10 noun(noun(table)) --> [table].
```

Εικόνα 7 : Κώδικας της γραμματικής.

```
?- s(S,[the,waiter,brought,the,meal,to,the,table],[]).
S = s(np(det(the), noun(waiter)), vp(verb(brought), np(det(the), noun(meal)), pp(prepare, np(det(the), noun(table)))))
```

Εικόνα 8 : Εμφάνιση συντακτικού δέντρου σε μορφή functor.

Στην παραπάνω εικόνα των αποτελεσμάτων φαίνεται το συντακτικό δέντρο σε μορφή functor, για την αναγνώριση της ζητούμενης – από την άσκηση - πρότασης [the, waiter, brought, the, meal, to, the, table].

Θέμα 3

Στη παρούσα ενότητα θα ασχοληθούμε με το 3^ο θέμα της εργασίας.

Εκφώνηση

Θέμα 3ο (20 μονάδες):

Το παρακάτω πρόγραμμα αναγνωρίζει και υπολογίζει αριθμητικές εκφράσεις όπως αναλύθηκε στο θεωρητικό μέρος. Να αναπτυχθεί ένα αντίστοιχο πρόγραμμα όπου οι αριθμοί είναι δυαδικοί και οι αριθμητικές εκφράσεις είναι αντίστοιχα αριθμητικές εκφράσεις δυαδικών αριθμών.

```
expression(Value) --> number(Value).
expression(Value) --> number(X), [+], expression(V), {Value is X+V}.
expression(Value) --> number(X), [-], expression(V), {Value is X-V}.
expression(Value) --> number(X), [*], expression(V), {Value is X*V}.
expression(Value) --> number(X), [/], expression(V), {V/=0, Value is X/V}.
expression(Value) --> left_parenthesis, expression(Value), right_parenthesis.
left_parenthesis --> ['('].
right_parenthesis --> [')'].
number(X) --> digit(X).
number(Value) --> digit(X), number(Y), {numberofdigits(Y,N), Value is X*10^N+Y}.
digit(0) --> [0].
digit(1) --> [1].
digit(2) --> [2].
digit(3) --> [3].
digit(4) --> [4].
digit(5) --> [5].
digit(6) --> [6].
digit(7) --> [7].
digit(8) --> [8].
digit(9) --> [9].
numberofdigits(Y,1) :- Z is Y/10, Z<1.
numberofdigits(Y,N) :-
Z is (Y - mod(Y,10))/10,
numberofdigits(Z,N1),
N is N1+1
```

Εισαγωγικό Πλαίσιο

Σκοπός της συγκεκριμένης άσκησης είναι η υλοποίηση ενός προγράμματος, το οποίο θα αναγνωρίζει και θα υπολογίζει αριθμητικές εκφράσεις των δυαδικών αριθμών.

Το συγκεκριμένο πρόγραμμα θα πρέπει να λειτουργεί στα πρότυπα του παραδοθέντος από εσάς αρχείου αλλά για δυαδικούς αριθμούς.

Στο συγκεκριμένο πρόγραμμα αναγνωρίζονται αριθμητικές εκφράσεις όπως αυτή της πρόσθεσης, της αφαίρεσης και του πολλαπλασιασμού των δυαδικών αριθμών.

Είναι γνωστό ότι τα μόνα ψηφία που αναγνωρίζονται είναι το 0 και το 1. Να σημειωθεί ότι το κατηγορημα για την εκτέλεση της αριθμητικής έκφρασης είναι το `expression` και αυτό για την αναγνώριση ο κανόνας `recognize`.

Η συγκεκριμένη γραμματική αναγνώριση της ζητούμενης πρότασης υπάρχει και στο αρχείο `'parse_and_evaluate2.pl'` στον φάκελο Subject 3.

Αρχικοποίηση Υλοποίησης

Αρχικά για να δηλώσουμε τα ψηφία μας θα πρέπει να γράψουμε τον κανόνα `digit` με τις δύο αρχικές περιπτώσεις:

`digit --> [0].`

`digit --> [1].`

Είναι δεδομένο ότι για να μπορέσουμε να εκτελέσουμε και τις πράξεις μεταξύ των δυαδικών αριθμών (μαθηματικές εκφράσεις), θα πρέπει να προσδώσουμε σημασιολογία στο μέρος του προγράμματος για αριθμούς.

Παρακάτω αναλύουμε τον κώδικα:

- **`number(X) --> digit(X).`**

Κάθε αριθμός που έχει ένα ψηφίο, η αξία του είναι αυτή του ψηφίου (ο αρχικός κανόνας, παραπάνω).

- **number(Value)-->digit(X), number(Y), {numberofdigits(Y,N), Value is $X*10^N+Y$ }.**

Όταν ο αριθμός A με X ψηφία, έχει ένα ψηφίο παραπάνω από ένα αριθμό B και ταυτόχρονα ο αριθμός B έχει N ψηφία, τότε ο A έχει αξία :
 $A = X*2^N + B$ (Σχέση 1)

Για παράδειγμα ο 153 έχει ένα ψηφίο, το 1, παραπάνω από το 53, και ο 53 έχει 2 ψηφία, η αξία του 965 είναι $9*2^2+65$.

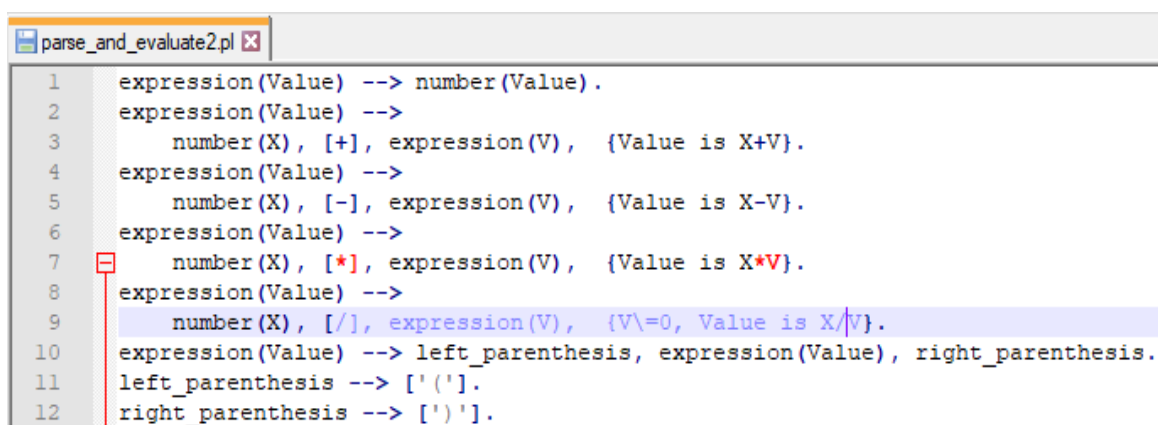
- **numberofdigits(Y,1) :- Z is Y/2, Z<1.**
numberofdigits(Y,N) :- Z is (Y - mod(Y,2))/2, numberofdigits(Z,N1),
N is N1+1.

Το συγκεκριμένο κομμάτι κώδικα, όταν το εκτελούμε έχει σαν αποτέλεσμα το παρακάτω:

?- number(V,[4,5,6],[]).

V = 456 .

Επόμενο βήμα είναι η προσθήκη σημασιολογίας και για το υπόλοιπο μέρος του κώδικα. Να σημειωθεί ότι η σημασιολογία στη Prolog δηλώνεται μέσα σε άγκιστρα.



