ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΑΠΑΡΑΣΤΑΣΗ ΓΝΩΣΗΣ ΚΑΙ ΕΠΕΞΕΡΓΑΣΙΑ ΦΥΣΙΚΗΣ ΓΛΩΣΣΑΣ

Διπλωματική Εργασία

της

Σαμαρά Ευγενίας
(mai21046)

Επιβλέπων Καθηγητής
Ρεφανίδης Ιωάννης

Θεσσαλονίκη, Οκτώβριος 2023

ΑΝΑΠΑΡΑΣΤΑΣΗ ΓΝΩΣΗΣ ΚΑΙ ΕΠΕΞΕΡΓΑΣΙΑ ΦΥΣΙΚΗΣ ΓΛΩΣΣΑΣ

Σαμαρά Ευγενία

Πτυχίο Εφαρμοσμένης Πληροφορικής, ΠΑΜΑΚ, 2020

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Ρεφανίδης Ιωάννης

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 31/10/2023

Ρεφανίδης Ιωάννης	Σακελλαρίου Ηλίας	Κολωνιάρη Γεωργία
.....	.....	.....

Σαμαρά Ευγενία

.....

Περίληψη

Υπάρχουν πολυπληθείς τρόποι αναπαράστασης γνώσης όσον αφορά το πεδίο της τεχνητής νοημοσύνης, και πιο συγκεκριμένα το πεδίο της επεξεργασίας φυσικής γλώσσας. Μετά το 2012, με την ανακοίνωση του Γράφου Γνώσης της Google, η αναπαράσταση αυτή επανήλθε στο προσκήνιο και πραγματοποιήθηκε εκτεταμένη έρευνα για την χρήση της σε εργασίες επεξεργασίας φυσικής γλώσσας. Αποτελεί έναν εύκολα κατανοητό και διαισθητικό τρόπο αναπαράστασης ο οποίος όμως βασίζεται παράλληλα και σε επίσημα πρότυπα. Στόχος της παρούσας εργασίας είναι η αναπαραγωγή φυσικού κειμένου από δομημένη μορφή γνώσης, και πιο συγκεκριμένα κειμενικών δεδομένων απλών προτάσεων δομημένα σε γράφο γνώσης. Πρώτα γίνεται βιβλιογραφική ανασκόπηση στις έννοιες της τεχνητής νοημοσύνης και των νευρωνικών δικτύων, τα οποία χρησιμοποιούνται σε διεργασίες επεξεργασίας φυσικής γλώσσας όπως η συντακτική και η σημασιολογική ανάλυση. Έπειτα γίνεται αναφορά στο θεωρητικό υπόβαθρο που είναι απαραίτητο για την κατανόηση της διαδικασίας επεξεργασίας φυσικής γλώσσας, της αναπαράστασης γνώσης και φυσικά των γράφων γνώσης. Τέλος, παρουσιάζεται αναλυτικά η μεθοδολογία ανάπτυξης του προγράμματος. Για την δημιουργία του γράφου χρησιμοποιήθηκε το περιβάλλον Neo4j και η γλώσσα Cypher, ενώ το πρόγραμμα αναπαραγωγής φυσικού κειμένου αναπτύχθηκε με την γλώσσα Python στο περιβάλλον PyCharm.

Λέξεις Κλειδιά: Αναπαράσταση γνώσης, επεξεργασία φυσικής γλώσσας, γράφοι γνώσης, Neo4j, αναπαραγωγή φυσικού κειμένου

Abstract

There are numerous ways of representing knowledge in the field of artificial intelligence, and more specifically the field of natural language processing. After 2012, with the announcement of Google's Knowledge Graph, this representation came back to the foreground and extensive research was conducted into its use in natural language processing tasks. It is an easy-to-understand and intuitive way of representation, but it is also based on formal standards. The aim of this work is the reproduction of natural text from a structured form , and more specifically textual data of simple sentences structured in a knowledge graph. First a literature review is done on the concepts of artificial intelligence and neural networks, which are used in natural language processes such as syntactic and semantic analysis. Then reference is made to the theoretical background necessary for understanding natural language processing, knowledge representation and of course knowledge graphs. Finally, the program development methodology is presented in detail. The Neo4j environment and the Cypher language were used to create the graph, while the natural text reproduction was developed using Python in the PyCharm environment.

Keywords: Knowledge representation, natural language processing, knowledge graphs, Neo4j, natural text reproduction

Πρόλογος – Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες πρώτα από όλα στον επιβλέποντα καθηγητή μου κ. Ρεφανίδη Ιωάννη, χωρίς την πολύτιμη βοήθεια και καθοδήγησή του οποίου δεν θα ήταν δυνατή η εκπόνηση της παρούσας εργασίας. Επιπλέον,θα ήθελα να ευχαριστήσω τους καθηγητές, κ. Σακελλαρίου Ηλία και κ. Κολωνιάρη Γεωργία για την συμμετοχή τους στην εξεταστική επιτροπή. Τέλος θα ήθελα να εκφράσω την ευγνωμοσύνη μου στους γονείς μου Μίλτο και Τασούλα που πάντα με υποστηρίζουν, όπως και στην αδερφή μου Γεωργία και σε όλους τους φίλους μου που ήταν δίπλα μου καθ' όλη τη διάρκεια των μεταπτυχιακών σπουδών μου.

Αφιερώνω την παρούσα διπλωματική εργασία στην ανιψιά μου Νεφέλη, η οποία πάντα πιστεύει σε μένα παρά τις όποιες δυσκολίες.

Περιεχόμενα

Κατάλογος Εικόνων

Πίνακας συντομεύσεων – ακρωνυμίων

ANN	Artificial eural Networks
BPTT	Back Propagation Through Time
DL	Deep Learning
FFNN	Feed Forward Neural Networks
LSTM	Long Short Term Memory

1. Εισαγωγή

1.1. Πρόβλημα – Σημαντικότητα του θέματος

Τα τελευταία χρόνια, με την εξέλιξη της τεχνολογίας τόσο στο πεδίο της Τεχνητής Νοημοσύνης και συγκεκριμένα της μηχανικής μάθησης, όσο και στο πεδίο της Επεξεργασίας Φυσικής Γλώσσας, η ανάγκη για τη βελτίωση του τρόπου αναπαράστασης των δεδομένων είναι επιτακτική. Το ίδιο ισχύει και για τις μεθόδους επεξεργασίας κειμενικών δεδομένων, τόσο από πηγές φυσικής γλώσσας όσο και από πηγές δομημένης γνώσης. Με μεθόδους μηχανικής μάθησης αντιμετωπίζονται προβλήματα όπως η αυτόματη περίληψη, η αυτόματη μετάφραση ή η παραγωγή φυσικής γλώσσας. Ωστόσο, οι μέθοδοι μηχανικής μάθησης δεν βασίζονται στην κατανόηση του κειμένου αλλά στην στατιστική, με συνέπεια η αποτελεσματικότητά τους να μην μπορεί να ξεπεράσει ένα όριο. Έτσι επανέρχεται πάλι στο προσκήνιο η χρήση παραδοσιακών τεχνικών επεξεργασίας φυσικής γλώσσας, βασισμένων σε γραμματικές, σε συνδυασμό με μεθόδους αναπαράστασης φυσικής γλώσσας, χρησιμοποιώντας τις μεθόδους μηχανικής μάθησης σε συμπληρωματικό και υποστηρικτικό ρόλο (π.χ., ως ευρετικές συναρτήσεις).

Ακόμα, από το 2012 και μετά, με την ανακοίνωση του Γράφου Γνώσης της Google (Google Knowledge Graph), η μορφή αυτή αναπαράστασης γνώσης επανήλθε στο επίκεντρο, αν και σαν δομή υπάρχει εδώ και δεκαετίες. Οι Γράφοι Γνώσης παρέχουν την ευκολία της αναπαράστασης των δεδομένων σε οντότητες και των σχέσεων μεταξύ των οντοτήτων αυτών, από οποιαδήποτε περιοχή (domain). Η αναπαράσταση γίνεται διαισθητικά, επομένως γίνεται εύκολα κατανοητή τόσο από τους ανθρώπους όσο και από τη μηχανή, με σωστή μοντελοποίηση ενός προβλήματος. Για αυτό το λόγο η χρήση των γράφων γνώσης σε εργασίες επεξεργασίας φυσικής γλώσσας είναι πλέον συνηθισμένη με όλο και περισσότερη επιστημονική βιβλιογραφία να δημοσιεύεται τα τελευταία χρόνια.

1.2. Σκοπός – Στόχοι

Στο πλαίσιο αυτής της εργασίας γίνεται προσπάθεια να απαντηθούν προβλήματα από το χώρο της επεξεργασίας φυσικής γλώσσας μέσω της αναπαράστασης γνώσης. Ο στόχος είναι να επιτευχθεί, σε περιορισμένο πλαίσιο, αντιστοίχιση μεταξύ δομημένης αναπαράστασης γνώσης και κειμένου σε φυσική γλώσσα. Από τη γνώση δηλαδή σε δομημένη μορφή να αναπαράγεται φυσικό κείμενο. Αυτό πραγματοποιήθηκε αρχικά με την χειροκίνητη δημιουργία ενός γράφου γνώσης στο περιβάλλον Neo4j από τα κειμενικά μας δεδομένα απλών προτάσεων και τη χρήση της γλώσσας Cypher. Η συντακτική και σημασιολογική ανάλυση του κειμένου προκειμένου να δημιουργηθεί σωστά ο γράφος και να οριστούν κατάλληλα οι οντότητες από τις προτάσεις και οι σχέσεις μεταξύ τους έγινε με τη χρήση του εργαλείου Stanza του Stanford NLP Group. Στη συνέχεια με την χρήση της γλώσσας Python και τη σύνδεση του Γράφου Γνώσης

στο προγραμματιστικό περιβάλλον PyCharm έγινε η αναπαραγωγή του φυσικού κειμένου.

1.3. Διάρθρωση της μελέτης

Η παρούσα διπλωματική εργασία αποτελείται από 6 κεφάλαια. Στα κεφάλαια 2, 3 και 4 γίνεται μια επισκόπηση του θεωρητικού υπόβαθρου πάνω στο οποίο βασίζεται η εργασία. Πιο συγκεκριμένα, στο κεφάλαιο 2 παρουσιάζονται οι έννοιες της μηχανικής μάθησης και των νευρωνικών δικτύων, μαζί με ορισμένες αρχιτεκτονικές νευρωνικών δικτύων που χρησιμοποιούνται για γλωσσολογική ανάλυση και επεξεργασία φυσικής γλώσσας (και χρησιμοποιούνται και από το εργαλείο Stanza). Στο κεφάλαιο 3 γίνεται αναφορά στην επεξεργασία της φυσικής γλώσσας, στα στάδια επεξεργασίας και στις μεθόδους που χρησιμοποιούνται για να επιτευχθεί, όπως η συντακτική και σημασιολογική ανάλυση. Στο κεφάλαιο 4 παρατίθεται το θεωρητικό υπόβαθρο που είναι απαραίτητο για την κατανόηση της αναπαράστασης γνώσης και και των γράφων γνώσης. Αναφέρονται οι διάφοροι μέθοδοι αναπαράστασης γνώσης, οι οντολογίες, η γλώσσα OWL, περιγράφεται η μέθοδος αναπαράστασης με γράφους γνώσης και η γλώσσα RDF. Στο κεφάλαιο 5 παρουσιάζεται αναλυτικά η μεθοδολογία που ακολουθήθηκε για την δημιουργία του γράφου γνώσης και του προγράμματος που αναπτύχθηκε στα πλαίσια της διπλωματικής. Αναφέρονται οι βιβλιοθήκες που χρησιμοποιήθηκαν, τα εργαλεία, τα κειμενικά δεδομένα και η επεξεργασία τους, το περιβάλλον Neo4j και η γλώσσα Cypher για την δημιουργία της δομημένης γνώσης με γράφο και τέλος, πως επιτεύχθηκε η αναπαραγωγή φυσικού κειμένου. Στο κεφάλαιο 6 παρουσιάζονται μια σύνοψη και τα συμπεράσματα της διπλωματικής στα οποία καταλήξαμε, μαζί με τις δυσκολίες που αντιμετωπίστηκαν και την περαιτέρω έρευνα που μπορεί να γίνει. Στο τέλος της διπλωματικής εργασίας παρατίθεται το σύνολο των βιβλιογραφικών πηγών που χρησιμοποιήθηκαν για τη συγγραφής της, για τις οποίες γίνονται αναφορές σε όλη την έκταση της εργασίας.

2. Τεχνητή Νοημοσύνη

Ο όρος "Τεχνητή Νοημοσύνη" προτάθηκε επίσημα σε ένα συνέδριο στο Πανεπιστήμιο Dartmouth,το 1959 (Zhang & Lu,2021). Αποτελεί κλάδο της Επιστήμης των Υπολογιστών και ασχολείται κυρίως με την αυτοματοποίηση της Ευφυούς συμπεριφοράς (Chowdhary,2020). Επιχειρείται δηλαδή η εκπαίδευση υπολογιστών ώστε να προσομοιώνουν ανθρώπινες ευφυής συμπεριφορές όπως η μάθηση, η λήψη αποφάσεων, η επεξεργασία φυσικής γλώσσας και πολλές άλλες. Η γνώση λαμβάνεται ως αντικείμενο (object) από την τεχνητή νοημοσύνη, μέσα από την οποία αποκτάται περισσότερη γνώση, παράγεται νέα γνώση, μελετάται και αναλύεται η ίδια η γνώση, αλλά και μέθοδοι επεξεργασίας της.

2.1. Μηχανική Μάθηση

Η Μηχανική μάθηση αποτελεί πεδίο της τεχνητής νοημοσύνης και διερευνά τη μελέτη και την κατασκευή αλγορίθμων που μπορούν να μαθαίνουν από τα δεδομένα, να κάνουν προβλέψεις σχετικά με αυτά και να βελτιώνουν την απόδοσή τους, χωρίς να χρειάζεται να προγραμματίζονται συνέχεια εκ νέου.

Οι αλγόριθμοι αυτοί χωρίζονται σε τέσσερις βασικές κατηγορίες, σύμφωνα με τον τρόπο εκμάθησης (Balakrishnama and Ganapathiraju, 1998):

1. Επιβλεπόμενη Μάθηση(Supervised Learning): Αλγόριθμοι που κατασκευάζουν ένα μαθηματικό μοντέλο πάνω σε ένα σύνολο δεδομένων τα οποία περιέχουν τις εισόδους και τα επιθυμητά αποτελέσματα. Το σύνολο δεδομένων εισόδου χωρίζεται σε σύνολο δεδομένων εκπαίδευσης και δοκιμαστικών δεδομένων. Χρησιμοποιούνται σε προβλήματα ταξινόμησης(classification), παλινδρόμησης (regression),διερμηνείας (interpretation), πρόγνωσης (prediction), και ενεργητικής μάθησης. Στα προβλήματα ταξινόμησης στόχος είναι η πρόβλεψη εξόδου διακριτών τιμών ενώ στης παλινδρόμησης (regression) στόχος είναι η πρόβλεψη εξόδου συνεχών τιμών,
2. Μη Επιβλεπόμενη Μάθηση(Unsupervised Learning): Αλγόριθμοι που δέχονται ως είσοδο ένα σύνολο δεδομένων το οποίο περιλαμβάνει μόνο τα δεδομένα εισόδου και εντοπίζει δομές σε αυτά. Η μάθηση γίνεται από δεδομένα που δεν έχουν προηγουμένως κατηγοριοποιηθεί, σε αντίθεση με την επιβλεπόμενη μάθηση. Χρησιμοποιούνται σε προβλήματα ομαδοποίησης (clustering) και ανάλυσης συσχετισμών (association analysis).

3. Ημι-επιβλεπόμενη μάθηση(Semi-supervised Learning): Αλγόριθμοι που βρίσκονται μεταξύ των δύο προηγούμενων μεθόδων. Γίνεται χρήση τόσο κατηγοριοποιημένων όσο και μη κατηγοριοποιημένων δεδομένων ταυτόχρονα.
4. Ενισχυτική Μάθηση (Reinforcement Learning): Σε αυτό το πεδίο της μηχανικής μάθησης πράκτορες λογισμικού καλούνται να λάβουν αποφάσεις σε ένα συγκεκριμένο πλαίσιο ώστε να μεγιστοποιήσουν το αποτέλεσμα.Προκειμένου δηλαδή να επιτευχθεί το καλύτερο αποτέλεσμα, επιλέγεται τη δράση με την υψηλότερη ανταμοιβή αλλά και εντοπίζονται άγνωστες ενέργειες. Σε αυτό ανήκουν και τομείς όπως η θεωρία παιγνίων , η θεωρία πληροφοριών και τα προβλήματα σχεδιασμού (Planning).

Τις τελευταίες δεκαετίες στο πεδίο της μηχανικής μάθησης οι εξελίξεις είναι ραγδαίες. Αυτές συμβαίνουν τόσο στους αλγόριθμους μάθησης, όσο και στις τεχνικές για πιο αποτελεσματική προεπεξεργασία των δεδομένων, όπως η η εξέλιξη των τεχνητών νευρωνικών δικτύων (artificial neural networks - ANN) προς τις ολοένα και πιο βαθιές αρχιτεκτονικές νευρωνικών δικτύων με βελτιωμένες δυνατότητες εκμάθησης που συνοψίζονται ως βαθιά μάθηση (deep learning - DL) (Janiesch et al, 2021).

2.2. Νευρωνικά Δίκτυα

Τα νευρωνικά δίκτυα μιμούνται τον τρόπο λειτουργίας του ανθρώπινου εγκεφάλου, ο οποίος αποτελείται από διακριτά στοιχεία, τους νευρώνες (neurons), που επικοινωνούν ο ένας με τον άλλο.Υπάρχουν περίπου 10 δισ. νευρώνες τοποθετημένοι σε ομάδες , οι οποίες αποτελούν καθεμία ένα φυσικό νευρωνικό δίκτυο. Κάθε νευρώνας αποτελείται από 3 κύρια τμήματα. Υπάρχουν οι δενδρίτες (dendrites),οι οποίοι λειτουργούν ως κανάλια εισόδου για το νευρώνα, το κυρίως κυτταρικό σώμα (cell body) και ο άξονας του κυττάρου-νευροάξονα (axon), που συνδέει ένα νευρώνα με άλλους νευρώνες. Ένα Τεχνητό Νευρωνικό Δίκτυο λειτουργεί κατ αντιστοιχία σε τρία επίπεδα (layers). Το επίπεδο εισόδου που δέχεται την είσοδο, το κρυφό επίπεδο που επεξεργάζεται την είσοδο και τέλος, το επίπεδο εξόδου στέλνει την υπολογισμένη έξοδο. Προσαρμόζονται στις μεταβαλλόμενες εισόδους, έτσι ώστε το δίκτυο να παράγει το καλύτερο δυνατό αποτέλεσμα χωρίς να χρειάζεται να επανασχεδιαστούν τα κριτήρια εξόδου (Mahesh,2020).

Ένας τεχνητός νευρώνας αποτελείται αρχικά από ένα σήμα, ή αλλιως μια τάση πόλωσης (bias) β, η οποία έχει σταθερή τιμή και είναι συνήθως -1 ή 1. Αποτελεί είσοδο για όλους τους νευρώνες. Μετά, υπάρχουν οι έξοδοι ι x1,…,xn κτλ διάφορων νευρώνων που γίνονται είσοδοι σε άλλους νευρώνες. Οι είσοδοι x1,…,xn ενός νευρώνα πολλαπλασιάζονται με συντελεστές βαρύτητας w1,…,wn και η συνολική είσοδος στον

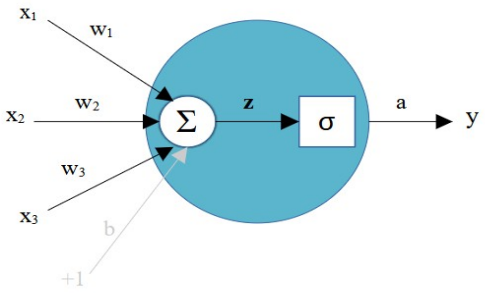
νευρώνα i είναι το άθροισμα όλων των επιμέρους εισόδων της , μετά τον πολλαπλασιασμό τους με τους συντελεστές βαρύτητας. Αυτό συμβαίνει στη συσκευή του αθροιστή (Σ).Μια επιπλέον είσοδος του αθροιστή είναι σταθερή, δηλ. xn+1= 1, και πολλαπλασιάζεται με τη σταθερά πόλωσης (bias) β. Η έξοδος (σ) του αθροιστή υπολογίζεται ως εξής:

$$\sum_{i=1}^{n} wixi + \beta = \sum_{i=1}^{n+1} wixi = w^T \bullet x \quad ,$$

όπου το $w^T \bullet x$ παριστάνει το εσωτερικό γινόμενο του διανύσματος $x = (x_1,\ldots, x_n, 1)^T$ εισόδου του τεχνητού νευρώνα επί το διάνυσμα $w = (w_1,\ldots, w_n, w_{n+1})^T$ των βαρών. (Καμπουρλάζος και Παπακώστας ,2015,σελ 13). Το σταθμισμένο (γραμμικό) άθροισμα σ των εισόδων του τεχνητού νευρώνα οδηγείται στη συνέχεια σε ένα (μη-γραμμικό) στοιχείο παραμόρφωσης f(σ), που ονομάζεται συνάρτηση ενεργοποίησης. Αυτό συμβαίνει καθώς σε διαφορετική περίπτωση η έξοδος του νευρώνα θα ήταν μια απλή γραμμική συνάρτηση.

Στην Εικόνα 2.1. (Jurafsky & Martin, 2023, σελ. 136) βλέπουμε σχηματικά ένα νευρωνικό δίκτυο, το οποίο έχει τρεις εισόδους x1, x2 και x3 , τα αντίστοιχα βάρη τους w1, w2 και w3 και τη σταθερά πόλωσης b με τιμή +1. Η έξοδος του δικτύου είναι η y. Ακόμα υπάρχει η έξοδος του αθροιστή , δηλαδή το z, και η έξοδος της σιγμοειδής συνάρτησης ενεργοποίησης, δηλαδή η a. Στην συγκεκριμένη περίπτωση y=a, κάτι που δεν ισχύει πάντα, ιδίως όταν πρόκειται για βαθύτερα νευρωνικά δίκτυα, στα οποία η y συμβολίζει την έξοδο ολόκληρου του δικτύου και η a ενός μεμονωμένου κόμβου.

Αυτή η περίπτωση η έξοδος της μονάδας y είναι η ίδια με το a, αλλά σε βαθύτερα δίκτυα θα διατηρήσουμε το y για να σημαίνει την τελική παραγωγή ολόκληρου του δικτύου, αφήνοντας ένα ως ενεργοποίηση ενός μεμονωμένου κόμβου (Jurafsky & Martin, 2023, σελ. 136).



Εικόνα 2.1. Ένα νευρωνικό δίκτυο με τρεις εισόδους

Μερικές από τις συναρτήσεις ενεργοποίησης f(σ) που έχουν προταθεί στη βιβλιογραφία είναι:

1. Γραμμική συνάρτηση :

$$f(\sigma) = \sigma$$

2. Συνάρτηση τύπου Perceptron:

$$f(\sigma) = \begin{cases} \sigma, & \sigma \geq 0 \\ 0 & \sigma < 0 \end{cases},$$

3. Δυαδική (δίτιμη) συνάρτηση με κατώφλι T:

$$f(\sigma) = \begin{cases} 1, & \sigma \geq T \\ 0 & \sigma < T \end{cases},$$

4. Σιγμοειδής (ή, Λογιστική) συνάρτηση:

$$f(\sigma) = \frac{1}{1+e^{-\sigma}}$$

5. Συνάρτηση υπερβολικής εφαπτομένης (tanh):

$$f(\sigma) = \frac{e^{z} - e^{-z}}{e^{z}+e^{-z}}$$

6. Συνάρτηση Relu:

$$f(\sigma) = ReLU(z) = max(z, 0)$$

2.2.1. Αρχιτεκτονική Νευρωνικών Δικτύων

Το πιο απλό νευρωνικό δίκτυο έχει έναν μόνο νευρώνα και είναι γνωστό ως μοντέλο απλού αισθητήρα, ή perceptron. Προτάθηκε το 1958 από τον Rosenblatt και είναι μαθηματικό μοντέλο αντίστοιχο ενός βιολογικού νευρώνα, ο μοναδικός δηλαδή νευρώνας του perceptron δέχεται πολλές εισόδους και έχει μία έξοδο. Ένα τέτοιο νευρωνικό δίκτυο φυσικά δεν μπορεί να εκπαιδευτεί σε περίπλοκα μοτίβα επομένως η αρχιτεκτονική ενός δικτύου πρέπει να γίνει πιο περίπλοκη. Να σημειωθεί ότι πλέον έχει επικρατήσει να χαρακτηρίζονται ως Perceptrons όλα τα τεχνητά νευρωνικά δίκτυα πρόσθιας τροφοδότησης που δεν περιέχουν στην αρχιτεκτονική τους κρυφά επίπεδα.

Πιο συγκεκριμένα (Γεωργούλη,2015;Παναγιωτακόπουλος,2022) ένα τεχνητό νευρωνικό δίκτυο οργανώνεται σε επίπεδα (layers). Ορισμένα δίκτυα έχουν και ενδιάμεσα επίπεδα τα οποία ονομάζονται κρυφά επίπεδα. Κάθε επίπεδο αποτελείται από έναν αριθμό μονάδων συνδεδεμένων μεταξύ τους, ώστε μία μονάδα να έχει συνδέσμους με πολλές άλλες μονάδες του ίδιου ή άλλου επιπέδου. Οι μονάδες είτε διεγείρουν είτε αναστέλλουν την ενεργοποίηση των άλλων μονάδων. Μία μονάδα λαμβάνει το σταθμισμένο άθροισμα όλων των εισόδων μέσω των συνδέσμων που καταλήγουν σε αυτήν και παράγει μέσω της συνάρτησης μετάβασης μία μοναδική έξοδο εάν το άθροισμα υπερβαίνει μία τιμή κατωφλίου. Το επίπεδο εισόδου (input layer) του δικτύου επικοινωνει με ένα ή περισσότερα κρυμμένα επίπεδα τα οποία με τη σειρά τους είναι συνδεδεμένα επίπεδο εξόδου (output layer) από το οποίο πηγάζει και η έξοδος. Άρα όταν δημιουργούμε την αρχιτεκτονική ενός νευρωνικού δικτύου πρέπει να καθορίσουμε τον αριθμό των ενδιάμεσων κρυφών επιπέδων του και των μονάδων ανά επίπεδο, καθώς και τον τρόπο που οι μονάδες συνδέονται μεταξύ τους. Ακόμα πρέπει να καθοριστεί η τιμή κατωφλιού και η συνάρτηση μετάβασης και τέλος οι κανόνες εκπαίδευσης που θα χρησιμοποιηθούν κατά τη διαδικασία της εκπαίδευσης. Ανάλογα με τον τρόπο που είναι συνδεδεμένες οι μονάδες μεταξύ τους, τα τεχνητά νευρωνικά χωρίζονται σε δίκτυα πρόσθιας τροφοδότησης (feed forward) και οπίσθιας τροφοδότησης (feed backward). Στα νευρωνικά δίκτυα πρόσθιας τροφοδότησης, οι μονάδες είναι οργανωμένες σε διαφορετικά επίπεδα, ώστε οι μονάδες του ενός επιπέδου να τροφοδοτούν τις μονάδες του επόμενου επιπέδου μέχρι να τροφοδοτηθούν και οι μονάδες του τελευταίου επιπέδου.Δεν υπάρχει έξοδος μονάδας ενός επιπέδου που να αποτελεί είσοδο μονάδας του ίδιου ή προηγούμενων επιπέδου. Αυτά τα νευρωνικά δίκτυα είναι τα δίκτυα οπισθοδιάδοσης (backpropagation). Στα νευρωνικά δίκτυα οπίσθιας τροφοδότησης ή αλλιώς στα ανατροφοδοτούμενα (recurrent), οι μονάδες ενός επιπέδου μπορούν να

τροφοδοτούν και μονάδες του ιδίου επιπέδου ή και προηγούμενων επιπέδων. Αν η ανατροφοδότηση αφορά κόμβους στο ίδιο επίπεδο, τότε τα δίκτυα καλούνται αυτοσυσχετιζόμενες μνήμες (autoassociated memories) διαφορετικά, καλούνται ετεροσυσχετιζόμενες μνήμες (heteroassociated memories) (Γεωργούλη,2015,σελ.160)

Τα τεχνητά νευρωνικά δίκτυα πολλών επιπέδων διαθέτουν τουλάχιστον ένα κρυφό επίπεδο. Οι κόμβοι τους μπορεί να είναι πλήρως συνδεδεμένοι (fully connected), όπου κάθε κόμβος του ενός επιπέδου συνδέεται με όλους τους κόμβους του επόμενου επιπέδου , ή μπορεί να είναι μερικώς συνδεδεμένοι (partially connected).

2.2.2. Εκπαίδευση με επίβλεψη - Κανόνας Δέλτα

Η πιο συνηθισμένη μέθοδος εκπαίδευσης ενός νευρωνικού δικτύου είναι η επιβλεπόμενη. Σε αυτήν, χρησιμοποιείται ένα σύνολο εκπαίδευσης δεδομένων (training set) *xn* μαζί με το αντίστοιχο σύνολο δεδομένων εξόδου *yn*. Κατά την εκπαίδευση ελέγχεται εάν η έξοδος *y* του νευρώνα για το δείγμα *x* είναι η αναμενόμενη. Αν είναι η σωστή, η διαδικασία της εκπαίδευσης συνεχίζει στο επόμενο δείγμα. Αν δεν είναι, τότε (Γεωργούλη,2015,σελ. 167) στην περίπτωση που η σωστή τιμή εξόδου είναι μεγαλύτερη από την τιμή που υπολόγισε ο νευρώνας, με τον κανόνα εκμάθησης αυξάνονται τα βάρη των εισόδων που είναι θετικές και μειώνονται τα βάρη των εισόδων που είναι αρνητικές. Στην περίπτωση που η σωστή τιμή εξόδου είναι μικρότερη μειώνονται τα βάρη των εισόδων που είναι θετικές και αυξάνονται τα βάρη των εισόδων που είναι αρνητικές. Η διαδικασία επαναλαμβάνεται μέχρι τα αποτελέσματα του νευρώνα να είναι σωστά για όλα τα δείγματα ή να μην μπορεί να βελτιωθεί περισσότερο η απόδοσή του.

Ο πιο γνωστός κανόνας εκμάθησης για την εκπαίδευση ενός νευρωνικού δικτύου είναι ο κανόνας Δέλτα. Χρησιμοποιείται για τον υπολογισμό των σφαλμάτων και την διόρθωση των τιμών των βαρών των εσωτερικών νευρώνων του δικτύου. Με τη βηματική συνάρτηση ενεργοποίησης, η μεταβολή των βαρών υπολογίζεται: $wi = wiold - d*(aj-ai)$, με *ai* την τρέχουσα έξοδο του νευρώνα i, *aj* την επιθυμητή έξοδο του νευρώνα i για το τρέχον δείγμα, *d* τον ρυθμό εκμάθησης (μεγαλύτερος του 0), ο οποίος καθορίζει τον ρυθμό σύγκλισης της μάθησης· ο μεγάλος ρυθμός μάθησης μπορεί να φέρει ως αποτέλεσμα γρηγορότερη σύγκλιση και ταλάντωση γύρω από τις βέλτιστες τιμών βαρών και ο μικρός ρυθμός μάθησης μπορεί να έχει ως αποτέλεσμα πιο αργή σύγκλιση και να οδηγήσει σε παγίδευση σε τοπικά ακρότατα, *wiold* το παλιό βάρος εισόδου από τον νευρώνα i και *wi* το νέο βάρος εισόδου του νευρώνα i (Γεωργούλη,2015,σελ. 168).

2.2.3. Σφάλμα

Κατά την εκπαίδευση, το σφάλμα μιας εξόδου ο ενός νευρώνα κ από μια επιθυμητή του έξοδο α, αναφορικά με ένα δείγμα p, ορίζεται η ποσότητα $Ek = (ak,p - ok,p)$. Το συνολικό σφάλμα για όλα τα παραδείγματα θα είναι το μέσο τετραγωνικό σφάλμα

$$E = \frac{1}{P}\sum_{p}(a_{k,p} - o_{k,p})^2$$

,

με το μέσο τετραγωνικό σφάλμα για όλους τους νευρώνες να είναι

$$E = \frac{1}{P * K}\sum_{p=1}^{P}\sum_{k=1}^{K}(a_{k,p} - o_{k,p})^2$$

.

Το συνολικό σφάλμα χρησιμοποιείται για τον τερματισμό της διαδικασίας εκπαίδευσης, όταν η τιμή του για όλα τα παραδείγματα και για όλους τους νευρώνες εξόδου πέσει κάτω από μια μικρή τιμή.

2.2.4. Οπισθοδιάδοση σφάλματος

Στα τεχνητά νευρωνικά δίκτυα πολλών επιπέδων η πιο συνηθισμένη μέθοδος επιβλεπόμενης εκπαίδευσης είναι η οπισθοδιάδοση σφάλματος (error back-propagation), ώστε να ελαχιστοποιείται το τετραγωνικό σφάλμα εξόδου στα δίκτυα (Καμπουρλάζος & Παπακώστας,2015,σελ. 21). Τα δίκτυα αυτά ονομάζονται επίσης και Backpropagation Τεχνητά Νευρωνικά Δίκτυα. Σύμφωνα με την (Γεωργούλη,2015,σελ. 169) η μέθοδος λειτουργεί με τον παρακάτω τρόπο.

Με εφαρμογή των συναρτήσεων μετάβασης σε κάθε μονάδα κρυφού ή εξωτερικού επιπέδου υπολογίζονται οι έξοδοι για κάθε δοσμένη είσοδο. Για κάθε μονάδα εξωτερικού επιπέδου λαμβάνονται υπόψη οι διαφορές μεταξύ του υπολογιζόμενου και του επιθυμητού αποτελέσματος και διαδίδονται προς τα πίσω στις μονάδες του κρυφών επιπέδων, έτσι ώστε να καθορίσουν τις απαραίτητες αλλαγές στα βάρη σύνδεσης μεταξύ των μονάδων. Τα σφάλματα των μονάδων του επόμενου επιπέδου μιας μονάδας είναι ανάλογα της τρέχουσας εισόδου της και των συντελεστών βαρύτητας που συνδέουν τη μονάδα με τις μονάδες του επόμενου επιπέδου. Στη συνέχεια εφαρμόζονται εκ νέου οι συναρτήσεις μετάβασης ώστε να υπολογιστεί το νέο σφάλμα. Έστω ότι έχουμε μία μονάδα εξόδου k ενός παραδείγματος p και θέλουμε να υπολογίσουμε το πραγματικό της σφάλμα Ek. Αυτό θα υπολογιστεί όπως και σε ένα τεχνητό νευρωνικό δίκτυο Perceptron $Ek = (akp - okp)$ και στη συνέχεια θα πολλαπλασιαστεί με την παράγωγο της συνάρτησης ενεργοποίησης της μονάδας k (uk). Αυτός είναι ο γενικευμένος κανόνας δέλτα και έτσι υπολογίζεται το προσαρμοσμένο σφάλμα νευρώνα: $\delta k = (akp - okp) \cdot g'$

(uk). Στις μονάδες του κρυφού επιπέδου i το αντίστοιχο σφάλμα υπολογίζεται από τα προσαρμοσμένα σφάλματα στις k μονάδες του επόμενου επιπέδου με τις οποίες η μονάδα συνδέεται με βάρη wik ως:

$$\delta_i = g'(u_i) \cdot \sum_1^k w_{ik} \cdot \delta_k$$

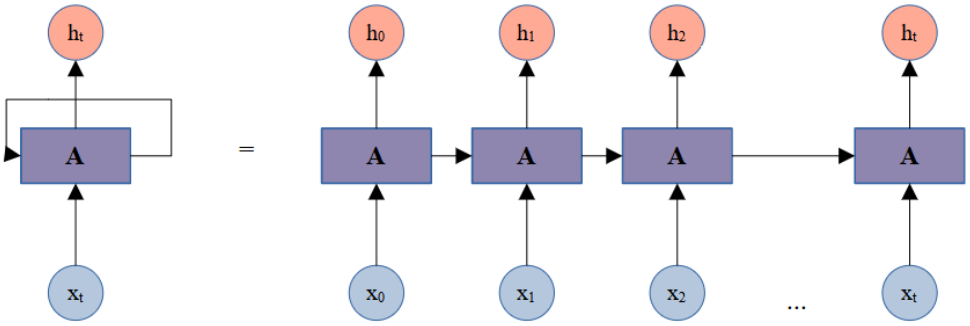
Η διαδικασία αυτή επαναλαμβάνεται σε πλήθος επαναλήψεων διόρθωσης σφάλματος έως ότου το σφάλμα φτάσει κάτω από ένα επιθυμητό όριο ή πραγματοποιηθεί ένας συγκεκριμένος αριθμός κύκλων εκπαίδευσης ή περάσει ένα συγκεκριμένο χρονικό διάστημα.

Για την αλλαγή των βαρών υπολογίζεται για κάθε μονάδα i το σφάλμα δi και η αλλαγή στα βάρη εισόδου σε όλους τους νευρώνες θα γίνει $\Delta wji = -d \cdot \delta i \cdot$ aj . Η αλλαγή στο βάρος από τον νευρώνα i στον νευρώνα j εξαρτάται λοιπόν από το σφάλμα του νευρώνα i, την έξοδο του νευρώνα j και το ρυθμό μάθησης (learning rate) d.