```
1 expression(Value) --> number(Value).
2 expression(Value) -->
3   number(X), [+], expression(V), {Value is X+V}.
4 expression(Value) -->
5   number(X), [-], expression(V), {Value is X-V}.
6 expression(Value) -->
7   number(X), [*, expression(V), {Value is X*V}.
8 expression(Value) -->
9   number(X), [/], expression(V), {V\=0, Value is X/V}.
10 expression(Value) --> left_parenthesis, expression(Value), right_parenthesis.
11 left_parenthesis --> ['('].
12 right_parenthesis --> [')'].
```

Εικόνα 9 : Προσθήκη μαθηματικής σημασιολογίας.

Σαν παραδοχή σε αυτή την εργασία έχουμε υποθέσει την εξής:

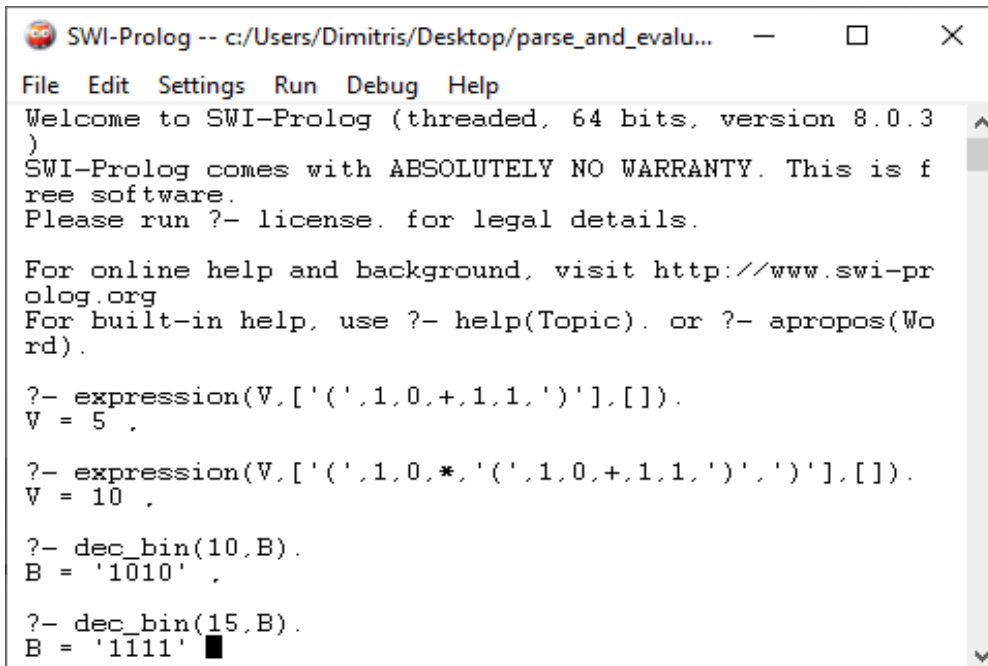
Το αποτέλεσμα των εκφράσεων θα είναι σε δεκαδική αναπαράσταση, καθώς, θεωρήθηκε.

Ωστόσο, αν ο χρήστης επιθυμεί την μετατροπή του αποτελέσματος σε δυαδική αναπαράσταση θα πρέπει να τρέξει τον κανόνα `dec_bin`.

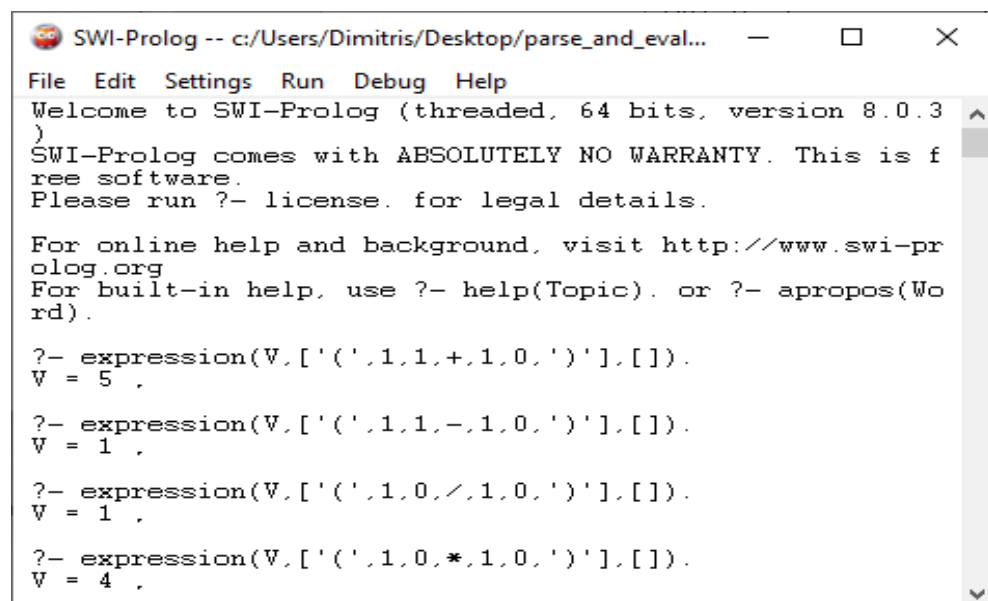
```
dec_bin(0,'0').  
dec_bin(1,'1').  
dec_bin(N,B):-N>1,X is N mod 2,Y is N//2,dec_bin(Y,B1),atom_concat(B1,X,B).
```

Εικόνα 10 : Κώδικας για την μετατροπή δεκαδικού σε δυαδικό.

Ο συγκεκριμένος κανόνας δέχεται δύο ορίσματα. Το πρώτο είναι ο αριθμός που επιθυμούμε να μετατρέψουμε (ο δεκαδικός) και το δεύτερο η μεταβλητή στην οποία θα αποθηκευτεί το αποτέλεσμα (ο αντίστοιχος δυαδικός).



Εικόνα 11 : Αποτελέσματα υλοποίησης 1.



Εικόνα 12 : Αποτελέσματα υλοποίησης 2.

Θέμα 4

Στη παρούσα ενότητα θα πραγματευτούμε το 4^ο από τα 7 – ζητούμενα - θέματα της εργασίας.

1. Εκφώνηση

Θέμα 4ο (20 μονάδες):

Ανάπτυξη Λεκτικού Αναλυτή σε άλλη γλώσσα Προγραμματισμού. Αναζητείστε στο διαδίκτυο ή αναπτύξτε εσείς Λεκτικό Αναλυτή που διαβάζει μια μικρή ιστορία και μπορεί να παράγει μια λίστα από προτάσεις, κάθε μια από τις οποίες περιέχει μια λίστα από λέξεις. Τεκμηριώστε πειστικά τον κώδικά σας.

2. Εισαγωγικό Πλαίσιο

Σκοπός αυτής της άσκησης είναι η ανάπτυξη ενός Λεκτικού Αναλυτή σε άλλη γλώσσα προγραμματισμού (εκτός της Prolog), 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.

Η υλοποίηση της συγκεκριμένης άσκησης έγινε στη 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 διαβαστεί προκειμένου να υποστεί την συγκεκριμένη διαδικασία, δηλαδή τον χωρισμό σε προτάσεις.

3. Επεξήγηση Κώδικα

Σε αυτήν την ενότητα θα γίνει επεξήγηση του κώδικα της υλοποιημένης εφαρμογής.

Το προαναφερθέν αρχείο βρίσκεται στο φάκελο Subject 4 με όνομα τίτλου lexical_analyzer.py .

3.1. Διαδικασία sentence_split()

Όπως προαναφέρθηκε και παραπάνω το πρόγραμμα έχει υλοποιηθεί σε Python.

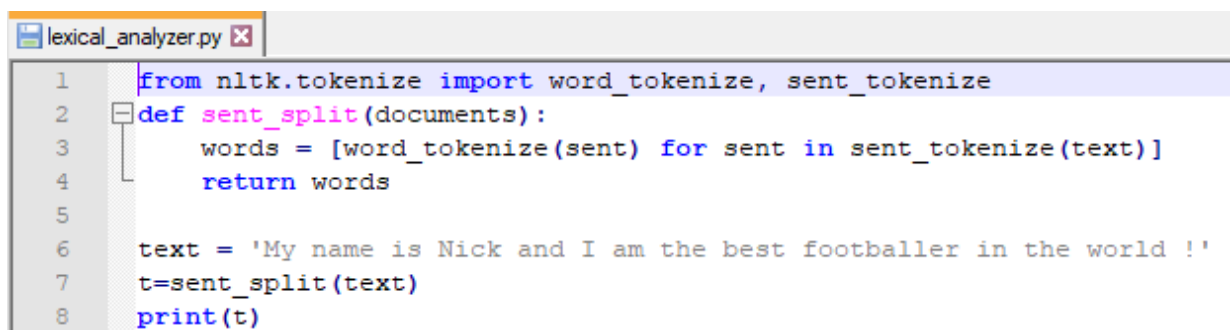
Η κύρια συνάρτηση, η οποία και επιτυγχάνει την δημιουργία λιστών από προτάσεις, όπου η καθεμία περιέχει μια λίστα από λέξεις, είναι η sentence_split().

Η συγκεκριμένη συνάρτηση έχει σαν όρισμα το κείμενο, το οποίο θα υποστεί την παραπάνω διαδικασία.

Αρχικά, για κάθε μια πρόταση θα εφαρμοστεί η μέθοδος tokenize(sent_tokenize).

Στη συνέχεια για κάθε λέξη της πρότασης που βρίσκεται στην πρόταση, εφαρμόζεται και για αυτή η μέθοδος tokenize(word_tokenize).

Οι λέξεις αυτές αποθηκεύονται σε μία λίστα, τη λίστα word.