2.2.5. Επαναλαμβανόμενα αναδρομικά δίκτυα

Τα επαναλαμβανόμενα νευρωνικά δίκτυα (recurrent neural networks - RNNs) περιέχουν έναν κύκλο στις συνδέσεις του δικτύου τους και η τιμή κάποιας μονάδας εξαρτάται άμεσα ή έμμεσα από τις δικές του προηγούμενες εξόδους ως είσοδο. Συνοπτικά, όπως αναφέρουν και οι (Tarwani & Edem, 2017;Jurafsky & Martin,2023;Staudemeyer & Morris,2019) αυτό τα καθιστά πιο ισχυρά για τη μοντελοποίηση εισόδων ακολουθιών καθώς δημιουργείται μια εσωτερική κατάσταση του δικτύου που του επιτρέπει να επιδεικνύει δυναμική συμπεριφορά. Η αναδρομική σύνδεση δημιουργεί μια μορφή εσωτερικής μνήμης και έτσι το νευρωνικό δίκτυο διατηρεί πληροφορίες από τις προηγούμενες καταστάσεις της επεξεργασίας της ακολουθίας. Πιο συγκεκριμένα, στα RNNs ένα διάνυσμα εισόδου που αντιπροσωπεύει την τρέχουσα είσοδο xt, πολλαπλασιάζεται με έναν πίνακα βάρους και στη συνέχεια διέρχεται από μια μη γραμμική συνάρτηση ενεργοποίησης ώστε να υπολογιστούν οι τιμές για ένα στρώμα κρυφών μονάδων. Το στρώμα αυτό αντιπροσωπεύει τη μνήμη του δικτύου από προηγούμενα χρονικά βήματα. Τέλος, το κρυφό επίπεδο χρησιμοποιείται για τον υπολογισμό της αντίστοιχης εξόδου yt, περιλαμβάνοντας μια επαναλαμβανόμενη σύνδεση ως μέρος της εισόδου του. Η τιμή ενεργοποίησης του κρυφού στρώματος δηλαδή εξαρτάται από την τρέχουσα είσοδο καθώς και από την τιμή ενεργοποίησης του κρυφού στρώματος από το προηγούμενο χρονικό βήμα (Jurafsky & Martin,2023,σελ.

186).Τα απλά επαναλαμβανόμενα δίκτυα αναφέρονται και ως δίκτυα Elman (Elman, 1990).



Εικόνα 2.2. Ένα απλό RNN δίκτυο.

Έστω ότι έχουμε μια είσοδο xt του δικτύου RNN και την έξοδό του ht, όπως φαίνεται στην Εικόνα 2.2.(Tarwani & Edem, 2017). Το δίκτυο μπορεί να τροφοδοτηθεί με λέξεις από μια πρόταση ή και χαρακτήρες μιας συμβολοσειράς, ως η είσοδος xt, ώστε να προκύψει η έξοδος ht. Έπειτα η έξοδος αυτή θα συγκριθεί με τα δεδομένα δοκιμής, τα οποία είναι συνήθως ένα υποσύνολο των αρχικών δεδομένων. Μαζί με το ποσοστό σφάλματος εφαρμόζεται η τεχνική Back Propagation Through Time , με την οποία ελέγχεται ξανά το δίκτυο και προσαρμόζονται τα βάρη για να εκπαιδευτεί το δίκτυο. Ο αλγόριθμος BPTT λειτουργεί με βάση το γεγονός ότι για μια πεπερασμένη χρονική περίοδο υπάρχει ένα Feed Forward Neural Network με την ίδια συμπεριφορά για κάθε RNN και για να αποκτηθεί αυτό το FFNN, πρέπει να ξεδιπλωθεί το RNN εγκαίρως Το αντίστοιχο νευρωνικό δίκτυο τροφοδοσίας απαιτεί ξεχωριστό στρώμα για κάθε χρονικό βήμα με τα ίδια βάρη για όλα τα επίπεδα και στην περίπτωση που τα βάρη είναι ίδια με το RNN, και τα δύο δίκτυα δείχνουν την ίδια συμπεριφορά (Staudemeyer & Morris,2019). Παρατηρείται ότι με το πέρας ορισμένων χρονικών βημάτων και με κάθε χρονικό βήμα μετέπειτα, τα σήματα σφάλματος (error signals) που διαδίδονται προς τα πίσω τείνουν είτε να αυξάνονται είτε να συρρικνώνονται. Όταν το σήμα σφάλματος αυξάνεται τα βάρη ταλαντεύονται, ενώ αν το σφάλμα εξαφανιστεί, η εκμάθηση θα χρειαστεί υπερβολικά πολύ χρόνο ή δεν θα πραγματοποιηθεί καθόλο((Staudemeyer & Morris,2019).Παρατηρείται ότι με το πέρας ορισμένων χρονικών βημάτων και με κάθε χρονικό βήμα μετέπειτα, τα σήματα σφάλματος (error signals) που διαδίδονται προς τα πίσω τείνουν είτε να αυξάνονται είτε να συρρικνώνονται. Όταν το σήμα σφάλματος αυξάνεται τα βάρη ταλαντεύονται, ενώ αν το σφάλμα εξαφανιστεί, η εκμάθηση θα

χρειαστεί υπερβολικά πολύ χρόνο ή δεν θα πραγματοποιηθεί καθόλου (Staudemeyer & Morris,2019). Η εξαφάνιση του σφάλματος μπορεί να αντιμετωπιστεί με τη μέθοδο της Μακροχρόνιας βραχυπρόθεσμης μνήμη (LSTM) η οποία γεφυρώνει ελάχιστες χρονικές καθυστερήσεις άνω των 1.000 διακριτών χρονικών βημάτων. Στην μέθοδο αυτή γίνεται η χρήση καρουζέλ σταθερών σφαλμάτων (constant error carousels) με τα οποία επιβάλλεται σταθερή ροή σφάλματος με ειδικά κελιά και η πρόσβαση στα κελιά αυτά γίνεται με πολλαπλασιαστικές μονάδες πύλης.

Τα αμφίδρομα LSTM (Bidirectional LSTM) στοχεύουν στην ανάλυση για κάθε δεδομένο σημείο μιας ακολουθίας, όχι μόνο προς μία κατεύθυνση, αυτή του παρελθόντος, αλλά προς αυτή του μέλλοντος. Σε υψηλό επίπεδο , το ότι το RNN είναι αμφίδρομο σημαίνει ότι η είσοδος παρουσιάζεται προς τα εμπρός και προς τα πίσω σε δύο ξεχωριστά δίκτυα LSTM, τα οποία είναι και τα δύο συνδεδεμένα στο ίδιο επίπεδο εξόδου (Staudemeyer & Morris,2019).

Έτσι, όσον αφορά το πεδίο της Επεξεργασίας Φυσικής Γλώσσας, υπάρχει η δυνατότητα η απόφασή του να εξαρτάται από πληροφορίες που προέρχονται από εκατοντάδες λέξεις κειμενικών δεδομένων στο παρελθόν. Για τον λόγο αυτό τα δίκτυα αυτά μπορούν να χρησιμοποιηθούν σε διαδικασίες μοντελοποίησης της γλώσσας καθώς και για την αλληλουχία εργασιών μοντελοποίησης όπως η επισήμανση μερών του λόγου, οι εργασίες ταξινόμησης κειμένου, η ανάλυση συναισθήματος κ.α. Περισσότερα για την Επεξεργασίας Φυσικής Γλώσσας θα δούμε στο Κεφάλαιο 3.

2.2.6. Μοντέλο Sequence-to-sequence (Seq2Seq)

Το μοντέλο εκμάθησης Sequence to Sequence (ακολουθία σε ακολουθία) ή αλλιώς Seq2Seq εμφανίστηκε πρώτη φορά από την Google για αυτόματη μετάφραση. Μία ακολουθία ενός τομέα, όπως μια πρόταση στα αγγλικά, μετατρέπεται σε ακολουθία ενός άλλου τομέα, όπως η μετάφραση της αγγλικής πρότασης στα Ισπανικά. Δεν λαμβάνεται υπόψη μόνο η τρέχουσα εισαγωγή/λέξη κατά τη διαδικασία μετάφρασης αλλά και η γειτονική της. Προκειμένου να ξεκινήσει η πρόβλεψη του στόχου απαιτείται ολόκληρη η ακολουθία εισόδου. Για αυτό τον λόγο το μοντέλο περιέχει τα δύο βασικά του στοιχεία, έναν κωδικοποιητή και έναν αποκωδικοποιητή και εκπαιδεύεται χρησιμοποιώντας ένα σύνολο δεδομένων ζευγών εισόδου-εξόδου, όπου η είσοδος είναι μια ακολουθία διακριτικών (tokens) και η έξοδος είναι επίσης μια ακολουθία διακριτικών(tokens). Εκπαιδεύεται ώστε να μεγιστοποιηθεί η πιθανότητα της σωστής

ακολουθίας εξόδου δεδομένης της ακολουθίας εισόδου και χρησιμοποιεί συνήθως επαναλαμβανόμενα αναδρομικά νευρωνικά δίκτυα. Το στοιχείο που λειτουργεί ως "κωδικοποιητής" έχει ένα επίπεδο RNN το οποίο επεξεργάζεται την ακολουθία εισόδου και επιστρέφει τη δική του εσωτερική κατάσταση. Απορρίπτονται οι έξοδοι του κωδικοποιητή RNN και ανακτάται μόνο αυτή η κατάσταση, η οποία λειτουργεί ως το πλαίσιο ή η προετοιμασία του αποκωδικοποιητή στο επόμενο βήμα (Chollet). Ο αποκωδικοποιητής έχει και αυτός ένα επίπεδο RNN και εκπαιδεύεται ώστε να προβλέπει τους επόμενους χαρακτήρες της ακολουθίας στόχου(target), δεδομένων των προηγούμενων χαρακτήρων αυτής. Ουσιαστικά, ο αποκωδικοποιητής μαθαίνει να δημιουργεί targets[t+1...] με δεδομένους targets[...t], υπό τον όρο της ακολουθίας εισόδου. Η μετατροπή των αλληλουχιών-στόχων στις ίδιες ακολουθίες με αντιστάθμιση κατά ένα χρονικό βήμα στο μέλλον ονομάζεται ως εκπαιδευτική διαδικασία «εξαναγκασμός δασκάλου» ("teacher forcing"). Αν δεν χρησιμοποιηθεί η διαδικασία teacher forcing, η εκπαίδευση γίνεται με επανεισαγωγή των προβλέψεων του αποκωδικοποιητή στον αποκωδικοποιητή.

3. Επεξεργασία Φυσικής Γλώσσας

Η επεξεργασία φυσικής γλώσσας αποτελεί μια συλλογή από υπολογιστικές τεχνικές που ως στόχο έχουν την ανάλυση και αναπαράσταση ανθρώπινων γλωσσών και αποτελεί αντικείμενο της υπολογιστικής γλωσσολογίας. Με την επεξεργασία της φυσικής γλώσσας εκτελούνται διάφορες εργασίες, όπως η αυτόματη σύνοψη, η μετάφραση κειμένου, η αναγνώριση ονοματικών οντοτήτων, η εξαγωγή σχέσεων, η σημασιολογική ανάλυση, η αναγνώριση ομιλίας, η τμηματοποίηση θεμάτων και η ανάλυση συναισθήματος. (Tarwani & Edem, 2017). Άλλες βασικές εφαρμογές της είναι η εξαγωγή πληροφοριών, η εξόρυξη δεδομένων,η αναγνώριση μερών του λόγου και η συντακτική ανάλυση,η ταξινόμηση κειμένου σε κατηγορίες,η ευρετηρίαση καθώς και η αναζήτηση μεγάλων κειμένων.

Η γλωσσολογία αποτελείται από δύο κλάδους, την υπολογιστική γλωσσολογία και τη θεωρητική γλωσσολογία (Chowdhary,2020):

1. Η θεωρητική γλωσσολογία εστιάζει κυρίως στη γλωσσική απόδοση και τη γραμματική ικανότητα , στο πώς δηλαδή οι άνθρωποι αποδέχονται ορισμένες προτάσεις ως ορθές ακολουθώντας τους γραμματικούς κανόνες και άλλες ως μη ορθές.
2. Η υπολογιστική γλωσσολογία ασχολήθηκε με την ανάπτυξη αλγορίθμων για το χειρισμό της φυσικής γλώσσας ως εισόδου την κατανόηση και τη δημιουργία φυσικής

Μία από τις μεγαλύτερες δυσκολίες στην επεξεργασία φυσικής γλώσσας είναι η διφορούμενη ερμηνεία που προκαλεί ασάφεια στη γλώσσα (ambiguity of language) (Γεωργούλη,2015). Κάτι τέτοιο συμβαίνει σε πολλά επίπεδα. Σε επίπεδο σύνταξης, μία πρόταση μπορεί να είναι μεν συντακτικά ορθή, αλλά να επιδέχεται πάνω από μία διερμηνεία, ανάλογα με την συντακτική της ανάλυση. Σε επίπεδο λεξιλογικό (ambiguity at lexical level), μία λέξη μπορεί να έχει διφορούμενο νόημα. Σε επίπεδο αναφορικό (ambiguity at referential level), μια πρόταση μπορεί να μην είναι σαφές σε ποιον, πού ή σε τι αναφέρεται. Σε επίπεδο σημασιολογικό (ambiguity at semantic level), μια πρόταση μπορεί να επιδέχεται τουλάχιστον δύο ή περισσότερες διαφορετικές ερμηνείες, ενώ διατηρείται η ίδια συντακτική ανάλυση. Σε επίπεδο πραγματολογικό (pragmatic level), μία πρόταση μπορεί να είναι ασαφής λαμβάνοντας υπόψη το πλαίσιο κειμένου που την περιέχει.
Στη συνέχεια θα δούμε τα βασικά στάδια για την επεξεργασία και τρόπους αντιμετώπισης των παραπάνω δυσκολιών.

Με τη χρήση Τεχνητής Νοημοσύνης και Μηχανικής Μάθησης το γραπτό κείμενο αναλύεται στο σημασιολογικό του περιεχόμενο και γίνεται εφικτή η επεξεργασία του (από τον ηλεκτρονικό υπολογιστή). Αρχικά, εκτελείται η προεπεξεργασία των κειμενικών δεδομένων, είτε με τη χρήση αλγορίθμων κατασκευασμένων για την εκάστοτε περίπτωση, είτε με χρήση Νευρωνικών Δικτύων και άλλων τεχνικών Μηχανικής Μάθησης.

Τα βασικότερα βήματα προεπεξεργασίας είναι τα εξής:

1. Τοκενοποίηση (Tokenization): Το κείμενο χωρίζεται σε επιμέρους τμήματα (tokens). Μία παράγραφος μπορεί να χωριστεί σε προτάσεις, μία πρόταση σε λέξεις, καθώς και μία λέξη στους επιμέρους χαρακτήρες της.
2. Κανονικοποίηση (Normalization): Έπειτα εκτελείται η κανονικοποίηση (normalization) του κειμένου, όπως για παράδειγμα η μετατροπή όλων των χαρακτήρων σε πεζά γράμματα.
3. Αφαίρεση Θορύβου (Noise Removal): Στο στάδιο αυτό αφαιρούνται λέξεις οι οποίες δεν συμβάλλουν στο νόημα του κειμένου, όπως οι προθέσεις, οι αντωνυμίες κ.ο.κ.
4. Λημματοποίηση (lemmatization ή stemming): Σκοπός αυτών των τεχνικών είναι να μειωθούν οι λέξεις που παράγονται από την ίδια ρίζα-βάση, όπως για παράδειγμα οι λέξεις car, cars, car's, cars' οι οποίες πηγάζουν από τη λέξη car. Η λημματοποίηση είναι πιο ακριβής μέθοδος αφού χρησιμοποιεί λεξιλόγια και μορφολογική ανάλυση ώστε να εντοπιστεί η ρίζα μιας λέξης , δηλαδή το λήμμα (lemma).
5. Part of Speech (PoS) tagging: Με αυτή την τεχνική οι λέξεις αναθέτονται σε κλάσεις όπως ουσιαστικά, ρήματα, επιρρήματα, επίθετα κ.ο.κ. Αυτό βελτιώνει και το στάδιο της λημματοποίησης που αναφέρθηκε παραπάνω.
6. Ορισμός εξαρτήσεων (Dependency Parsing): Με τη μέθοδο αυτή ορίζονται οι σχέσεις που υπάρχουν μεταξύ των λέξεων και των φράσεων σε μία πρόταση του κειμένου, όπως για παράδειγμα σε ποιό ουσιαστικό(ή φράση) αναφέρεται μία αντωνυμία.

3.1. Στάδια επεξεργασίας φυσικής γλώσσας.

Υπάρχουν τρία στάδια στην επεξεργασίας φυσικής γλώσσας:

1. Η συντακτική ανάλυση (syntactic analysis), η οποία εξετάζει τη δομή των προτάσεων. Κομμάτι της ανάλυσης αυτής αποτελεί και η γραμματική ανάλυση μιας πρότασης και προσφέρει μια δηλωτική αναπαράσταση των συντακτικών στοιχείων σχετικά με τη γλώσσα,δηλαδή έναν συντακτικό αναλυτή που συγκρίνει την πρόταση που εισάγεται με τη γραμματική (parsing) που υπάρχει διαθέσιμη στην εκάστοτε εφαρμογή (Γεωργούλη,2015)
2. Η σημασιολογική ανάλυση (semantic analysis), κατά την οποία πραγματοποιείται μετατροπή των προτάσεων σε εσωτερικές δομές αναπαράστασης γνώσης, χρησιμοποιώντας τη νοηματική σημασία των λέξεων. Για να επιτευχθεί κάτι τέτοιο, απαιτούνται οι εξελιγμένες Γραμματικές Οριστικών Προτάσεων. Σημαντικότερο πρόβλημα αποτελεί η ασάφεια σε επίπεδο λεκτικό που προκύπτει από την πολυσημία (ambiguity) των λέξεων.(Γεωργούλη) Ένας τρόπος αντιμετώπισης της ασάφειας αυτής είναι το να λαμβάνονται υπόψη χαρακτηριστικά της κλίσης των ουσιαστικών και ρημάτων: π.χ. γένος, αριθμός, πτώση των ουσιαστικών, πρόσωπα ρημάτων. Έτσι γίνεται ευκολότερη και η μετάφραση της φυσικής γλώσσας που έχουμε στην είσοδο στην επίσημη γλώσσα κάποιου συστήματος επεξεργασίας(Chowdhary,2020).
3. Η πραγματολογική ανάλυση στοχεύει στην κατανόηση του κειμένου και στον χειρισμό διαλόγων. Κάθε πρόταση επιχειρείται να ενταχθεί στο γενικότερο νοηματικό πλαίσιο των συμφραζομένων της (Γεωργούλη,2015). Για να συμβεί αυτό πρέπει να ληφθούν υπόψη οι συνθήκες μέσα στις οποίες μια πρόταση έχει λεχθεί καθώς και ποιες λέξεις της, όπως αντωνυμίες κτλ αναφέρονται σε ονόματα ή οντότητες άλλων προτάσεων. Πρέπει να υπάρχει γενική γνώση για τον τρόπο που λειτουργεί ο κόσμος μέσα στον οποίο αναφέρεται η πρόταση, πιθανά σενάρια που μπορεί να διαδραματιστούν, να μπορούν να γίνουν εύλογα συμπεράσματα με βάση τη γνώση αυτή από ένα σύστημα επεξεργασίας φυσικής γλώσσας (Γεωργούλη,2015).