```
lexical_analyzer.py x
1 from nltk.tokenize import word_tokenize, sent_tokenize
2 def sent_split(documents):
3     words = [word_tokenize(sent) for sent in sent_tokenize(text)]
4     return words
5
6 text = 'My name is Nick and I am the best footballer in the world !'
7 t=sent_split(text)
8 print(t)
```

Εικόνα 13 : Εικόνα κώδικα.

3.2. Εκτέλεση Υλοποιήσιμου

Για την εκτέλεση της διαδικασίας, αφού πρώτα έχει ανοιχτεί και διαβαστεί το αρχείο με την βοήθεια των συναρτήσεων `.open()` και `.read()`.

Αναλογικά, θα κληθεί η συνάρτηση `sentence_split`, με όρισμα το κείμενο που έχει διαβαστεί από το επιλεγμένο αρχείο.

4. Αποτελέσματα Εφαρμογής

Σε αυτή την ενότητα θα αναφερθούμε στα αποτελέσματα της εφαρμογής μας.

Στην παρακάτω εικόνα βλέπουμε το αποτέλεσμα της εκτέλεσης του κώδικα το οποίο περιλαμβάνει το εξής κείμενο:

My name is Nick and I am the best footballer in the world !

```
= RESTART: C:\Users\Dimitris\Desktop\Εργασία Επεξεργασία Φυσικής Γλώσσας\Subject 4\lexical_analyzer.py  
[['My', 'name', 'is', 'Nick', 'and', 'I', 'am', 'the', 'best', 'footballer', 'in', 'the', 'world', '!']]
```

Εικόνα 14 : Αποτέλεσμα εκτέλεσης προγράμματος 1.

the dog needs food. the cat has the food. the dog hates the cat. the dog chased the cat. the cat is scary.