3.2. Συντακτική Ανάλυση

Από εδώ και πέρα κάνουμε την παραδοχή ότι η γλώσσα στην οποία θα αναφερόμαστε είναι η **αγγλική**, καθώς για αυτή αναπτύχθηκε η εφαρμογή και στηρίχθηκε η μεθοδολογία, όπως θα δούμε αναλυτικά παρακάτω.
Έχοντας μία δεδομένη πρόταση, με την συντακτική ανάλυση θα πρέπει να προσδιορίζονται τα συστατικά της μέρη, δηλαδή το ρήμα, το υποκείμενο και το αντικείμενό του ρήματος, οι τροποποιητικές λέξεις (modifiers) καθώς και οι φράσεις που τροποποιούν.

Τα μέρη του λόγου χωρίζονται σε δύο μεγάλες κατηγορίες,την κλειστή τάξη και την ανοιχτή τάξη (Jurafsky & Martin,2023, σελ. 160). Στις κλειστές τάξεις ανήκουν τα μέρη του λόγου με σταθερή ιδιότητα, λόγου χάρη οι προθέσεις, λέξεις δηλαδή που επινοούνται σπάνια (it, and, you, of, κ.ο.κ) Στις ανοιχτές τάξεις ανήκουν τα μέρη του λόγου που υπάρχει η δυνατότητα να δημιουργηθούν/επινοηθούν νέα, όπως τα ρήματα, τα επιρρήματα, τα ουσιαστικά (συμπεριλαμβανομένων των κύριων ουσιαστικών), τα επίθετα, τα επιφωνήματα.

Τα κυριότερα μέρη του λόγου είναι τα εξής (Jurafsky & Martin,2023, σελ. 161). :

1. Nouns (Ουσιαστικά). Είναι συνήθως λέξεις για άτομα, μέρη ή πράγματα. Τα κοινά ουσιαστικά περιλαμβάνουν συγκεκριμένους όρους όπως woman(γυναίκα),teacher(δάσκαλος), doctor(γιατρός), book (βιβλίο), tree (δέντρο) κ.ο.κ, αφαιρέσεις(λέξεις που αναφέρονται σε έννοιες) όπως algorithm, peace κ.ο.κ., και όροι που μοιάζουν με ρήματα. Εμφανίζονται με προσδιοριστές(determiners) αλλά και στον πληθυντικό (teachers, books, trees κ.ο.κ.). Μπορούν να είναι μετρήσιμα αλλά και μη μετρήσιμα. Τέλος, τα κύρια ουσιαστικά (proper nouns) είναι ονόματα συγκεκριμένων προσώπων ή οντοτήτων, όπως πόλεων, επιχειρήσεων κτλ.
2. Verbs(Ρήματα):αναφέρονται σε ενέργειες και διαδικασίες, συμπεριλαμβανομένων των κύριων ρημάτων. Κλίνονται, και η κατάληξή τους αλλάζει συνήθως στο τρίτο πρόσωπο.
3. Adjectives (επίθετα): λέξεις που περιγράφουν τις ιδιότητες των ουσιαστικών όπως healthy (υγιής), cheerful(χαρούμενος),narrow(στενός) κ.ο.κ.

4. Adverbs(επιρρήματα): Τροποποιούν άλλες λέξεις, συνήθως ρήματα αλλά και φράσεις ρημάτων (verb phrases). Μπορεί να είναι τοπικά,χρονικά, κατεύθυνσης ή να προσδιορίζουν την έκταση κάποιας ενέργειας.
5. Interjections (επιφωνήματα - oh, hey, alas, uh, um): Μικρή ανοιχτή τάξη μερών λόγου που περιλαμβάνει και χαιρετισμούς και απαντήσεις ερωτήσεων.
6. Adpositions (προθέσεις): Εμφανίζονται πριν από τα ουσιαστικά. Μπορούν να υποδεικνύουν χωρικές ή χρονικές σχέσεις, είτε κυριολεκτικές είτε μεταφορικές.
7. Particle(μόριο): Τα μόρια είναι σαν προθέσεις ή επιρρήματα και χρησιμοποιούνται σε συνδυασμό με ένα ρήμα. resembles a preposition or an adverb and is used in combination with a verb. Particles often have extended meanings that aren't quite the same as the prepositions they resemble, as in the particle over in she turned the paper over. Ένα ρήμα και ένα μόριο που λειτουργούν ως ενιαία μονάδα ονομάζονται φραστικό ρήμα (phrasal verb).
8. Determiners(προσδιοριστικά):Δείχνουν την αρχή μιας φράσης ουσιαστικού (noun phrase). Σε αυτή την κατηγορία ανήκουν και τα άρθρα, όπως a, an κ.ο.κ.
9. Conjunctions(σύνδεσμοι): Ενώνουν δύο φράσεις, προτάσεις κτλ.
10. Pronouns(αντωνυμίες):Με αυτές γίνεται αναφορά σε μια οντότητα ή ένα γεγονός. Υπάρχουν οι προσωπικές(you, she, I, it, me, etc.), οι κτητικές (my, your, his, her, its, one's, our, their)καθώς και οι αντωνυμίες wh- (what, who, whom, whoever).
11. Auxiliary verbs(βοηθητικά ρήματα):Σηματοδοτούν σημασιολογικά χαρακτηριστικά ενός κύριου ρήματος. Σε αυτή την κατηγορία ανήκουν και τα ρήματα be, do,have καθώς και τα modal verbs(εγκλιτικά ρήματα) Τα βοηθητικά ρήματα στα αγγλικά περιλαμβάνουν το ρήμα copula be, τα δύο ρήματα do και have, μορφές, καθώς και εγκλιτικά ρήματα (modal verbs).

Με τη συντακτική ανάλυση ομαδοποιούνται οι παραπάνω λέξεις σε μεγαλύτερες ενότητες, ανάλογα με το πως συμπεριφέρονται και ποιο συγκεκριμένο μέρος του λόγου έχουν σαν «κύριο» (head). Έτσι δημιουργούνται οι φράσεις.
Προκειμένου λοιπόν να εξεταστεί ποια είναι η συντακτική δομή μιας πρότασης (και κατ επέκταση αν είναι ορθή η δομή αυτή) πραγματοποιείται η γραμματική ανάλυσή της (parsing). Η πρόταση μετατρέπεται σε μια ιεραρχική δομή ώστε να φανούν τα συστατικά της μέρη και έπειτα χρησιμοποιείται μια γραμματική και ένας συντακτικός αναλυτής ώστε να συγκριθεί η πρόταση που εισάγεται με τη γραμματική και να κατασκευαστεί το δέντρο ανάλυσής της (parsing tree). Αν υπάρχει αμφισημία στις λέξεις, δοκιμάζεται η συντακτική ανάλυση διαφόρων ερμηνειών των λέξεων μέσα από το λεξικό και

επιλέγεται εκείνη που ταιριάζει καλύτερα τόσο στη συντακτική όσο και στη σημασιολογική και πραγματολογική ανάλυση της πρότασης.

3.2.1.Γραμματικές

Η Γραμματική μιας γλώσσας αποτελεί τον μηχανισμό με τον οποίο μπορούν να παραχθούν όλες οι δυνατές συμβολοσειρές μιας γλώσσας και ορίζει την κατασκευή του συντακτικού δέντρου. Ένα Αλφάβητο είναι ένα πεπερασμένο σύνολο συμβόλων και ως Γλώσσα ορίζεται το σύνολο των συμβολοσειρών που παράγεται από ένα αλφάβητο. Μια Τυπική Γραμματική ή (Formal Grammar) είναι η τετράδα $G = (N, \Sigma, P, S)$ όπου (Σακελλαρίου et. al, 2015, σελ. 286):

1. Ν είναι ένα πεπερασμένο σύνολο από μη-τερματικά σύμβολα (non-terminals) και εκφράζουν αφαιρέσεις ,
2. Σ είναι ένα πεπερασμένο σύνολο από τερματικά σύμβολα (terminals) το οποίο είναι διαφορετικό από το Ν και αντιστοιχεί σε λέξεις της γλώσσας,
3. Ρ είναι ένα πεπερασμένο σύνολο από κανόνες παραγωγής (production rules) της μορφής $\alpha \rightarrow \beta$ όπου το α $(N \cup \Sigma)^* N (N \cup \Sigma)^*$ και το β $(N \cup \Sigma)^*$, είναι δηλαδή της μορφής $(N \cup \Sigma)^* N (N \ \Sigma)^* \rightarrow (N \cup \Sigma)^*$. Το σύμβολο → ερμηνεύεται ως "αποτελείται από" ή "παράγει". Το σύμβολο * (ονομάζεται και κλείσιμο Kleene) υποδηλώνει ότι υπάρχουν μηδέν ή περισσότερες επαναλήψεις από το σύμβολο αυτό. Παραδείγματος χάρη, ο κανόνας $\alpha \rightarrow \beta \gamma^*$ συμβολίζει πως "η έκφραση α παράγει συμβολοσειρές που ξεκινούν με το β και αποτελούνται από μηδέν ή περισσότερα γ."
4. $\Sigma \in N$ είναι το αρχικό σύμβολο (initial symbol).

3.2.2. Γραμματικές χωρίς συμφραζόμενα (Context Free Grammars)

Αποτελούνται από ένα σύνολο κανόνων οι οποίοι θέτουν τον τρόπο με τον οποίο τα σύμβολα της γλώσσας μπορούν να ομαδοποιηθούν και να ταξινομηθούν σε συνδυασμό με το λεξικό της γλώσσας. Οι κανόνες αυτοί μπορούν να ενσωματωθούν ιεραρχικά,έτσι ώστε να συνδυάζονται με προηγούμενους κανόνες που εκφράζουν γεγονότα σχετικά με το λεξικό.

Με αυτό τον τρόπο μπορούν να δημιουργηθούν προτάσεις ή να ανατεθεί μιας δομή σε μια δεδομένη πρόταση. Για παράδειγμα, οι κανόνες(Jurafsky & Martin,2023, σελ. 359):

NP → Det Nominal
NP → Proper Noun
Nominal → Noun | Nominal Noun

εκφράζουν ότι ένα NP (Noun Phrase- ονομαστική φράση) μπορεί να αποτελείται είτε από ένα Proper Noun, είτε από έναν προσδιορισμό (Det) ακολουθούμενο από ένα Nominal (κατηγορία που χρησιμοποιείται για την ομαδοποίηση ουσιαστικών και επιθέτων με βάση κοινές ιδιότητες), ενώ ένα Nominal μπορεί να αποτελείται από ένα ή περισσότερα Nouns.
Μπορούν να συνδυαστούν με τους κανόνες:

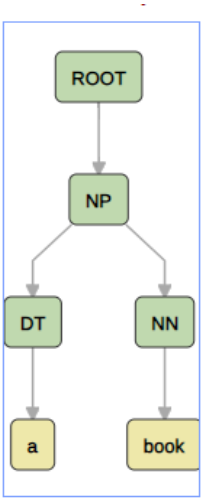
Det → a
Det → the
Noun → book

Στο δεξί μέρος του κανόνα υπάρχει μια ταξινομημένη λίστα ενός ή περισσότερων τερματικών και μη τερματικών, ενώ στα αριστερά υπάρχει ένα μόνο μη τερματικό σύμβολο που εκφράζει κάποια γενίκευση, δηλαδή ένα μέρος του λόγου.

3.2.3. Συντακτικό Δέντρο

Τα συντακτικά δέντρα αποτελούν το αποτέλεσμα της συντακτικής ανάλυσης, του προσδιορισμού δηλαδή των συστατικών στοιχείων της πρόταση όπως αναφέραμε και παραπάνω. Στη ρίζα του δέντρου βρίσκεται το αρχικό σύμβολο, κάθε ενδιάμεσος κόμβος αντιπροσωπεύει ένα μη-τερματικό σύμβολο της γραμματικής και τέλος, τα φύλλα του δέντρου αντιπροσωπεύουν τα τερματικά σύμβολα της γραμματικής. Το δέντρο κατασκευάζεται από τον συντακτικό αναλυτή με τη χρήση των κανόνων γραμματικής και σε κάθε βήμα επιλέγεται ένας κόμβος που εκπροσωπεί ένα αριστερό μη-τερματικό σύμβολο ενός κανόνα και από κάτω διασπάται σε κόμβους που αντιπροσωπεύουν τα σύμβολα του δεξιού μέρους του κανόνα (Γεωργούλη,2015,σελ.249).

Σύμφωνα με το παράδειγμα κανόνων που είδαμε παραπάνω μπορούμε να δημιουργήσουμε το εξής συντακτικό δέντρο, στο οποίο η ρίζα αποτελεί το αρχικό σύμβολο NP, οι ενδιάμεσοι κόμβοι τα μη τερματικά Det και Nominal από τα οποία μπορεί να αποτελείται ένα NP και τέλος, τα φύλλα με τα τερματικά σύμβολα της γραμματικής, δηλαδή τις λέξεις a και book. Το συντακτικό δέντρο της Noun Phrase φαίνεται στην Εικόνα 3.1., το οποίο προέκυψε με την χρήση του online εργαλείου http://stanza.run/.



Εικόνα 3.1 Συντακτικό δέντρο της Noun Phrase a book.

Το συγκεκριμένο παράδειγμα είναι πολύ απλοϊκό, αντικατοπτρίζει όμως τον τρόπο λειτουργίας των συντακτικών δέντρων τα οποία εφαρμόζονται φυσικά σε πολύ πιο περίπλοκες δομές κανόνων.

3.2.4. Ιεραρχία Chomsky

Ο Noam Chomsky όρισε τέσσερις τύπους γλωσσών που η διαφορά τους έγκειται στο είδος των κανόνων και στους τύπους εκφράσεων που παράγουν. Στην Εικόνα 3.2 φαίνονται σχηματικά οι γραμματικές ιεραρχίας Chomsky.

Πιο συγκεκριμένα (Σακελλαρίου et al, σελ. 287):

1. Οι γλώσσες Τύπου-0 ορίζονται από γραμματικές που αναγνωρίζονται από μία Μηχανή Turing και παράγουν όλες τις γλώσσες που μπορούν να αναγνωριστούν από αυτές. Οι γλώσσες αυτές ονομάζονται Αναδρομικά Απαριθμήσιμες Γλώσσες (Recursively Enumerable ή Unrestricted).

   Οι κανόνες είναι της μορφής:

   ● $\alpha \rightarrow \beta$

   όπου:

   ● α και β είναι τυχαίες συμβολοσειρές ενός λεξιλογίου, δηλαδή

   $\alpha,\beta \ (N \cup \Sigma)^*$ και

   ● $\alpha \neq \varepsilon$

2. Οι γλώσσες Τύπου-1 ορίζονται από γραμματικές που αναγνωρίζονται από μία μη-ντετερμινιστική μηχανή Turing με πεπερασμένο μήκος ταινίας (Linear-bounded Automata). Οι γλώσσες αυτές χειρίζονται διαφορετικά συμφραζόμενα επειδή έχουν γνώση του τι ειπώθηκε ήδη και δεν έχουν προκαθορισμένο τι θα ειπωθεί μετά (Context Sensitive). Οι κανόνες είναι της μορφής:

   ● $\alpha A \beta \rightarrow \alpha B \beta$, ή

   ● $S \rightarrow \varepsilon$

   όπου:

   ● $\alpha,\beta,B \ (N \cup \Sigma)^*$

   ● $A, S \ N$, με S το αρχικό σύμβολο της γραμματικής.

   ● $B \neq \varepsilon$, δηλαδή πρέπει να περιέχει τουλάχιστον ένα στοιχείο (τερματικό ή μη τερματικό).

3. Οι γλώσσες Τύπου-2 ορίζονται από γραμματικές που αναγνωρίζονται από Αυτόματα Στοίβας (Push-down Automata). Οι γλώσσες αυτές χειρίζονται διαφορετικά συμφραζόμενα αλλά μόνο τα πιο πρόσφατα, καθώς γνωρίζουν τι ειπώθηκε μόλις πριν αλλά δεν έχουν όλην την εικόνα για το τι ειπώθηκε γενικά (Context Free). Η φυσική γλώσσα έχει τη δομή μιας γλώσσας Τύπου-2 και αποτελούν την θεωρητική βάση των πιο πολλών γλωσσών προγραμματισμού (Παναγιωτακόπουλος et. al,2022, σελ.318). Οι κανόνες είναι της μορφής:

   ● $A \rightarrow \alpha$ όπου:

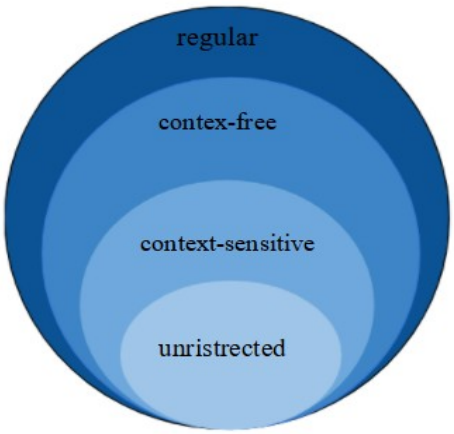
   ● $A \ N$ και

   ● $\alpha \in (N \cup \Sigma)^*$

4. Οι γλώσσες Τύπου-3 ορίζονται από γραμματικές που αναγνωρίζονται από Αυτόματα Πεπερασμένων Καταστάσεων (Finite State Automata). Οι γλώσσες αυτές ονομάζονται Κανονικές (Regular) και δεν έχουν γνώση για το τι έχει ήδη ειπωθεί. Είναι οι πιο απλές γλώσσες και οι κανόνες τους είναι της μορφής:
    - $A \rightarrow \varepsilon$
    - $A \rightarrow \alpha$
    - $A \rightarrow \alpha B$

    όπου:
    - $A, B \in N$
    - $\alpha \in \Sigma$

Σχηματικά, όλα τα παραπάνω παρουσιάζονται στην Εικόνα 3.2.

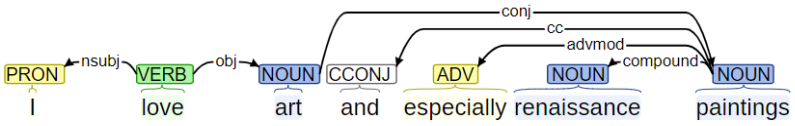


Εικόνα 3.2 Γραμματικές ιεραρχίας Chomsky.

3.2.5. Ανάλυση εξάρτησης (Dependency Parsing)

Έναν άλλο γραμματικό φορμαλισμό αποτελούν οι γραμματικές εξάρτησης. Σε αυτούς τους φορμαλισμούς όπως αναλύεται και από τους (Jurafsky & Martin,2023,σελ. 381-385) τα φραστικά συστατικά και οι κανόνες δομής φράσεων δεν παίζουν άμεσο ρόλο. Η συντακτική δομή της πρότασης περιγράφεται αποκλειστικά με όρους κατευθυνόμενων δυαδικών γραμματικών σχέσεων μεταξύ των λέξεων. Έστω ότι έχουμε την πρόταση “I love art and especially renaissance paintings”.

Η ανάλυση εξάρτησης θα είναι όπως φαίνεται στην Εικόνα 3.3., η οποία προέκυψε με την χρήση του online εργαλείου http://stanza.run/.



Εικόνα 3.3. - Dependency parsing

Οι σχέσεις μεταξύ των λέξεων απεικονίζονται πάνω από την πρόταση με κατευθυνόμενα, επισημασμένα τόξα από τις κεφαλές στα εξαρτώμενα στοιχεία. Αυτές οι σχέσεις, που εξαρτώνται από την κεφαλή(head), κωδικοποιούν σημαντικές πληροφορίες που συνήθως κρύβονται στις πιο περίπλοκες αναλύσεις δομής φράσεων. Μπορούν να αναπαρασταθούν ως μια δομή εξάρτησης, δηλαδή ένα κατευθυνόμενο γράφημα G = (V,A), το οποίο αποτελείται από ένα σύνολο κορυφών V και ένα σύνολο διατεταγμένων ζευγών κορυφών A,τα τόξα. Υποθέτουμε ότι το σύνολο των κορυφών V, αντιστοιχεί ακριβώς στο σύνολο των λέξεων που υπάρχουν σε μια δεδομένη πρόταση. Το σύνολο των τόξων A, καταγράφει τις σχέσεις εξάρτησης από τη κεφαλή και της γραμματικής συνάρτησης μεταξύ των στοιχείων στο V. Οι δομές πρέπει να συνδέονται, να έχουν καθορισμένο ριζικό (root) κόμβο, από τον οποίο ξεκινάει το δέντρο και τέλος, να είναι άκυκλες ή επίπεδες.
Αντιπροσωπεύουν τη σημασιολογική σχέση μεταξύ των κατηγορημάτων (predicate) και των παραμέτρων τους (arguments) τους και για αυτό το λόγο οι γραμματικές εξάρτησης χρησιμοποιούνται ευρέως στην επεξεργασία της φυσικής γλώσσας.
Η κεφαλή παίζει το ρόλο της κεντρικής οργανωτικής λέξης και το στοιχείο που εξαρτάται από αυτή είναι ένα είδος τροποποιητή. Η σχέση εξαρτώμενης κεφαλής γίνεται σαφής συνδέοντας απευθείας τις κεφαλές με τις λέξεις που εξαρτώνται άμεσα από αυτές. Εκτός από τον καθορισμό των εξαρτημένων από την κεφαλή ζευγών, οι γραμματικές εξάρτησης μας επιτρέπουν να ταξινομήσουμε τα είδη γραμματικών σχέσεων ή γραμματικών συναρτήσεων που έχει το εξαρτώμενο στοιχείο σε σχέση με την κεφαλή του.(Jurafsky & Martin,2023,σελ. 382) Αυτές περιλαμβάνουν γνωστές έννοιες όπως υποκείμενο, άμεσο αντικείμενο και έμμεσο αντικείμενο.

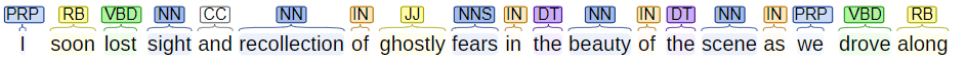
Πρέπει να υπάρχει ένας μόνο καθορισμένος ριζικός κόμβος που δεν έχει εισερχόμενα τόξα και κάθε κορυφή να έχει ακριβώς ένα εισερχόμενο τόξο.Τέλος, πρέπει να υπάρχει

μια και μοναδική διαδρομή από τον ριζικό κόμβο σε κάθε κορυφή στο V, εξασφαλίζοντας έτσι πως υπάρχει ένας μόνο ριζικός κόμβος από τον οποίο υπάρχει μια μοναδική κατευθυνόμενη διαδρομή προς καθεμία από τις λέξεις της πρότασης.
Μερικές από τις σχέσεις ανάλυσης στα αγγλικά όπως αναφέρονται από τους De Marneffe et al.(2021) στο έργο τους Universal Dependencies (UD)είναι οι εξής:
acl: adnominal clause; finite or non-finite clause modifying a nominal
advcl :adverbial clause modifying a predicate or modifier word
advmod :adverb or adverbial phrase modifying a predicate or modifier word
amod: adjectival modifier of a nominal
appos: appositional modifier; a nominal used to define, name,
or describe the referent of a preceding nominal
aux: auxiliary; links a function word expressing tense, mood, aspect,
voice, or evidentiality to a predicate
case: links a case-marking element (preposition, postposition, or clitic)
to a nominal
cc: links a coordinating conjunction to the following conjunct
ccomp :clausal complement of a verb or adjective without an
obligatorily controlled subject
compound: any kind of word-level compounding (noun compound,
serial verb, phrasal verb)
conj : conjunct; links two elements which are conjoined
cop: copula; links a function word used to c
onnect a subject and a nonverbal predicate to
the nonverbal predicate
csubj: clausal syntactic subject of a predicate
det: determiner (article, demonstrative, etc.) in a nominal
nmod :nominal modifier; a nominal modifying another nominal
nsubj :nominal subject; nominal core argument which is
the syntactic subject (or pivot) of apredicate
obj :object; the core argument nominal which is the most basic core argument
that is notthe subject, typically the most directly affected participant
punct: punctuation attached to the head of its clause or phrase
root: root of the sentence
xcomp :clausal complement of a verb or adjective with an obligatorily
controlled subject.

3.2.6. Επισήμανση μερών του λόγου (Part-of-speech tagging)

Η διαδικασία της επισήμανσης των μερών του λόγου (Part-of-speech tagging) αναφέρεται στον προσδιορισμό του μέρους του λόγου που αντιστοιχεί σε κάθε λέξη ενός κειμένου. Σε αυτήν τη διαδικασία, μια ακολουθία (τοκενοποιημένων) λέξεων συνοδεύεται από ένα σύνολο ετικετών. Η είσοδος αποτελείται από μια ακολουθία x1, x2,..., xn τοκενοποιημένων λέξεων και ένα σύνολο ετικετών, ενώ η έξοδος αποτελείται από μια ακολουθία y1, y2,..., yn ετικετών. Κάθε έξοδος yi αντιστοιχεί ακριβώς σε μια είσοδο xi στο κείμενο. Η προσθήκη ετικετών αποτελεί μια διαδικασία αποσαφήνισης. Οι λέξεις είναι αμφίσημες, καθώς μπορεί να ανήκουν σε περισσότερα από ένα μέρη του λόγου, και ο στόχος είναι να εντοπιστεί η σωστή ετικέτα για κάθε περίπτωση. Για παράδειγμα, μια λέξη μπορεί να είναι είτε ρήμα είτε ουσιαστικό σε μια πρόταση της αγγλικής γλώσσας. Επιπλέον, μπορεί να είναι προσδιοριστική ή συμπληρωματική. Η ανάθεση ετικετών POS έχει ως στόχο να αντιμετωπίσει αυτήν την ασάφεια, επιλέγοντας την κατάλληλη ετικέτα για κάθε πλαίσιο.

Έστω ότι έχουμε την πρόταση “I soon lost sight and recollection of ghostly fears in the beauty of the scene as we drove along”. Το αποτέλεσμα του Part-of-speech tagging θα είναι αυτό της Εικόνας 3.4., το οποίο προέκυψε με την χρήση του online εργαλείου http://stanza.run/.



Εικόνα 3.4. Part-of-speech tagging.

4. Αναπαράσταση Γνώσης και Γράφοι Γνώσης

Η Γνώση σαν έννοια είναι πολύ δύσκολο να προσδιοριστεί και να περιγραφεί με έναν μόνο ορισμό. Από τον αρχαίο φιλόσοφο Πλάτων περιγράφεται ως «δικαιολογημένη αληθινή πίστη», δηλαδή ένα άτομο γνωρίζει ότι μια πρόταση είναι αληθινή εάν (και μόνο εάν) πιστεύει στην αλήθεια της πρότασης και ταυτόχρονα όμως μπορεί να δικαιολογήσει την πίστη του αυτή (Jakus et. al, 2013, σελ. 47). Στην επιστήμη των υπολογιστών η Γνώση θα πρέπει να περιέχει πληροφορίες πιστοποιημένες ως προς την ποιότητά τους, να μπορεί να εφαρμοστεί, να είναι λειτουργική αλλά και σε πολλές περιπτώσεις να έχει προγνωστική αξία (Jakus et. al, 2013, σελ. 48).

Προκειμένου να επιλύσουμε ένα πρόβλημα πρέπει να μπορούμε να το ορίσουμε και να το περιγράψουμε, ανεξάρτητα τον επιστημονικό τομέα τον οποίο μπορεί να ανήκει. Αυτό συμβαίνει με την Αναπαράσταση Γνώσης**.** Η αναπαράσταση γνώσης επιτυγχάνεται με τη χρήση Γλωσσών και ενός συνόλου συντακτικών και σημασιολογικών παραδοχών. Πιο συγκεκριμένα, οι γλώσσες που χρησιμοποιούνται για την Αναπαράσταση Γνώσης έχουν(Παναγιωτακόπουλος et. al,2022, σελ.282):

1. ένα λεξιλόγιο που περιλαμβάνει όρους, σύμβολα, σταθερές και μεταβλητές,
2. ένα συντακτικό που επιτρέπει τη δημιουργία εκφράσεων με βάση το λεξιλόγιο αυτό,
3. μια σημασιολογία για την ερμηνεία των εκφράσεων και την παραγωγή μιας τιμής αληθείας,
4. ένα σύνολο από κανόνες συμπερασμάτων (inference rules) οι οποίοι καθορίζουν το πώς μία συνθήκη μπορεί να εξαχθεί από άλλες συνθήκες που είναι σε ισχύ.

Είναι απαραίτητος λοιπόν ένας φορμαλισμός ώστε η αναπαράσταση να είναι ακριβής και κατανοητή, να αποφεύγονται η ασαφεια, η ασυνέπεια, και η δυσκολία στη μοντελοποίηση και να είναι δυνατή η επεξεργασία και η εξαγωγή συμπερασμάτων από τα συστήματα των υπολογιστών.

4.1. Μέθοδοι Αναπαράστασης.

Στο πεδίο της Τεχνητής Νοημοσύνης, μερικές από τις σημαντικότερες μεθόδους αναπαράστασης είναι τα Σχήματα Λογικής Αναπαράστασης, οι Δομημένες Μορφές Αναπαράστασης Γνώσης και οι Κανόνες.

4.1.1. Σχήματα Λογικής Αναπαράστασης

Χρησιμοποιούν εκφράσεις της τυπικής λογικής, η οποία βασίζεται σε λογικά προτασιακά σχήματα. Η Προτασιακή λογική περιλαμβάνει δηλωτικές προτάσεις (declarative sentences) οι οποίες μπορούν να είναι αληθείς (true) ή ψευδείς (false), αλλά όχι και τα δύο, υπάρχει δηλαδή μια μόνο τιμή αληθείας για κάθε πρόταση. Με τους 5 λογικούς συνδέσμους:

1. Άρνηση: όχι (not) ,
2. Διάζευξη: ή (or) ,
3. Σύζευξη και (and) ,
4. Συνεπαγωγή αν ... τότε (if ... then) ,
5. Ισοδυναμία αν και μόνο αν (if and only if),

συντίθενται οι προτάσεις, απλές και περίπλοκες.

Η Κατηγορηματική λογική αποτελεί στην ουσία προέκταση της Προτασιακής λογικής. Υπάρχει η δυνατότητα καθορισμού σχέσεων και γενικεύσεων σχετικά με τις προτάσεις με την χρήση σταθερών που παριστάνουν αντικείμενα του κόσμου, συναρτησιακών συμβόλων που χρησιμοποιούνται για την κατασκευή δομών που αντιπροσωπεύουν πάλι αντικείμενα που προσδιορίζονται μέσω άλλων αντικειμένων του ίδιου κόσμου και τέλος με τη χρήση κατηγορημάτων που περιγράφουν τις ιδιότητες και τις σχέσεις των αντικειμένων που αντιπροσωπεύονται από σύμβολα.Με τη χρήση συναρτήσεων παράγονται αξίες και όλα τα παραπάνω σε συνδυασμό εξάγουν τα συμπεράσματα
Όλα τα παραπάνω, σε συνδυασμό με τη χρήση συναρτήσεων που παράγουν μια αξία συνθέτουν την Κατηγορηματική λογική Πρώτης Τάξης(First-Order Predicate Logic), την πιο συνηθισμένη γλώσσα λογικής αναπαράστασης.

4.1.2. Δομημένες Μορφές Αναπαράστασης Γνώσης

Σε αντίθεση με τις λογικές μορφές αναπαράστασης, οι Δομημένες Μορφές Αναπαράστασης Γνώσης επιτρέπουν μεγαλύτερη ελευθερία στην αναπαράσταση. Αποτελούνται από τα σημασιολογικά δίκτυα, τα σχήματα, τα πλαίσια και τα σενάρια.

Όλες οι δομημένες μορφές αναπαράστασης περιλαμβάνουν(Γεωργούλη,2015,σελ. 80):

1. Έννοιες,
2. Διάφορα είδη των δεσμών μεταξύ των εννοιών όπως "έχει-μέρος" (has-part ή aggregation), "ανήκει" (is-a ή specialization) αλλά και περισσότερο εξειδικευμένες σχέσεις ανάλογα με τον τομέα,
3. Μπορεί να έχουν Κληρονόμηση (inheritance) ή/και κάποιο είδος διαδικαστικού συνημμένου.

Στα σημασιολογικά δίκτυα η γνώση συγκεντρώνεται σε συνεκτικές σημασιολογικές ομάδες. Η γνώση χαρτογραφείται σε ένα δίκτυο και υπάρχει γραφική αναπαράσταση των αντικειμένων γνώσης, των ιδιοτήτων τους και της σχέσης μεταξύ τους (Chowdhary,2020,σελ.180). Στα δίκτυα αυτά δεν υπάρχει απαραίτητα ιεραρχία αλλά υπάρχει κληρονομικότητα ιδιοτήτων. Οι πληροφορίες είναι ομαδοποιημένες και συνδέονται μεταξύ τους με σχεσιακούς δεσμούς. Η Γνώση που περιέχουν για την εκτέλεση συγκεκριμένων εργασιών προέρχεται από ένα συγκεκριμένο τομέα (domain).Προσεγγίζονται οι ιδιότητες και οι σχέσεις των αντικειμένων που περιέχει, τα γεγονότα, τα στοιχεία, οι καταστάσεις και οι ενέργειες από τις οποίες αποτελείται (Γεωργούλη,2015,σελ. 80). Κάθε σημασιολογικό δίκτυο αναπτύσσεται ανάλογα τη γνώση που αναπαριστά και τις ανάγκες που πρέπει να καλύψει. Τα περισσότερα δίκτυα όμως αποτελούνται από κόμβους που δηλώνουν αντικείμενα, αλλά μπορεί ακόμα να αναπαριστούν τις ιδιότητες και τις τιμές των οντοτήτων, έννοιες, περιστατικά, γεγονότα που λαμβάνουν χώρα κ.ο.κ., ακμές ή συνδέσμους που δηλώνουν τις σχέσεις ανάμεσα στα αντικείμενα αυτά και ετικέτες που δηλώνουν συγκεκριμένα αντικείμενα και τις σχέσεις μεταξύ τους. Με τους κόμβους και τις ακμές σχηματίζονται στη συνέχεια τα γραφήματα, με τις ετικέτες να τοποθετούνται στις άκρες αντιπροσωπεύοντας τη σχέση που υπάρχει μεταξύ των κόμβων και αναπαριστώντας έννοιες (δηλαδή τη σημασιολογία του δικτύου) (Chowdhary,2020, σελ. 182). Οι πιο συνηθισμένες ιεραρχίες που υπάρχουν μεταξύ των κόμβων είναι οι παρακάτω(Γεωργούλη,2015, σελ. 81):

1. a_kind_of (δεσμός AKO): Η σχέση αυτή υπάρχει μεταξύ κόμβων οντοτήτων που αντιπροσωπεύουν κλάσεις αντικειμένων. Σε αυτούς τους κόμβους με αυτή τη σχέση μπορούν να προστεθούν νέοι δεσμοί που προσδίδουν νέες ιδιότητες.
2. is_a (δεσμός ISA): Η σχέση αυτή είναι παρόμοια με τη σχέση AKO, αλλά αντιπροσωπεύει υποκλάση οντοτήτων αντί για κλάση και έτσι στον κόμβο δεν επιτρέπεται να του προστεθούν νέες ιδιότητες,γίνεται όμως να κληρονομηθούν οι ήδη υπάρχουσες ιδιότητες από κόμβους κλάσεων ψηλότερα στην ιεραρχία ή να αλλάξουν οι τιμές στις οι ιδιότητες που κληρονομούνται.
3. instance_of (δεσμός INSTANCE_OF): Η σχέση αυτή υπάρχει μόνο μεταξύ κόμβων οντοτήτων που αποτελούν στιγμιότυπα του κόσμου που απεικονίζεται στη δομή και κόμβων γενικότερων κλάσεων.

Τα πλαίσια αποτελούνται από τα πλαίσια κλάσεων που περιγράφουν μία ανώτερη κλάση πραγμάτων και τα πλαίσια στιγμιοτύπων που περιγράφουν ένα συγκεκριμένο στοιχείο.Αποτελούνται ακόμα από σχισμές που περιέχουν έννοιες ή παραπέμπουν σε άλλα πλαίσια.Με τα πλαίσια ένα σημασιολογικό δίκτυο εκφράζεται σε υψηλότερο επίπεδο από αυτό τηςγραφικής αναπαράστασης, οι κόμβοι και οι συνδέσεις της οποίας μπορούν να αναπαριστούν και με αυτόν τον τρόπο. Το πλαίσιο παίρνει το όνομά του από αυτό του αντίστοιχου κόμβου και οι θέσεις του πλαισίου από τα ονόματα των συνδέσεων.Οι έννοιες ή αξίες που περιέχονται στις θέσεις αντιστοιχούν στις έννοιες ή αξίες που απεικονίζουν οι κόμβοι στους οποίους καταλήγουν οι συνδέσεις (Γεωργούλη, 2015, σελ. 90). Κατ' επέκταση, με τα σενάρια δομείται η γνώση στο πλαίσιο της κατανόησης της γλώσσας προκειμένου να επεξεργαστεί η γλώσσα και οι υψηλού επιπέδου σκέψη. Η γνώση αναπαριστάται επεισοδιακά με παράθεση πλαισίων σε χρονική σειρά και περιλαμβάνει στοιχεία όπως οι συνθήκες εισόδου του σεναρίου, τα αποτελέσματα, ιδιότητες (αντικείμενα) και ρόλους (οντότητες), κανάλια και σκηνές , δηλαδή ακολουθίες των γεγονότων που συμβαίνουν.