```
= RESTART: C:\Users\Dimitris\Desktop\Εργασία Επεξεργασία Φυσικής Γλώσσας\Subject 4\lexical_analyzer.py  
[['the', 'dog', 'needs', 'food', '.'], ['the', 'cat', 'has', 'the', 'food', '.'], ['the', 'dog', 'hates', 'the', 'cat', '.'], ['the', 'dog', 'chased', 'the', 'cat', '.'], ['the', 'cat', 'is', 'scary', '.']]
```

Εικόνα 15 : Αποτέλεσμα εκτέλεσης προγράμματος 2.



5. Απαιτήσεις Συστήματος και Εκτέλεση

Η παραπάνω άσκηση υλοποιήθηκε, όπως προαναφέρθηκε, στην γλώσσα προγραμματισμού Python, version 3.8.

Για να μπορεί να γίνει η εκτέλεση του προγράμματος είναι απαραίτητο να είναι εγκατεστημένη η βιβλιοθήκη ntlk.

Θέμα 5

Στη παρούσα ενότητα θα γίνει κατά βάθος ανάλυση του 5^{ου} ερωτήματος της άσκησης.

1. Εκφώνηση

Θέμα 5ο (20 μονάδες):

Ανάπτυξη Συντακτικού Αναλυτή σε άλλη γλώσσα Προγραμματισμού. Αναζητείστε στο διαδίκτυο ή αναπτύξτε εσείς Συντακτικό Αναλυτή που με βάση τους κανόνες συντακτικής ανάλυσης της πρότυπης λύσης σε Prolog που σας δόθηκε παράγει το συντακτικό δένδρο της πρότασης. Τεκμηριώστε πειστικά τον κώδικά σας.

2. Εισαγωγικό Πλαίσιο

Η παρούσα άσκηση αποσκοπεί στην ανάπτυξη ενός Συντακτικού Αναλυτή σε διαφορετική γλώσσα προγραμματισμού.

Αυτός με βάση τους συντακτικούς κανόνες της πρότυπης λύσης θα εμφανίζει το συντακτικό δέντρο μιας πρότασης. Η άσκηση αυτή πραγματοποιήθηκε με τη γλώσσα προγραμματισμού Python version 3.8.

Ακόμα χρησιμοποιήθηκε η βιβλιοθήκη nltk, η οποία είναι η πλέον κατάλληλη βιβλιοθήκη της Επεξεργασία Φυσικής Γλώσσας στη Python. Έτσι στη συγκεκριμένη εφαρμογή ανάλογα την πρόταση που έχουμε στον κώδικά μας θα εμφανίζεται το αντίστοιχο συντακτικό δέντρο, το οποίο θα είναι σε μορφή functor.

3. Επεξήγηση Κώδικα

Σε αυτήν την ενότητα θα γίνει επεξήγηση του κώδικα της υλοποιημένης εφαρμογής.

Το προαναφερθέν αρχείο βρίσκεται στο φάκελο Subject 5 με όνομα τίτλου syntantic_analyzer.py .

3.1. Συντακτικοί κανόνες (Grammar Rules)

Το πρόγραμμα έχει υλοποιηθεί σε Python 3.8.

Πρωταρχικό στάδιο για την υλοποίηση ενός συντακτικού δέντρου είναι ο καθορισμός των συντακτικών κανόνων (γραμματική – grammar rules).

Αρχικά, θα πρέπει η κάθε πρόταση να αναλύεται σε επιμέρους στοιχεία. Στη συγκεκριμένη γραμματική η πρόταση αναλύεται σε noun_phrase και verb_phrase. Οι κανόνες είναι ίδιοι με αυτούς της πρότυπης λύσης.

Παρακάτω απεικονίζονται οι απαραίτητες πληροφορίες της γραμματικής:

```
syntantic_analyzer.py - C:\Users\Dimitris\Desktop\syntantic_analyzer.py (3.8.3)
File Edit Format Run Options Window Help
import nltk
grammar = nltk.CFG.fromstring("""
S -> NP VP
NP -> PN | Det N | N
VP -> IV | IV Adv | AV Adj | TV PN NP | V NP
IV -> 'runs' | 'run' | 'running' | 'hurts' | 'hurt' | 'hurting' | 'walks' | 'walk' | 'walking' | 'jumps' | 'jump' | 'jumping' | 'shoots' | 'shoot' | 'shooting'
AV -> 'is' | 'are' | 'does' | 'do'
TV -> 'gives' | 'give' | 'gave' | 'giving'
PN -> 'mary' | 'john' | 'tomy'
Adv -> 'quickly' | 'slowly' | 'independently'
Det -> 'the' | 'a' | 'an'
N -> 'food' | 'cat' | 'dog' | 'dogs' | 'cat' | 'cats' | 'book' | 'books' | 'feather' | 'feathers' | 'baby' | 'babies' | 'boy' | 'boys' | 'girls' | 'girl' | 'icecream' | 'icecreams'
Adj -> 'scary' | 'tall' | 'short' | 'blonde' | 'slim' | 'fat'
V -> 'chased' | 'chase' | 'needs' | 'hates' | 'hate' | 'has' | 'have' | 'loves' | 'love' | 'kicks' | 'kick' | 'jumps' | 'jump'
""")
sentence = "mary gave john a book"
sent=sentence.split()
rd_parser = nltk.RecursiveDescentParser(grammar)
print("Parsing the sentence:"+ " " + sentence)
for tree in rd_parser.parse(sent):
    print("The tree for the above sentence is:")
    print(tree)
    break
```

Εικόνα 16 : Εικόνα Κώδικα.

3.2. Καθορισμός πρότασης

Το πρόγραμμα για να μπορέσει να δημιουργήσει το συντακτικό δέντρο θα πρέπει να γνωρίζει για ποια πρόταση θα εργαστεί.

Η συγκεκριμένη πρόταση θα χωριστεί (split) σε μία λίστα. Όπως θα δούμε και παρακάτω αυτή η λίστα θα μας χρειαστεί στην παραγωγή του συντακτικού δέντρου.

3.3. Ορισμός του Parser

Προκειμένου να ολοκληρωθεί η διαδικασία επιτυχώς θα πρέπει να ορίσουμε τον parser, δηλαδή να του «εισάγουμε» την γραμματική, σύμφωνα με την οποία θα λειτουργεί.

Η Python και η βιβλιοθήκη nltk, μας δίνουν τη δυνατότητα να φτιάξουμε έναν απλό simple down CFG parser.

Ειδικότερα, η κατάλληλη συνάρτηση για αυτό είναι η RecursiveDescentParser(), η οποία δέχεται σαν είσοδο τη γραμματική που ορίσαμε παραπάνω.

3.4. Δημιουργία Συντακτικού Δέντρου

Με βάση τη παραπάνω διαδικασία θα δημιουργηθεί το τελικό αποτέλεσμα μετά την εκτέλεση ενός for-loop.

Αυτό που εκτελεί το συγκεκριμένο είναι για κάθε λέξη που υπάρχει στην λίστα και μετέπειτα αντιστοιχίζεται με κάποιο κανόνα της γραμματικής. Η διαδικασία επαναλαμβάνεται αναδρομικά.

4. Αποτελέσματα Εφαρμογής

Σε αυτή την ενότητα θα αναφερθούμε στα αποτελέσματα της εφαρμογής μας. Στην παρακάτω εικόνα βλέπουμε το αποτέλεσμα της εκτέλεσης του κώδικα για το εξής κείμενο:

mary gave john a book

```
===== RESTART: C:\Users\Dimitris\Desktop\syntactic_analyzer.py ===  
Parsing the sentence: mary gave john a book  
The tree for the above sentence is:  
(S (NP (PN mary)) (VP (TV gave) (PN john) (NP (Det a) (N book)))))
```

Εικόνα 17 : Εικόνα αποτελεσμάτων

5. Απαιτήσεις Συστήματος και Εκτέλεση

Η παραπάνω άσκηση υλοποιήθηκε, όπως προαναφέρθηκε, στην γλώσσα προγραμματισμού Python, version 3.8.

Για να μπορεί να γίνει η εκτέλεση του προγράμματος είναι απαραίτητο να είναι εγκατεστημένη η βιβλιοθήκη nltk.

Η επίλυση της άσκησης βρίσκεται στον φάκελο Subject 5 και συγκεκριμένα στα αρχείο syntactic_analyzer.py.

Θέμα 6

Στη παρούσα ενότητα θα γίνει αναλυτική επεξήγηση του 6^{ου} ερωτήματος της άσκησης.

1. Εκφώνηση

Θέμα 6ο (20 μονάδες):

Ανάπτυξη Σημασιολογικού Αναλυτή σε άλλη γλώσσα Προγραμματισμού. Αναζητείστε στο διαδίκτυο ή αναπτύξτε εσείς Σημασιολογικό Αναλυτή που με βάση τους κανόνες σημασιολογικής ανάλυσης της πρότυπης λύσης σε Prolog που σας δόθηκε παράγει τα σημαινόμενα της πρότασης (σχέσεις μεταξύ ρημάτων, ουσιαστικών, επιθέτων, κ.λπ). Τεκμηριώστε πειστικά τον κώδικά σας.

2. Εισαγωγικό Πλαίσιο

Αν και η φυσική γλώσσα μας – ανεξαρτήτως χώρας - είναι πεπερασμένη, οι σημασιολογικοί συνδυασμοί είναι πρακτικά άπειροι.

Είναι πρακτικά αδύνατον να αποθηκεύονται όλες τις προτάσεις ως έχουν, και έτσι χρησιμοποιούνται δυναμικά σημασιολογικά δίκτυα (semantic networks).

Οι σχέσεις αυτές χρησιμοποιούνται για την εξαγωγή γνώσεις σε επόμενο επίπεδο.

3. Επεξήγηση Κώδικα – Αποτελέσματα

Σε αυτήν την ενότητα θα γίνει επεξήγηση του κώδικα της υλοποιημένης εφαρμογής.

Το προαναφερθέν αρχείο βρίσκεται στο φάκελο Subject 5 με όνομα τίτλου semantic_analyzer.py .

```
semantic_analyzer.py - C:\Users\Dimitris\Desktop\semantic_analyzer.py (3.8.3)
File Edit Format Run Options Window Help

#import modules
import os.path
from gensim import corpora
from gensim.models import LsiModel
from nltk.tokenize import RegexpTokenizer
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from gensim.models.coherencemodel import CoherenceModel
import matplotlib.pyplot as plt
def load_data(path,file_name):
    """
    Input : path and file_name
    Purpose: loading text file
    Output : list of paragraphs/documents and
            title(initial 100 words considred as title of document)
    """
    documents_list = []
    titles=[]
    with open( os.path.join(path, file_name) ,"r") as fin:
        for line in fin.readlines():
            text = line.strip()
            documents_list.append(text)
    print("Total Number of Documents:",len(documents_list))
    titles.append( text[0:min(len(text),100)] )
    return documents_list,titles
def preprocess_data(doc_set):
    """
    Input : docuemnt list
    Purpose: preprocess text (tokenize, removing stopwords, and stemming)
    Output : preprocessed text
    """
    # initialize regex tokenizer
    tokenizer = RegexpTokenizer(r'\w+')
    # create English stop words list
    en_stop = set(stopwords.words('english'))
    # Create p_stemmer of class PorterStemmer
    p_stemmer = PorterStemmer()
    # list for tokenized documents in loop
    texts = []
    # loop through document list
```

Εικόνα 18 : Εικόνα Κώδικα semantic_analyzer.py 1.

```
semantic_analyzer.py - C:\Users\Dimitris\Desktop\semantic_analyzer.py (3.8.3)
File Edit Format Run Options Window Help
# Loop through documents
for i in doc_set:
    # clean and tokenize document string
    raw = i.lower()
    tokens = tokenizer.tokenize(raw)
    # remove stop words from tokens
    stopped_tokens = [i for i in tokens if not i in en_stop]
    # stem tokens
    stemmed_tokens = [p_stemmer.stem(i) for i in stopped_tokens]
    # add tokens to list
    texts.append(stemmed_tokens)
return texts
def prepare_corpus(doc_clean):
    """
    Input : clean document
    Purpose: create term dictionary of our corpus and Converting list of documents
    Output : term dictionary and Document Term Matrix
    """
    # Creating the term dictionary of our corpus, where every unique term is as
    dictionary = corpora.Dictionary(doc_clean)
    # Converting list of documents (corpus) into Document Term Matrix using dict
    doc_term_matrix = [dictionary.doc2bow(doc) for doc in doc_clean]
    # generate LDA model
    return dictionary, doc_term_matrix
def create_gensim_lsa_model(doc_clean, number_of_topics, words):
    """
    Input : clean document, number of topics and number of words associated with
    Purpose: create LSA model using gensim
    Output : return LSA model
    """
    dictionary, doc_term_matrix = prepare_corpus(doc_clean)
    # generate LSA model
    lsamodel = LsiModel(doc_term_matrix, num_topics=number_of_topics, id2word =
    print(lsamodel.print_topics(num_topics=number_of_topics, num_words=words))
    return lsamodel
def compute_coherence_values(dictionary, doc_term_matrix, doc_clean, stop, start
    """
    Input : dictionary : Gensim dictionary
            corpus : Gensim corpus
            texts : List of input texts
            stop : Max num of topics
    purpose : Compute c_v coherence for various number of topics
    Output : model_list : List of LSA topic models
            coherence_values : Coherence values corresponding to the LDA model
    """
    coherence_values = []
    model_list = []
    for num_topics in range(start, stop, step):
        # generate LSA model
        model = LsiModel(doc_term_matrix, num_topics=number_of_topics, id2word =
        model_list.append(model)
        coherencemodel = CoherenceModel(model=model, texts=doc_clean, dictionary
        coherence_values.append(coherencemodel.get_coherence())
    return model_list, coherence_values
def plot_graph(doc_clean, start, stop, step):
    dictionary, doc_term_matrix = prepare_corpus(doc_clean)
    model_list, coherence_values = compute_coherence_values(dictionary, doc_term
        stop, start, step)
    # Show graph
    x = range(start, stop, step)
    plt.plot(x, coherence_values)
Ln: 35 Col: 0
```

Εικόνα 19 : Εικόνα Κώδικα semantic_analyzer.py 2.