4.2. Οντολογίες

Οι οντολογίες ανήκουν στην κατηγορία της δομημένης αναπαράστασης γνώσης και χρησιμοποιούνται κυρίως στην Τεχνητή Νοημοσύνη και στον Σημασιολογικό Ιστό για η διαχείριση γνώσης, τη επεξεργασία φυσικής γλώσσας, τον σχεδιασμό βάσεων δεδομένων αλλά και πολλών άλλων λειτουργιών.

Η οντολογία είναι ένας σαφής και επίσημος (formal) προσδιορισμός κοινών εννοιών και των σχέσεων μεταξύ των εννοιών αυτών (conceptualization) ενός τομέα-πεδίου(domain) ενδιαφέροντος(Buitelaar et al.,2005). Μπορεί να οριστεί ως οι έννοιες, οι σχέσεις,τα χαρακτηριστικά και οι ιεραρχίες που υπάρχουν στον τομέα (Asim et. al.,2018). Μία οντολογία πρέπει να είναι αναγνώσιμη από τη μηχανή και αποδεκτή από την κοινότητα του τομέα τον οποίο αφορά (Buitelaar et al.,2005).
Τα συστατικά μιας οντολογίας ορίζονται ως (Al-Aswadi et al., 2020):

$$O =< C, H, R, A >$$

όπου:
O: οντολογία
C: σύνολο κλάσεων εννοιών, οι οποίες σχετίζονται με
ένα πεδίο ή κάποιες εργασίες
H: ταξινομικές σχέσεις, σύνολο δηλαδή ιεραχικών δεσμών μεταξύ των εννοιών
R: μη-ταξινομικές σχέσεις, σύνολο δηλαδή εννοιολογικών δεσμών
A: σύνολο κανόνων και αξιωμάτων, δηλαδή προτάσεων π
ου είναι πάντα αληθείς.

Σύμφωνα με τους (Asim et al.,2018;Buitelaar et al.,2005;Γεωργούλη,2015) όσον αφορά την επεξεργασία φυσικής γλώσσας, η ανάπτυξη μιας οντολογίας στοχεύει στην απόκτηση γλωσσικών γνώσεων σχετικά με τους όρους που χρησιμοποιούνται για την αναφορά σε μια συγκεκριμένη έννοια του κειμένου και πιθανά συνώνυμα των όρων αυτών. Περαιτέρω, με τη χρήση κανόνων μπορούν να αποκτηθούν επιπλέον συστατικά του κειμένου τα οποία δεν έχουν κωδικοποιηθεί ρητά από την οντολογία αλλά μπορούν να εξαχθούν από αυτή.
Με την χρήση κατηγορηματικής λογικής πρώτης τάξεως σε μία οντολογία μπορούν να οριστούν κατηγορίες, υποκατηγορίες μιας κατηγορίας, αντικείμενα ως μέλη κατηγοριών, ιδιότητες κοινές στα μέλη μιας κατηγορίας και αναγνώριση μελών μιας κατηγορίας βάσει των κοινών χαρακτηριστικών τους.
Με την χρήση ταξινομίας (taxonomy) αντιμετωπίζεται ο μεγάλος όγκος αντικειμένων της οντολογίας με τον ορισμό υποκατηγοριών, ιεραρχικής δομής κατηγοριών και υποκατηγοριών, κληρονομικότητα - τα στιγμιότυπα μιας υποκατηγορίας κληρονομούν τις ιδιότητες του γονέα. Δομείται έτσι ένα σημασιολογικό δίκτυο με σχέσεις AKO (a_kind_of) και ISA (is_a).

4.2.1. Δημιουργία Οντολογίας

Όσον αφορά τη δημιουργία μιας οντολογίας έχει θεσπιστεί η σχηματοποίησή της σε στρώματα (Ontology learning layer cake). Υπάρχουν τα έξι εξής στρώματα: όροι, συνώνυμα, έννοιες, ιεραρχίες εννοιών, σχέσεις και κανόνες(Buitelaar et al.,2005).
Η κατασκευή μιας οντολογίας μπορεί να οριστεί ως μία επαναληπτική μέθοδος κατά την οποία η οντολογία ή δημιουργείται από την αρχή ή στηρίζεται πάνω σε μία ήδη υπάρχουσα και αυτή εμπλουτίζεται (enriching /populating)(Al-Aswadi et al., 2020).

Σύμφωνα με τους (Buitelaar et al,2005; Al-Aswadi et al., 2020), η διαδικασία κατασκευής μιας οντολογίας περιλαμβάνει τα εξής βήματα:

1. Καθορισμός του domain για τη δημιουργία καλά καθορισμένων όρων και εννοιών. Ένας όρος μπορεί να είναι token πολλαπλών λέξεων ή μονής λέξης, το οποίο υποδηλώνει μια συγκεκριμένη έννοια σε έναν δεδομένο τομέα.Οι όροι είναι γλωσσικές υλοποιήσεις συγκεκριμένων εννοιών και απαραίτητη για την εκτέλεση πιο συνθετών διαδικασιών. Κατά την εξαγωγή όρων πραγματοποιούνται διάφορα επίπεδα γλωσσικής επεξεργασίας όπως ανάλυση φράσεων για τον εντοπισμό σύνθετων φράσεων ουσιαστικών που εκφράζουν όρους, ανάλυση εξαρτήσεων στη δομή (dependency structure analysis) ώστε να προσδιοριστεί η εσωτερική σημασιολογική τους δομής. Συνήθως χρησιμοποιείται ένας part-of-speech tagger πάνω στο σώμα δεδομένων του τομέα και έπειτα εντοπίζονται οι πιθανοί όροι με μη αυτόματο τρόπο.
2. Προσδιορισμός των βασικών όρων, εννοιών και των σχέσεών τους στο domain. Ο σχηματισμός εννοιών αποτελεί ασαφή διεργασία καθώς δεν υπάρχει ένας μοναδικός ορισμός του τι είναι μια έννοια. Ο γενικός ορισμός μιας έννοιας είναι ότι πρέπει να περιλαμβάνει την πρόθεση της έννοιας, την επέκταση των εννοιών και τα λεξικά σημεία (όροι) που χρησιμοποιούνται για να αναφέρονται σε αυτές τις έννοιες. Η ανακάλυψη σχέσεων στοχεύει στην εξαγωγή μιας νέας σχέσης μεταξύ γνωστών εννοιών.
3. Ιεράρχηση εννοιών η οποία περιλαμβάνει την επαγωγή, την επέκταση και τον επαναπροσδιορισμό των εννοιών.
4. Ανακάλυψη συνωνύμων με στόχο την εύρεση όρων που δηλώνουν την ίδια έννοια και εμφανίζονται στο ίδιο σύνολο για μια επιλεγμένη έννοια.Ένα σύνολο συνωνύμων είναι ο πιο συνηθισμένος τύπος λεξιλογικών σχέσεων. Υπάρχουν

έτοιμα διαθέσιμα σύνολα, όπως το σύνολο WordNet, τεχνικές ομαδοποίησης και άλλες παρόμοιες μέθοδοι.

5. Καθιέρωση ή εξαγωγή των κανόνων και των αξιωμάτων που περιγράφουν τις δομικές ιδιότητες του domain. Η εξαγωγή κανόνων και αξιωμάτων είναι η τελευταία υπο-εργασία στη διαδικασία κατασκευής και στοχεύει στο να συναγάγει τους κανόνες οι οποίοι βασίζονται στις έννοιες και σχέσεις που έχουν εξαχθεί.
6. Κωδικοποίηση (αναπαράσταση) των κατασκευασμένων οντολογιών χρησιμοποιώντας γλώσσες αναπαράστασης που υποστηρίζουν την οντολογία όπως RDF, RDFS ή OWL.
7. Συνδυασμός των κατασκευασμένων οντολογιών με υπάρχουσες οντολογίες (εάν υπάρχουν).
8. Αξιολόγηση των κατασκευασμένων οντολογιών χρησιμοποιώντας γενικές και ειδικές μετρικές αξιολόγησης.

Για την κατασκευή της οντολογίας ακολουθούμε τις ακόλουθες τεχνικές όπως αναφέρουν και οι (Aidan et al., 2020;Al-Aswadi et al., 2020;Asim et al.,2018) :

1. Προεπεξεργασία του κειμένου με τη χρήση γλωσσικών τεχνικών όπως ανάλυση κειμένου (parsing), part of speech tagging και λημματοποίηση.
2. Εξαγωγή σχετικών όρων και εννοιών του domain με τη χρήση διαφόρων τεχνικών επεξεργασίας φυσικής γλώσσας όπως συντακτική ανάλυση, πλαίσια υποκατηγοριοποίησης και εξαγωγή λέξεων ρίζας μαζί με ορισμένες τεχνικές από το στατιστικό τομέα όπως η ανάλυση αντίθεσης, η ανάλυση συνεμφάνισης, η λανθάνουσα σημασιολογική ανάλυση (LSA) και η ομαδοποίηση.
3. Ομαδοποίηση εννοιών και εύρεση ταξινομικών και μη ταξινομικών σχέσεων μεταξύ των εννοιών αυτών με τη χρήση τεχνικών και στατιστικών προσεγγίσεων Επεξεργασίας Φυσικής Γλώσσας που περιλαμβάνουν λεξικοσυντακτική ανάλυση (lexico-syntactic analysis) , ανάλυση εξάρτησης(dependency analysis),υπαγωγή όρων (term subsumption), ανάλυση τυπικής έννοιας (formal concept analysis - FCA), ιεραρχική ομαδοποίηση (hierarchical clustering) και εξόρυξη κανόνων συσχέτισης (association rule mining - ARM).
4. Σχηματισμός αξιωμάτων με τη χρήση ILP - inductive logic programming.
5. Αξιολόγηση ακεραιότητας της ανεπτυγμένης οντολογίας με τη χρήση διαφόρων μέτρων αξιολόγησης.

Η σειρά των παραπάνω τεχνικών είναι ενδεικτική και δεν αντιπροσωπεύει την σειρά εκτέλεσης των βημάτων. Η κατασκευή μπορεί να γίνει είτε χειροκίνητα, είτε συνεταιριστικά, είτε ημι-αυτόματα. Στην χειροκίνητη, η κατασκευή εκτελείται από ειδικούς. Στην συνεταιριστική , σχεδόν όλες οι ενέργειες που εκτελούνται για την κατασκευή επιβλέπονται από ειδικούς. Τέλος, στην ημι-αυτόματη, η κατασκευή εκτελείται αυτόματα με περιορισμένη την παρέμβαση από χρήστες ή ειδικούς. Πλήρως αυτόματη κατασκευή δεν υφίσταται. Το επίπεδο απλά της ανθρώπινης παρέμβασης κατά τη δημιουργία είναι αρκετά μικρότερο από αυτό στην ημιαυτόματη κατασκευή.

4.2.2. Γλώσσες Οντολογίας

Οι διάφορες γλώσσες οντολογίας μπορούν να ταξινομηθούν σε κατηγορίες ανάλογα το σύστημα λογικής στο οποίο βασίζονται (Jakus et. al, 2013, σελ. 33):

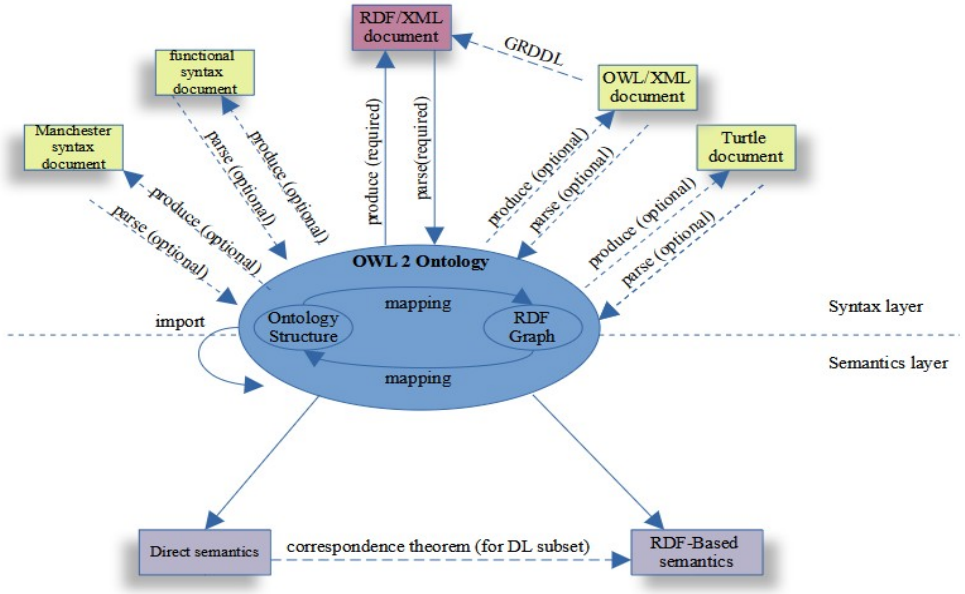
1. Λογική Περιγραφή: Οι γλώσσες αυτές προέρχονται από το πεδίο της Τεχνητής Νοημοσύνης. Δύο από τις πιο γνωστές είναι η KL-ONE (Brachman και Schmolze 1985) και η LOOM (MacGregor και Bates 1987).
2. Λογική πρώτης Τάξης: Στην κατηγορία αυτή ανήκουν οι γλώσσες CYCL (Lenat και Guha 1989), KIF (Genesereth και Fikes 1992), Ontolingua (Gruber 1993) και Common Logic (CLWG 2012).
3. Σημασιολογικός Ιστός: Οι γλώσσες αυτές αναπτύχθηκαν για την ανταλλαγή γνώσης στο διαδίκτυο και βασίζονται επίσης στη λογική, επομένως ανήκουν ουσιαστικά και στις προηγούμενες κατηγορίες. Τέτοιες είναι οι XML-based Ontology exchange Language (XOL) (Karp et al. 1999), Simple HTML Ontology Extension (SHOE) (Heflin et al. 1999), Ontology Interchange Language (OIL) (Fensel et al. 2001) και η πιο γνωστή Web Ontology Language (OWL) (OWL 2009), στην οποία θα αναφερθούμε εκτενέστερα παρακάτω, σε συνδυασμό και με τους Γράφους Γνώσης.
4. Αναπαράσταση με εννοιολογικά γραφήματα (conceptual graphs) και σημασιολογικά δίκτυα.

4.2.2.1. Η γλώσσα OWL (Ontology Web Language)

Προκειμένου να αναπαρασταθούν οι οντολογίες χρησιμοποιείται συνήθως η γλώσσα OWL (Ontology Web Language) η οποία βασίζεται στην Τυπική λογική και δημιουργήθηκε από την Κοινοπραξία Παγκόσμιου Ιστού (World Wide Web Consortium

– W3C) (Παναγιωτακόπουλος et. al,2022, σελ.291). Υπάρχουν και άλλες γλώσσες όπως είδαμε παραπάνω, αλλά η OWL είναι η πιο διαδεδομένη και μπορεί επίσης να μεταφραστεί στην γλώσσα RDF, περισσότερα για την οποία αναφέρονται στο Κεφάλαιο 4.5.

Όπως παρουσιάζεται και στην επίσημη τεκμηρίωση της W3C (OWL 2 Web Ontology Language Document Overview (Second Edition) (w3.org)) η δομή της γλώσσας OWL είναι η ακόλουθη σχηματικά:



Εικόνα 4.1. Η δομή της Ontology Web Language

Στην Εικόνα 4.1. φαίνονται τα κύρια δομικά στοιχεία της και πως αυτά σχετίζονται μεταξύ τους. Στο κέντρο απεικονίζεται η αφηρημένη έννοιας μιας οντολογίας η οποία μπορεί να είναι είτε αφηρημένη δομή είτε ένα γράφημα RDF. Στην κορυφή υπάρχουν διάφορες συγκεκριμένες συντακτικές που χρησιμοποιούνται για σειριοποίηση και ανταλλαγή οντολογιών, ενώ στο κάτω μέρος βρίσκονται οι δύο σημασιολογικές προδιαγραφές που ορίζουν την έννοια των οντολογιών OWL. Συνήθως οι ανάγκες ενός χρήστη καλύπτονται με τη χρήση μίας μόνο σύνταξης και μίας σημασιολογίας.

Οι κύριες έννοιες στην γλώσσα OWL είναι η οντότητα ή το στιγμιότυπο, ο τύπος ή η κλάση και οι συσχετίσεις μεταξύ των κλάσεων που μεταφέρονται σε συσχετίσεις μεταξύ οντοτήτων. Οι έννοιες οντότητα και κλάση συνδέονται με τη σχέση του στιγμιότυπου.

Τα πεδία ενός τύπου αποτελούν ανεξάρτητους τύπους οι οποίοι έχουν τη σχέση has (έχει) με τον βασικό τύπο.

Η βάση της οντολογίας είναι τριάδες της μορφής Υποκείμενο- Κατηγόρημα – Αντικείμενο. Το υποκείμενο αντιπροσωπεύει την οντότητα στην οποία αναφερόμαστε, το κατηγόρημα εκφράζει τι θέλουμε να πούμε για την οντότητα αυτή και αντικείμενο την τιμή ή την πληροφορία που αφορά την οντότητα (Παναγιωτακόπουλος et. al,2022, σελ.292). Ουσιαστικά με τις τριάδες περιγράφουμε τα χαρακτηριστικά και τις λειτουργίες του υποκειμένου και μπορούν να αναπαρασταθούν και ως γραφήματα γνώσης (τι είναι γράφημα/γράφος γνώσης αναλύουμε στο Κεφάλαιο 4.4).

Για την παραγωγή συμπερασμάτων από μία οντολογία χρησιμοποιείται η Κατηγορηματική Λογική και οι κανόνες επαγωγής. Οι κανόνες αυτοί χρησιμοποιούν τις υπάρχουσες τριάδες και, με τυπική λογική, είτε κάνουν έλεγχο ορθότητας είτε παράγουν νέες τριάδες. (Παναγιωτακόπουλος et. al,2022). Έτσι ελέγχεται η συνάφεια της οντολογίας και παράγονται συμπεράσματα - νέα γνώση.