```
# Show graph
x = range(start, stop, step)
plt.plot(x, coherence_values)
plt.xlabel("Number of Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.show()
start, stop, step=2, 12, 1
plot_graph(clean_text, start, stop, step)
# LSA Model
number_of_topics=7
words=10
document_list, titles=load_data("", "articles.txt")
clean_text=preprocess_data(document_list)
model=create_gensim_lsa_model(clean_text, number_of_topics, words)
```

Εικόνα 20 : Εικόνα Κώδικα semantic_analyzer.py 3.

Θέμα 7

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

1. Εκφώνηση

Θέμα 7ο (40 μονάδες):

Χρησιμοποιείτε 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 7^ο 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 κάνει ερωτήσεις.

Τα παραπάνω στην ολότητά τους θα επεξηγηθούν στις παρακάτω ενότητες.

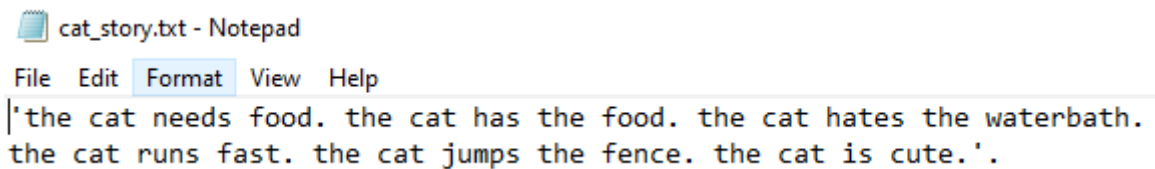
2.2. Δημιουργία Ιστοριών

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.

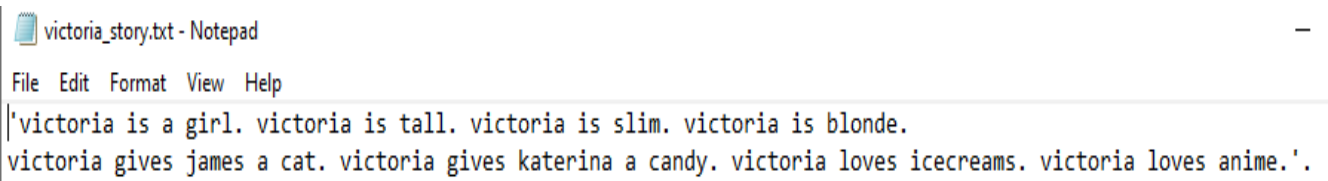
Τα προαναφερθέντα αρχεία ιστοριών βρίσκονται στον φάκελο Subject 7 με ονόματα cat_story.txt και victoria_story.txt και απεικονίζονται παρακάτω:

- cat_story.txt



Εικόνα 21 : 1^η Ιστορία cat_story.txt

- victoria_story.txt



Εικόνα 22 : 2^η 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 χρησιμοποιείται 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(likes,s)-->[likes]. v(likes,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 τρεις 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, χρησιμοποιείται μετά από αυτό και αποτελείται μόνο από μια λέξη. Παράδειγμα:

adv(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:

$$det(a) \rightarrow [a].$$
$$det(the) \rightarrow [the].$$

- 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:

$$n(book) \rightarrow [book].$$
$$n(cat) \rightarrow [cat].$$

- 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):- rh(N)) δεν γίνεται για πρακτικούς λόγους και όχι επιστημονικής ακρίβειας. Ένας άλλος τύπος φράσης είναι ο εξής και ο μόνος που χρησιμοποιείτε στο συγκεκριμένο πρόγραμμα, πέρα από το proper noun:

$$np(N) \rightarrow det(_), n(N).$$

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

$$vp(1, V, N) \rightarrow v(V, s), n(N).$$

$$vp(2, is, A) \rightarrow an(is), adj(A).$$
$$vp(3, V, N) \rightarrow v(V, s), pn(N).$$

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_nr) σε ένα αντικείμενο (sem_vp). Η ακολουθία φράσεων είναι η εξής:

$$sem(1, Sem) \rightarrow sem_nr(N), sem_vp(1, V, N1)$$

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 γίνεται 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:

$$sem(2, Sem) \rightarrow sem_nr(N), sem_vp(2, _, A),$$

Ο τύπος γνώσης που παράγεται είναι:

A(N) ή αλλιώς επίθετο(πρόσωπο)

Η μετατροπή γίνεται με τον παρακάτω τρόπο:

$$\text{Sem} = \dots[A, N]$$

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:

$$\text{sem}(3, \text{Sem}) \rightarrow \text{sem_np}(N), \text{sem_iv}(V, s),$$

Ο τύπος γνώσης που παράγεται είναι:

V(N) ή αλλιώς ρήμα(υποκείμενο)

Η μετατροπή γίνεται με τον παρακάτω τρόπο:

$$\text{Sem} = \dots[V, N]$$

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:

$$\text{sem}(4, \text{Sem}) \rightarrow \text{sem_np}(N), \text{sem_iv}(V, s), \text{sem_adv}(A),$$

Ο τύπος γνώσης που παράγεται είναι:

$V(N,A)$ ή αλλιώς ρήμα(υποκείμενο, επίρρημα)

Η μετατροπή γίνεται με τον παρακάτω τρόπο:

$$\text{Sem} = \dots[V,N,A]$$

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.

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

$$\text{sem}(5,\text{Sem}) \rightarrow \text{sem_np}(N), \text{sem_tv}(V,s), \text{sem_np}(N1), \text{sem_np}(N2),$$

Ο τύπος γνώσης που παράγεται είναι:

$V(N,N1,N2)$ ή αλλιώς ρήμα(υποκείμενο, άμεσο_αντικείμενο, αντικείμενο)

Η μετατροπή γίνεται με τον παρακάτω τρόπο:

$$\text{Sem} = \dots[V,N,N1,N2]$$

Παράδειγμα:

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

A = gives(victoria, katerina, candy),

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

2.5. Εισαγωγή Δεδομένων στη Βάση Γνώσης

Με 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, !.
```