4.2.3. Εργαλεία δημιουργίας οντολογιών

Μία οντολογία μπορεί να δημιουργηθεί είτε χειροκίνητα χρησιμοποιώντας εργαλεία λογισμικού, είτε αυτόματα, από ήδη υπάρχοντες πόρους πληροφοριών. Τα πιο συνηθισμένα εργαλεία λογισμικού για την δημιουργία οντολογιών είναι το TERMINAE, το Protégé, το KAON, το HOZO και το OntoStudio και η μεθοδολογία για την κατασκευή των οντολογιών, που εφαρμόζεται επίσης και σε ορισμένα από αυτά τα εργαλεία, αναπτύχθηκε στο πλαίσιο των έργων OntoClean , On-To-Knowledge και Dogma (Jakus et. al, 2013, σελ. 33). Έχουν αναπτυχθεί ακόμα προσεγγίσεις για την αυτόματη ή ημιαυτόματη δημιουργία οντολογίας, η οποία ονομάζεται ως εκμάθηση οντολογίας. Η εκμάθηση γίνεται από ήδη υπάρχοντες πόρους, όπως έγγραφα κειμένου, διαδικτυακά έγγραφα και πολυμέσα. Στην παρούσα εργασία μας ενδιαφέρει η εκμάθηση οντολογίας συγκεκριμένα από κειμενικά δεδομένα. Η εκμάθηση οντολογίας (ontology learning) από κειμενικά δεδομένα είναι μια διαδικασία απόκτησης γνώσης από κείμενο, εφαρμόζοντας ένα σύνολο μεθόδων και τεχνικών από διάφορα πεδία όπως η επεξεργασία φυσικής γλώσσας (NLP), η εξόρυξη δεδομένων και η μηχανική μάθηση, με σκοπό την εξαγωγή οντολογικών στοιχείων και στη συνέχεια την κατασκευή οντολογιών (Browarnik & Maimon, 2015).

4.3. Λεξιλογική βάση δεδομένων WordNet

Το WordNet είναι μια μεγάλη, διαθέσιμη για το κοινό, λεξιλογική βάση δεδομένων της αγγλικής γλώσσας. Τα ουσιαστικά, τα ρήματα, τα επίθετα και τα επιρρήματα ομαδοποιούνται σε σύνολα γνωστικών συνωνύμων (συνόλων - synsets), το καθένα εκφράζοντας μια ξεχωριστή έννοια (Fellbaum,2005). Τα σύνολα συνδέονται μεταξύ τους μέσω εννοιολογικών-σημασιολογικών και λεξιλογικών σχέσεων. Η δομή του το καθιστά χρήσιμο εργαλείο για την υπολογιστική γλωσσολογία και την επεξεργασία φυσικής γλώσσας. Συνδέει όχι μόνο μορφές λέξεων - συμβολοσειρές γραμμάτων - αλλά συγκεκριμένες έννοιες λέξεων (Fellbaum,2005). Λέξεις με κοντινή απόσταση στο δίκτυο είναι αποσαφηνισμένες σημασιολογικά και οι μεταξύ τους σημασιολογικές σχέσεις είναι επισημασμένες. Η κύρια σχέση που υπάρχει μεταξύ των λέξεων στο WordNet είναι η συνωνυμία. Τα συνώνυμα, δηλαδή λέξεις που δηλώνουν την ίδια έννοια,ομαδοποιούνται σε μη διατεταγμένα σύνολα (synets). Τα 117.000 synets που περιλαμβάνονται στο WordNet συνδέονται και με άλλα synets βασισμένα σε ένα μικρό αριθμό «εννοιολογικών σχέσεων». Ένα synet περιέχει έναν σύντομο ορισμό και σε αρκετές περιπτώσεις μία ή περισσότερες σύντομες προτάσεις που απεικονίζουν τη χρήση των μελών synset (Fellbaum,2005). Μορφές λέξεων που αντιπροσωπεύουν διάφορες διακριτές σημασίες αντιπροσωπεύονται και σε τόσα διακριτά σύνολα, με κάθε ζεύγος μορφής-σημασίας στο WordNet να είναι μοναδικό (Fellbaum,2005).
Όπως φαίνεται και στην Εικόνα 4.2., με την χρήση του WordNet online και την αναζήτηση συνωνύμων για την λέξη paint, έχουμε το παρακάτω αποτέλεσμα. Αρχικά παρουσιάζονται οι έννοιες της λέξης σε περίπτωση που είναι ουσιαστικό μαζί με μία πρόταση παραδείγματος. Το ίδιο και αν η λέξη είναι ρήμα. Βλέπουμε στη επεξήγηση ότι το S συμβολίζει τις σημασιολογικές σχέσεις του Synet και το W αντιπροσωπεύει τις λεξιλογικές σχέσεις της λέξης.

Εικόνα 4.2. Αποτέλεσμα αναζήτησης λέξης στο Wordnet.

4.4. Γράφοι Γνώσης

Οι γράφοι γνώσης (knowledge graphs) αποτελούν μέθοδο αναπαράστασης και οργάνωσης δεδομένων προερχόμενων από πολλαπλές και διαφορετικές πηγές. Η αναπαράσταση αυτή γίνεται με ευέλικτο και ομοιόμορφο τρόπο με τη χρήση υπαρχόντων προτύπων, τεχνικών και εργαλείων(Opdahl,2020). Υπόκεινται στο επίκεντρο της επιστημονικής έρευνας από το 2012, αν και ουσιαστικά η αναπαράσταση γνώσης με γραφήματα μελετάται εδώ και δεκαετίες (Ehrlinger & Wöß, 2016).

Με τη χρήση ενός μοντέλου που βασίζεται στη γραφική αναπαράσταση των δεδομένων καθίσταται δυνατή η ενσωμάτωση, η διαχείριση και η εξαγωγή γνώσης από τα δεδομένα αυτά, τα οποία μπορεί να προέρχονται από πολυπληθής και ποικίλες πηγές μεγάλης κλίμακας (Aidan et al., 2020). Τα δεδομένα μπορούν να προέρχονται ακόμη από δομημένες, ημιδομημένες ή και μη δομημένες πηγές. Η εξέλιξη και επεξεργασία τους πραγματοποιείται με ευέλικτο τρόπο (Aidan et al., 2020). Πρόκειται δηλαδή για ένα αφηρημένο πλαίσιο (abstract framework), το οποίο βασίζεται στην ανάλυση και στην εξαγωγή οντοτήτων και σχέσεων (Ehrlinger & Wöß, 2016;Tiwari et al., 2021) Πιο αναλυτικά θα αναφερθούμε παρακάτω.

4.4.1. Ορολογία

Ένας συγκεκριμένος ορισμός για το τί είναι γράφος γνώσης δεν υπάρχει στην βιβλιογραφία. Στα [(Aidan et al., 2020;Chaudhri et al., 2022;,4;Tiwari et al., 2021) ορίζεται ως ένα γράφημα δεδομένων, ένα κοινό πλαίσιο, το οποίο αποσκοπεί στην αναπαράσταση, συσσώρευση και στην μετατροπή της γνώσης που υπάρχει στον πραγματικό κόσμο σε κόμβους οι οποίοι αντιπροσωπεύουν οντότητες ενδιαφέροντος και σε ακμές οι οποίες αντιπροσωπεύουν τις πιθανές διαφορετικές σχέσεις μεταξύ των οντοτήτων αυτών. Παρέχεται έτσι μία συνεκτική και διαισθητική αφαίρεση (abstraction) για ένα πλήθος πεδίων (domains) όπου οι κόμβοι και τα μονοπάτια μεταξύ αυτών εκφράζουν τις διαφορετικές και πιθανότατα περίπλοκες σχέσεις μεταξύ των οντοτήτων και των πεδίων.

Η συσχέτιση των διαφόρων νοημάτων που εκφράζουν οι κόμβοι και οι ακμές μπορεί να γίνει αρχικά με την δήλωσή τους σε ένα αρχείο τεκμηρίωσης (documentation) με τη μορφή συμβολοσειράς και σε μία ανθρωπίνως κατανοητή γλώσσα. Σε προγραμματιστικό επίπεδο, η δήλωση αυτή μπορεί να πραγματοποιηθεί με τη χρήση μιας έγκυρης γλώσσας προδιαγραφών (formal specification language), όπως η λογική πρώτης τάξεως (first-order logic). Ένα τρέχον πεδίο της επιστημονικής έρευνας αποσκοπεί στην επίτευξη του αυτόματου υπολογισμού των νοημάτων, τα οποία αποτυπώνονται σε ένα διάνυσμα (vector) ως μία ακολουθία αριθμών (Chaudhri et al., 2022).

Η πληροφορία (όπως για παράδειγμα κόμβοι, οντότητες, συσχετίσεις κ.ο.κ.) προστίθεται σε έναν γράφο γνώσης συνδυάζοντας ανθρωποκεντρικές, ημιαυτόματες ή και πλήρως αυτοματοποιημένες μεθόδους. Βασική προϋπόθεση όμως αποτελεί η παραδοχή ότι ανεξαρτήτως μεθόδου, οι καταγεγραμμένες πληροφορίες είναι εύκολα κατανοητές και επαληθεύσιμες από τον άνθρωπο (Chaudhri et al., 2022).

4.4.2. Μοντέλα γράφων γνώσης

Όπως αναφέρθηκε και προηγουμένως, στους γράφους γνώσης τα δεδομένα μοντελοποιούνται ως γράφοι. Ένας γράφος γνώσης μπορεί να είναι της παρακάτω μορφής ((Aidan et al., 2020;Fensel et al., 2020):

1. Γράφοι κατευθυνόμενων ακμών (Directed Edge-labelled Graphs):

   Οι γράφοι κατευθυνόμενων άκμων ή πολυσχασιακοί γράφοι αποτελούνται από ένα σετ κόμβων και από ένα σετ κατευθυνόμενων άκμων με ετικέτες (directed labeled edges) μεταξύ των κόμβων αυτών. Όπως έχει ήδη αναφερθεί, οι κόμβοι αναπαριστούν οντότητες και οι ακμές αναπαριστούν δυαδικές σχέσεις μεταξύ των οντοτήτων αυτών. Στα τυπικά σχεσιακά μοντέλα το σχήμα της βάσης δεδομένων πρέπει να οριστεί εκ των προτέρων και η βάση να προγραμματιστεί με βάση αυτό. Η μοντελοποίηση με κατευθυνόμενους γράφους γνώσης όμως προσφέρει μεγαλύτερη ευελιξία στην ενσωμάτωση νέων δεδομένων και πηγών, καθώς τα δεδομένα δεν οργανώνονται ιεραρχικά και επιτρέπεται η κυκλική αναπαράσταση.
2. Ετερογενείς και Ομογενείς Γράφοι (Heterogeneous and Homogeneous Graphs):Σε αυτή την κατηγορία γράφων κάθε κόμβος και ακμή έχει έναν τύπο, ο οποίος αποτελεί κομμάτι του ίδιου του γράφου. Δεν ορίζεται δηλαδή ως ξεχωριστή σχέση. Αν η ακμή ενώνει δύο κόμβους του ίδιου τύπου, ονομάζεται ομογενής. Αν ενώνει δύο κόμβους διαφορετικού τύπου ονομάζεται ετερογενής. Σε αντίθεση με τους κατευθυνόμενους γράφους, οι ετερογενεις γράφοι έχουν μια σχέση ένα προς ένα μεταξύ των κόμβων και των τύπων (Aidan et al., 2020). Είναι δυνατός ο διαχωρισμός των κόμβων σύμφωνα με τον τύπο τους.
3. Γράφοι Ιδιοτήτων (Property Graphs):Με τους γράφους ιδιοτήτων υπάρχει η δυνατότητα ένα σετ από ιδιότητες-τιμές και μια ετικέτα να συσχετιστούν με κόμβους και ακμές. Τέτοιοι γράφοι χρησιμοποιούνται σε δημοφιλείς βάσεις δεδομένων γραφημάτων, όπως η Neo4j (Aidan et al., 2020).Με αυτό τον τρόπο αυξάνεται από τη μία πλευρά η ευελιξία και η κωδικοποίηση των δεδομένων ακόμα περισσότερο, από την άλλη όμως απαιτείται πιο περίπλοκος χειρισμός της βάσης.
4. Γράφος Συνόλου Δεδομένων (Graph Dataset): Ο γράφος συνόλου δεδομένων αποτελείται από ένα σετ επώνυμων γράφων και έναν προεπιλεγμένο γράφο (default graph). Κάθε επώνυμος γράφος αποτελείται από ένα αναγνωριστικό (ID) γράφου και έναν γράφο, ενώ ο προεπιλεγμένος γράφος δεν έχει ID και αναφέρεται "από προεπιλογή" εάν δεν έχει καθοριστεί ένα ID σε έναν γράφο(Aidan et al., 2020). Ο προεπιλεγμένος γράφος μπορεί να διαχειρίζεται τα meta-data των επώνυμων γράφων. Με τους γράφους συνόλου δεδομένων γίνεται δυνατή η διαχείριση και η αναζήτηση δεδομένων από πολλές πηγές, με κάθε πηγή να μπορεί να αναπαρίσταται και να είναι διαχειρίσιμη ως ένας ξεχωριστός γράφος (Aidan et al., 2020)προσφέροντας πληθώρα προγραμματιστικών επιλογών ανάλογα το σκοπό που θέλουμε να επιτεύξουμε.

5. Άλλα μοντέλα γράφων: Ένα άλλο μοντέλο γράφων αποτελεί η κωδικοποίηση με υπερκόμβους (hypernodes). Στο μοντέλο αυτό οι κόμβοι του γράφου μπορούν να περιέχουν ξεχωριστές ακμές ή και εμφωλευμένους γράφους. Υπάρχει ακόμα η δυνατότητα οι ακμές να συνδέουν σετ και όχι μόνο ζευγάρια κόμβων, με τη χρήση του μοντέλου υπεργράφου (hypergraph). Με την ευελιξία που προσφέρουν γενικά οι διάφορες κατηγορίες γράφων, είναι εν τέλη εφικτή η μετατροπή των δεδομένων από το ένα μοντέλο στο άλλο (Aidan et al., 2020).

4.5. Πρότυπο RDF (Resource Description Framework)

Το σύνηθες πρότυπο που χρησιμοποιούν οι γράφοι γνώσεων είναι το RDF (Resource Description Framework). Στην ουσία πρόκειται για ένα μοντέλο οργάνωσης δεδομένων. Με το μοντέλο αυτό καθίσταται δυνατός ο διαμοιρασμός, η επεξεργασία και η έκθεση δομημένων και αδόμητων δεδομένων μεταξύ διαφόρων εφαρμογών (https://www.w3.org/RDF/). Τα δεδομένα αναπαρίστανται ως γράφοι, με κόμβους (nodes) που αναπαριστούν αντικείμενα, πληροφορίες ή έννοιες και ακμές (edges) που αναπαριστούν σημασιολογικές σχέσεις (semantic relations). Δύο κόμβοι που συνδέονται με μία ακμή - δηλαδή σχέση- ονομάζονται τριάδα (triple).

Πιο συγκεκριμένα, μία τριάδα αποτελείται από ένα σημασιολογικό πόρο (υποκείμενο- subject), ο οποίος έχει μία συγκεκριμένη σημασιολογική σχέση (ιδιότητα- property) με έναν άλλο σημασιολογικό πόρο (αντικείμενο - object) ή απλή τιμή (literal). Οι πόροι και οι ιδιότητες προσδιορίζονται με τα αναγνωριστικά Internationalized Resource Names (IRN) και οι απλές τιμές με τη χρήση τύπου δεδομένων XML Schema Definition (XSD).

Το RDF είναι ένα πλαίσιο ανοιχτού κόσμου, επιτρέπει στον οποιοσδήποτε να κάνει δηλώσεις για οποιοδήποτε πόρο και είναι ανεξάρτητο από οποιαδήποτε συγκεκριμένη σύνταξη σειριοποίησης. Κατά τον σχεδιασμό μιας εφαρμογής πρέπει να λαμβάνεται αυτό υπόψη. Τα βασικότερα στοιχεία που χρησιμοποιεί το πρότυπο RDF για την αφηρημένη σύνταξή του είναι είναι ένα μοντέλο δεδομένων γραφήματος, ένα λεξιλόγιο που βασίζεται σε URI, οι τύποι των δεδομένων, οι τιμές literal, η σειριοποιημένη σύνταξη XML(μια συνιστώμενη φόρμα σειριοποίησης XML, η οποία μπορεί να χρησιμοποιηθεί για την κωδικοποίηση του μοντέλου δεδομένων για ανταλλαγή πληροφοριών μεταξύ των εφαρμογών) και η συνέπεια που χρειάζεται να υπάρχει όσο το δυνατόν περισσότερο.

Μια αναφορά URI ή κυριολεκτική λέξη που χρησιμοποιείται ως κόμβος προσδιορίζει τι αντιπροσωπεύει αυτός ο κόμβος. Μια αναφορά URI που χρησιμοποιείται ως κατηγόρημα προσδιορίζει μια σχέση μεταξύ των πραγμάτων που αντιπροσωπεύονται από τους κόμβους που συνδέει. Μια αναφορά URI κατηγορήματος μπορεί επίσης να είναι ένας κόμβος στο γράφημα. Ένας κενός κόμβος είναι ένας κόμβος που δεν είναι αναφορά URI ή κυριολεκτικό. Στην αφηρημένη σύνταξη RDF, ένας κενός κόμβος είναι απλώς ένας μοναδικός κόμβος που μπορεί να χρησιμοποιηθεί σε μία ή περισσότερες δηλώσεις RDF, αλλά δεν έχει εγγενές όνομα. Ο κενός αυτός κόμβος μπορεί να περιλαμβάνει ένα τοπικό αναγνωριστικό που διακρίνεται από όλα τα URI και literals και όταν τα γραφήματα συγχωνεύονται οι κόμβοι αυτοί διατηρούνται διακριτοί ώστε να μπορεί να διατηρηθεί το νόημα τους. Καθώς οι κενοί κόμβοι δεν αποτελούν μέρος της αφηρημένης σύνταξης RDF η αναπαράστασή τους εξαρτάται από τη συγκεκριμένη σύνταξη που χρησιμοποιείται στην κάθε εφαρμογή.
Περισσότερες πληροφορίες για την αφηρημένη σύνταξη και τα χαρακτηριστικά της RDF υπάρχουν διαθέσιμα εδώ RDF 1.2 Concepts and Abstract Syntax (w3.org).

4.6. Ανοιχτοί γράφοι γνώσης

Υπάρχουν αρκετοί ανοιχτοί διαθέσιμοι γράφοι γνώσης όπως οι DBpedia, YAGO και Freebase, και κατασκευάζονται συχνά από ημι-δομημένη γνώση, όπως αυτή που παρέχει η Wikipedia, ή συλλέγονται από τον Ιστό με συνδυασμό στατιστικών και γλωσσικών μεθόδων. Το αποτέλεσμα είναι μεγάλης κλίμακας γράφοι γνώσης οι οποίοι προσπαθούν να κάνουν μια καλή αντιστάθμιση μεταξύ πληρότητας και ορθότητας(Paulheim,2017).