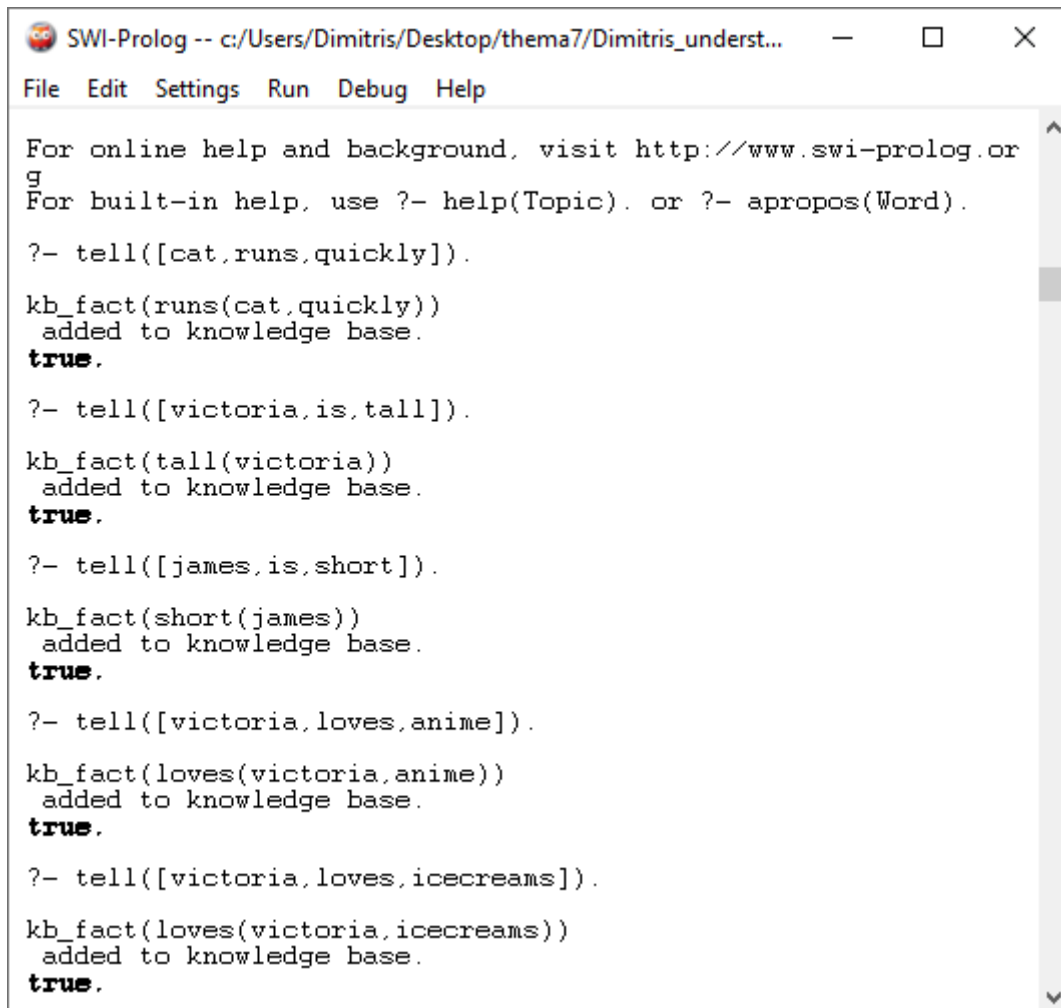
Ο χρήστης εισάγει μόνο το κείμενο (Sentence) και αυτό δοκιμάζεται από την Prolog σε ένα ερώτημα της μορφής:

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

Με αυτό τον τρόπο η Prolog εκτελεί το `assert(kb_fact(Sem))` και περνάει την γνώση στο πρόγραμμα.

Μετά την εισαγωγή των δεδομένων στη Βάση Γνώσης εμφανίζεται σχετικό μήνυμα επιτυχίας.

Ακολουθεί σχετική εικόνα από τη στιγμή που εισάγονται δεδομένα από το χρήστη στη Βάση Γνώσης (Knowledge Base).



Εικόνα 23 : Εισαγωγή Δεδομένων στη Βάση Γνώσης με την εντολή Tell.

2.6. 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.

2.6.1. 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:

```

688  /*-----*/
689  /*          yes/no queries          */
690  /*-----*/
691  q(1,tf,Sem) --> sem_av(does), sem_pn(N), sem_v(V,q), sem_n(N1),      {Sem=..[V,N,N1]}.
692  q(1,tf,Sem) --> sem_av(did), sem_pn(N), sem_v(V,q), sem_n(N1),      {Sem=..[V,N,N1]}.
693  q(2,tf,Sem) --> sem_av(is), sem_pn(N), sem_adj(A),                  {Sem=..[A,N]}.
694  q(3,tf,Sem) --> sem_av(does), sem_pn(N), sem_v(V,q), sem_n(N1),      {Sem=..[V,N,N1]}.
695  q(4,tf,Sem) --> sem_av(is), sem_pn(N), sem_iv(V,q),                  {Sem=..[V,N]}.
696  q(5,tf,Sem) --> sem_av(is), sem_pn(N), sem_iv(V,q), sem_adv(A),      {Sem=..[V,N,A]}.
697  q(6,tf,Sem) --> sem_av(does), sem_pn(N), sem_tv(V,q), sem_pn(N1), sem_np(N2), {Sem=..[V,N,N1,N2]}.
698  q(6,tf,Sem) --> sem_av(does), sem_pn(N), sem_tv(V,q), sem_pn(N1), sem_np(N2), {Sem=..[V,N,N1,N2]}.
```

Εικόνα 24 : 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 συνέχεια εξετάζεται αυτό και αν ισχύει, η Prolog τυπώνει Yes. Ειδάλλως τυπώνει No.

2.6.2. Ερωτήσεις Γεγονότων

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

Ένα γεγονός, μια καταχώρηση στην βάση γνώσης αποτελείται 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 :

```

714  /*-----*/
715  /*          fact queries          */
716  /*-----*/
717  q(1,fact,F) --> [who], sem_v(V,s),sem_n(N1), {Sem=..[V,F,N1], kb_fact(Sem)}.
718  q(1,fact,F) --> [what],sem_av(does),sem_pn(N),sem_v(V,q), {Sem=..[V,N,F], kb_fact(Sem)}.
719  q(2,fact,F) --> [who], sem_av(is), sem_adj(A), {Sem=..[A,F],kb_fact(Sem)}.
720  q(3,fact,F) --> [who], sem_vp(1,V,N1), {Sem=..[V,F,N1],kb_fact(Sem)}.
721  q(3,fact,F) --> [who], sem_av(does),sem_pn(N),sem_v(V,q), {Sem=..[V,N,F],kb_fact(Sem)}.
722  q(4,fact,F) --> [who], sem_av(is),sem_iv(V,q), {Sem=..[V,F],kb_fact(Sem)}.
723  q(5,fact,F) --> [how], sem_av(does),sem_pn(N),sem_iv(V,s), {Sem=..[V,N,F],kb_fact(Sem)}.
724  q(5,fact,F) --> [how], sem_av(is),sem_pn(N),sem_iv(V,q), {Sem=..[V,N,F],kb_fact(Sem)}.
725  q(5,fact,F) --> [who], sem_iv(V,s),sem_adv(A), {Sem=..[V,F,A],kb_fact(Sem)}.
726  q(6,fact,F) --> [who], sem_tv(V,s),sem_pn(N1),sem_np(N2), {Sem=..[V,F,N1,N2],kb_fact(Sem)}.
727  q(6,fact,F) --> [who], sem_av(is),sem_pn(N),sem_tv(V,q2),sem_np(N2),[to], {Sem=..[V,N,F,N2],Sem}.
728  q(6,fact,F) --> [what],[is],sem_pn(N),sem_tv(V,q2),[to],sem_pn(N1), {Sem=..[V,N,N1,F],Sem}.

```

Εικόνα 25 : 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:

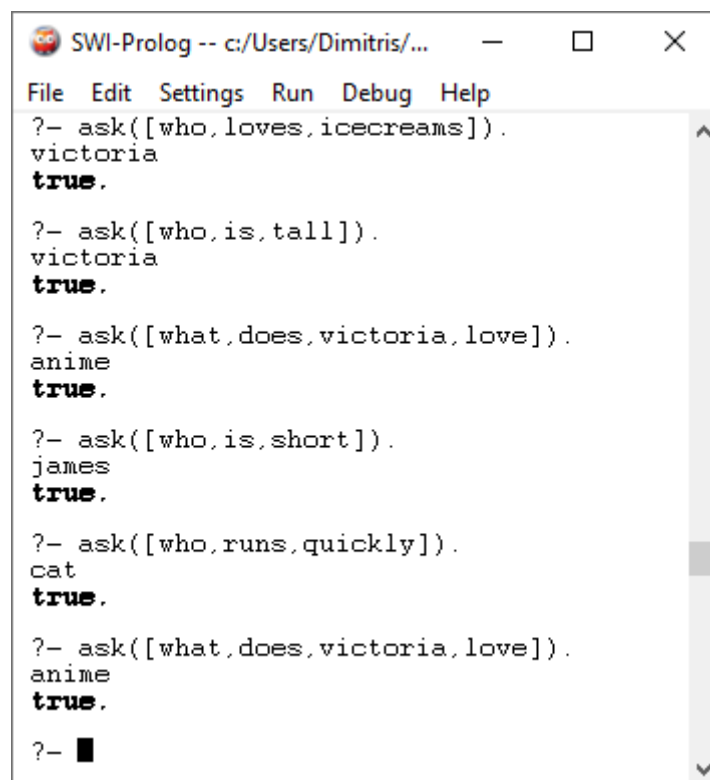
- Το q() τροφοδοτείτε με το X, την φυσική γλώσσα.
- Αυτή παρέχει όλους τους όρους που χρειάζονται, και παράγεται η μορφή γνώσης με έναν άγνωστο, το Fact.
- Όταν δημιουργηθεί το Sem, θέτετε ως στόχος γεμίζοντας το fact με πιθανή απάντηση, αν αυτή υπάρχει.

2.7. Αποτελέσματα Υποβολής Ερωτήσεων στη Βάση Γνώσης

Σε αυτή την ενότητα θα σας παραθέσουμε τα αποτελέσματα των ερωτήσεων μας στη Βάση Γνώσης.

Για να γίνει σωστά η διαδικασία, ο χρήστης πρέπει πρώτα να έχει εισάγει δεδομένα στη Βάση με την εντολή `tell` και έπειτα να τη ρωτήσει για αυτά με την εντολή `ask`.

Ακολουθούν τα σχετικά αποτελέσματα από την παραπάνω διαδικασία:



```
SWI-Prolog -- c:/Users/Dimitris/...
File Edit Settings Run Debug Help
?- ask([who, loves, icecreams]).
victoria
true.

?- ask([who, is, tall]).
victoria
true.

?- ask([what, does, victoria, love]).
anime
true.

?- ask([who, is, short]).
james
true.

?- ask([who, runs, quickly]).
cat
true.

?- ask([what, does, victoria, love]).
anime
true.

?- 
```

Εικόνα 26 : Αποτελέσματα ερωτήσεων στη Βάση Γνώσης με την εντολή `ask`.

Περιεχόμενα Απεσταλμένου Αρχείου

Το τελικό αρχείο της εργασίας (Εργασία_Επεξεργασίας_Φυσικής_Γλώσσας.zip) θα περιέχει τα παρακάτω:

1. Ένα φάκελο για κάθε θέμα της εργασίας που θα εμπεριέχει σε υποφακέλους τις υλοποιήσεις και όλα τα απαραίτητα αρχεία για το κάθε ζητούμενο ερώτημα.
2. Ένα αρχείο κειμένου με το όνομα και τον Αριθμό Μητρώου (Αριθμός_Μητρώου.txt).
3. Ένα αρχείο κειμένου με τη Ρήτρα που θα ήθελα να θέσω για το μάθημα σας (Ρήτρα.txt).
4. Το παρόν έγγραφο Εργασία Επεξεργασίας Φυσικής Γλώσσας.pdf, το οποίο περιλαμβάνει όλα όσα ζητήθηκαν από την εκφώνηση της εργασίας.