Πιο συγκεκριμένα, ορισμένοι ανοιχτοί γράφοι γνώσης είναι οι ακόλουθοι (Fensel et al., 2020;Paulheim,2017):

1. DBpedia: Αποτελεί το βασικότερο κεντρικό σύνολο δεδομένων στον Σημασιολογικό Ιστό και είναι συνδεδεμένο με πολλά άλλα σύνολα δεδομένων (Fensel et al., 2020). Δημοσιεύτηκε πρώτη φορά το 2007. Με τη χρήση εξαγωγέων που ρυθμίζονται για την εξαγωγή διαφορετικών ειδών δεδομένων εξάγεται και ο Γράφος γνώσης από τα δομημένα δεδομένα που υπάρχουν στις σελίδες της Wikipedia.Η κύρια πηγή για αυτήν την εξαγωγή είναι τα ζεύγη κλειδιών-τιμών από τα πλαίσια πληροφοριών (infoboxes) της. Είναι χτισμένο πάνω από την οντολογία DBpedia και η έκδοση του Οκτωβρίου 2016 περιείχε

13B τριπλέτες RDF, ενώ η οντολογία DBpedia περιέχει 760 κλάσεις και περίπου 3000 ιδιότητες (Fensel et al., 2020).

2. OpenCyc και Cyc Ο γράφος γνώσης Cyc αναπτύχθηκε και διατηρείται από την CyCorp Inc, χρονολογείται από τη δεκαετία του 1980 είναι ένας από τα παλαιότερους γράφους γνώσης (Paulheim,2017). Η OpenCyc είναι η ελεύθερα διαθέσιμη έκδοση και συμπεριλαμβάνει ένα καταληκτικό σημείο προς τον Σημασιολογικό ιστό Semantic Web ο οποίος περιέχει συνδέσμους προς το DBpedia και άλλα σύνολα ανοιχτών συνδεδεμένων δεδομένων. Αποτελείται από 120.000 περιπτώσεις (instances) και 2,5 εκατομμύρια γεγονότα ορισμένα για τις περιπτώσεις αυτές. Ακόμα περιλαμβάνει μια ιεραρχία περίπου 45.000 τύπων και 19.000 πιθανών σχέσεων.
3. Freebase Είναι ένα δημόσιο, επεξεργάσιμος Γράφος γνώσης με πρότυπα σχημάτων για τα περισσότερα είδη πιθανών οντοτήτων (δηλ. πρόσωπα, πόλεις, ταινίες, κ.λπ.). Αγοράστηκε από την Google τον Μάρτιο του 2015 και έτσι έκλεισε η Freebase έκλεισε στις 31 Μαρτίου 2015. Περιέχει περίπου 50 εκατομμύρια οντότητες και 3 δισεκατομμύρια γεγονότα καθώς και περίπου 27.000 τύπους οντοτήτων και 38.000 τύπους σχέσεων.
4. Wikidata: Αφού τερματίστηκε η λειτουργία του Freebase τα δεδομένα του μεταφέρθηκαν στην Wikidata, που είναι επίσης ένα γράφημα γνώσης στο οποίο για κάθε αξίωμα, μπορούν να συμπεριληφθούν μεταδεδομένα προέλευσης. Περιέχει περίπου 16 εκατομμύρια περιπτώσεις και 66 εκατομμύρια δηλώσεις, με το σχήμα του να ορίζει περίπου 23.000 τύπους και 1.600 σχέσεις.
5. YAGO:Ο γράφος εξάγεται επίσης από τη Wikipedia και χτίζει την ταξινόμηση του από το σύστημα κατηγοριών της Wikipedia και τον λεξιλογικό πόρο WordNet. Οι ιδιότητες από τα infobox αντιστοιχίζονται χειροκίνητα σε ένα σταθερό σύνολο χαρακτηριστικών. Στοχεύει σε μια αυτόματη συγχώνευση γνώσης που εξάγεται από διάφορες γλωσσικές εκδόσεις της Wikipedia, χρησιμοποιώντας διαφορετικές ευρετικές μεθόδους. Κατά την έκδοση YAGO3, περιέχει 4,6 εκατομμύρια οντότητες και 26 εκατομμύρια στοιχεία για αυτούς τους τύπους και το σχήμα περιλαμβάνει περίπου 488.000 τύπους και 77 σχέσεις.
6. NELL: Εκτός από προσεγγίσεις που χρησιμοποιούν ημι-δομημένο περιεχόμενο ως βάση (π.χ. DBpedia και YAGO, που αναφέρονται προηγουμένως) έχουν προταθεί και μέθοδοι εξαγωγής γραφημάτων γνώσης από μη δομημένα δεδομένα. Μια απο αυτές είναι η μέθοδος NELL (Never Ending Language Learning), η οποία λειτουργεί σε διαδικτυακή κλίμακα και συγκεκριμένα σε ένα σύνολο ιστοσελίδων μεγάλης κλίμακας. Εκμεταλλεύεται μια συνδεδεμένη διαδικασία που μαθαίνει μοτίβα κειμένου αντίστοιχου τύπου και ισχυρισμούς σχέσεων, τα οποία

και εφαρμόζει για την εξαγωγή νέων οντοτήτων και σχέσεων (Paulheim,2017). Τα δεδομένα στο NELL μπορούν να μετασχηματιστούν σε RDF τριπλέτες και να παρέχονται επίσης ως Συνδεδεμένα Ανοικτά Δεδομένα. Το NELL περιέχει περίπου 2 εκατομμύρια οντότητες και 433.000 σχέσεις μεταξύ αυτών ενώ ορίζει 285 κλάσεις και 425 σχέσεις.

7. Knowledge Vault: Αποτελεί έργο της Google. Η γνώση εξάγεται από διαφορετικές πηγές, όπως έγγραφα κειμένου, πίνακες HTML και δομημένους σχολιασμούς (annotations) στον Ιστό με Microdata ή MicroFormats.Τα εξαγόμενα γεγονότα συνδυάζονται χρησιμοποιώντας τόσο τις τιμές εμπιστοσύνης του εξαγωγέα, όσο και τις προηγούμενες πιθανότητες για τις δηλώσεις, οι οποίες υπολογίζονται χρησιμοποιώντας το γράφημα γνώσης Freebase και στη συνέχεια από αυτές τις συνιστώσες, υπολογίζεται μια τιμή εμπιστοσύνης για κάθε γεγονός και μόνο τα σίγουρα γεγονότα λαμβάνονται στο Knowledge Vault (Paulheim,2017). Περιέχει περίπου 45 εκατομμύρια οντότητες και 271 εκατομμύρια δηλώσεις γεγονότων, ενώ χρησιμοποιεί 1.100 τύπους οντοτήτων και 4.500 τύπους σχέσεων.

5. Μεθοδολογία

Στο παρόν κεφάλαιο αναλύουμε τη μεθοδολογία που ακολουθήθηκε για την ανάπτυξη του προγράμματός μας στο πλαίσιο της διπλωματικής εργασίας. Γίνεται αρχικά αναφορά στη γλώσσα προγραμματισμού και στις βιβλιοθήκες που χρησιμοποίηθηκαν. Στη συνέχεια παρουσιάζεται το σύστημα διαχείρισης βάσεων δεδομένων γράφων Neo4j και η γλώσσα Cypher, με το οποίο δημιουργήθηκε και αποθηκεύτηκε ο Γράφος Γνώσης. Τέλος, αναπτύσσεται η διαδικασία που ακολουθήθηκε.

5.1. Γλώσσα , Βιβλιοθήκες και Πακέτα

Το πρόγραμμα αναπτύχθηκε με την γλώσσα προγραμματισμού Python στο περιβάλλον PyCharm. Επιλέχθηκε η γλώσσα Python, έκδοση 3.9 καθώς είναι πλέον η πιο διαδεδομένη για εργασίες τεχνητής νοημοσύνης και επεξεργασίας φυσικής γλώσσας, τόσο λόγω της ευκολίας στη συνταξή της, όσο και των πακέτων και των βιβλιοθηκών που υπάρχουν διαθέσιμα για την διευκόλυνση στην ανάπτυξη των εργασιών αυτών. Οι βιβλιοθήκες που χρησιμοποιήθηκαν για την ανάπτυξη του προγράμματος είναι οι Pandas, NumPy και το πακέτο Stanza. Παρακάτω θα δούμε πιο αναλυτικά τα χαρακτηριστικά των βιβλιοθηκών αυτών.

5.1.1. Βιβλιοθήκη Pandas

Το pandas είναι μια βιβλιοθήκη ανοικτού κώδικα, με άδεια χρήσης BSD(Berkeley Software Distribution),δηλαδή ανοιχτού κώδικα, που παρέχει υψηλή απόδοση, εύχρηστες δομές δεδομένων και εργαλεία ανάλυσης δεδομένων για τη γλώσσα προγραμματισμού Python (https://pandas.pydata.org/docs/index.html, ημερ. προσβ. 2/10/2023). Από αυτήν χρησιμοποιήθηκε συγκεκριμένα η δομή δεδομένων DataFrame. Πρόκειται για δισδιάστατα, μεταβλητά σε μέγεθος, δυνητικά ετερογενή δεδομένα σε μορφή πίνακα. Η δομή δεδομένων περιέχει επίσης επισημασμένους άξονες (γραμμές και στήλες). Οι αριθμητικές πράξεις ευθυγραμμίζονται και στις ετικέτες των γραμμών και των στηλών (https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html).

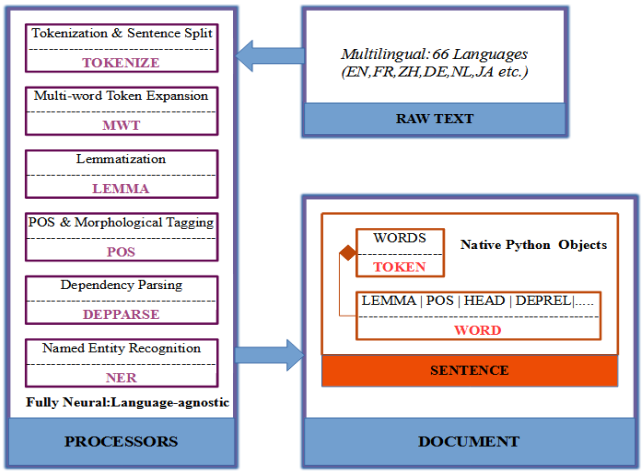
5.1.2. Βιβλιοθήκη NumPy

Η βιβλιοθήκη NumPy αποτελεί ένα βασικό πακέτο για επιστημονικούς υπολογισμούς στην Python. Παρέχει ένα πολυδιάστατο αντικείμενο πίνακα (multidimensional array object) και διάφορα παράγωγα αντικείμενα, καθώς και μια ποικιλία από ρουτίνες για διάφορες και γρήγορες πράξεις σε πίνακες, συμπεριλαμβανομένων μαθηματικών, λογικών, χειρισμού σχήματος, ταξινόμησης, επιλογής, εισόδου/εξόδου , διακριτούς μετασχηματισμούς Fourier, βασική γραμμική άλγεβρα, βασικές στατιστικές πράξεις, τυχαία προσομοίωση και πολλά περισσότερα (https://numpy.org/doc/stable/).

5.1.3. Πακέτο Stanza

Η Stanza αποτελεί ένα πακέτο (εργαλείο) ανάλυσης φυσικής γλώσσας για τη γλώσσα Python, ανοιχτού κώδικα. Όπως αναφέρεται αναλυτικά από τους Peng et al. (2020) , υποστηρίζει 66 γλώσσες και διαθέτει μια πλήρως νευρωνική διοχέτευση (έναν νευρωνικό αγωγό - neural pipeline) για ανάλυση κειμένου συμπεριλαμβανομένων των tokenization, lemmatization, part-of speech tagging, syntactic structure ,dependency parsing και named entity recognition. Το πακέτο έχει εκπαιδευτεί με 112 σύνολα δεδομένων,εκ των οποίων είναι τα Universal Dependencies treebanks, με την ίδια νευρωνική αρχιτεκτονική να γενικεύεται καλά και να επιτυγχάνει ανταγωνιστική απόδοση σε όλες τις γλώσσες που δοκιμάστηκαν. Ακόμα, παρέχει διεπαφή με το λογισμικό Java Stanford CoreNLP, με το οποίο εκτείνεται τη λειτουργικότητά του και μπορούν να καλυφθούν και άλλες εργασίες, όπως η εξαγωγή σχέσεων (relation extraction) και constituency parsing. Οι τεχνικές που χρησιμοποιήθηκαν για την εκπαίδευσλη του είναι τα νευρωνικά δίκτυα Bi-LSTM και η αρχιτεκτονική Sequence-to-Sequence.
Ο πηγαίος κώδικας και τα προεκπαιδευμένα μοντέλα μπορούν να βρεθούν εδώ https://stanfordnlp.github.io/stanza/ .

Παρακάτω, στην Εικόνα 5.1. (Peng et al., 2020).,φαίνεται σχηματικά ο αγωγός επεξεργασίας φυσικής γλώσσας του νευρωνικού δικτύου του πακέτου Stanza.

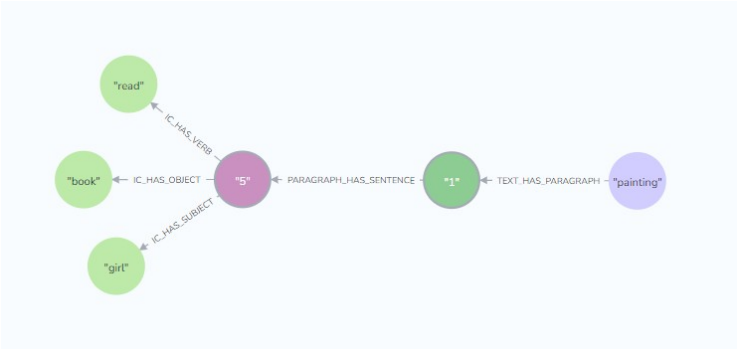


Εικόνα 5.1 Αγωγός επεξεργασίας φυσικής γλώσσας Stanza

5.2. Neo4j Graph database και Cypher

Για την δημιουργία του γράφου γνώσης επιλέχθηκε η βάση δεδομένων Neo4j. Ανήκει στην κατηγορία NoSQL, ενός τύπου συστήματος διαχείρισης βάσεων δεδομένων που έχει σχεδιαστεί για να χειρίζεται και να αποθηκεύει μεγάλους όγκους μη δομημένων και ημιδομημένων δεδομένων. Αναφέρεται σε «μη σχεσιακές» βάσεις δεδομένων που περιλαμβάνουν ένα ευρύ φάσμα διαφορετικών αρχιτεκτονικών βάσεων δεδομένων και μοντέλων δεδομένων. Η Neo4j ανήκει φυσικά στην κατηγορία της γραφικής αναπαράστασης.
Χρησιμοποιήθηκε η cloud εκδοχή της Neo4j, η AuraDB Free καθώς είναι ασφαλής, αξιόπιστη, πλήρως αυτοματοποιημένη, γρήγορη στα ερωτήματα (queries) και με πρόσβαση από οποιοδήποτε σημείο, επομένως προσφέρει μεγάλη ευελιξία κατά τον προγραμματισμό. Καθώς επιτρέπει τη δημιουργία μέχρι και 200000 κόμβων και 400000 σχέσεων, το μέγεθος δεν αποτελεί κανένα απολύτως πρόβλημα και μελλοντικά μπορούν να προστεθούν πολλά περισσότερα δεδομένα.

Ο γράφος είναι της μορφής όπως φαίνεται στην Εικόνα 5.2. όπου βλέπουμε ενδεικτικά τους κόμβους και τις σχέσεις μεταξύ τους. Θα μιλήσουμε αναλυτικότερα για αυτούς στην ενότητα της μοντελοποίησης του προβλήματος. Οι σχέσεις είναι κατευθυνόμενες ακμές από έναν κόμβο προς έναν άλλο. Οι κόμβοι είναι οι οντότητες σε ένα γράφημα.



Εικόνα 5.2. Κόμβοι και σχέσεις σε έναν γράφο Neo4j.

5.2.1. Γλώσσα Cypher

Η γλώσσα που χρησιμοποιείται στη Neo4j είναι η Cypher. Αναπτύχθηκε το 2011 από τους μηχανικούς της Neo4j ώστε να είναι ισοδύναμη με την SQL, αλλά για βάσεις δεδομένων γραφημάτων. Όπως αναφέρεται και στο επίσημο εγχειρίδιο της Cypher, παρέχει έναν οπτικό τρόπο αντιστοίχισης προτύπων και σχέσεων, βασιζόμενη στον ακόλουθο τύπο σύνταξης ascii-art:

```
(nodes)-[:CONNECT_TO]→(otherNodes)
```

όπου τα (nodes) αντιπροσωπεύουν τους κόμβους και το -[:CONNECT_TO]→ την σχέση που υπάρχει μεταξύ τους.
Με εντολές (clauses) όπως οι MATCH, RETURN, WITH, CREATE, DELETE κ.τ.λ. εκτελούνται τα ερωτήματα. Περισσότερες πληροφορίες και πιο αναλυτικές οδηγίες για την σύνταξη της γλώσσας μπορούν να βρεθούν στον επίσημο ιστότοπο https://neo4j.com/docs/cypher-manual/current/introduction/ .

5.3. Υλοποίηση του αλγόριθμου

Στην υποενότητα αυτή θα δούμε πως από τη γνώση σε δομημένη μορφή αναπαράχθηκε φυσικό κείμενο, δηλαδή τα βήματα που ακολουθήθηκαν για την δημιουργία του γράφου γνώσης μας και την ανάπτυξη του αλγορίθμου για την αναπαραγωγή κειμένου.

5.3.1 Σύνολο κειμενικών δεδομένων.

Τα κειμενικά δεδομένα που χρησιμοποιήθηκαν για τη δημιουργία δομημένης γνώσης και του γράφου γνώσης συγγράφηκαν από εμάς για τον σκοπό αυτό και είναι οι παρακάτω παράγραφοι που αποτελούνται από ανεξάρτητες προτάσεις. Μια ανεξάρτητη πρόταση είναι μια ομάδα λέξεων που περιέχει ένα υποκείμενο και ένα ρήμα και εκφράζει μια ολοκληρωμένη σκέψη. Μπορεί ακόμα να περιέχει ένα αντικείμενο και μια προθετική φράση (prepositional phrase) αλλά δεν είναι απαραίτητο, αρκεί το νόημα που εκφράζεται στην πρόταση να είναι ολοκληρωμένο και να στέκει από μόνο του.

Paragraph 1:

1. I love art.
2. I can express emotions.
3. The painting will be a landscape.
4. Trees burst colors.
5. A girl reading a book.
6. Clouds float the sky.
7. Everything will be calm.

Paragraph 2:

1. I read books.
2. Books inspire me.
3. Stories provoke emotions.
4. Authors challenge society.
5. Characters defy life.
6. I feel comfort.
7. Life is magical.

5.3.2. Επεξεργασία κειμενικών δεδομένων

Προκειμένου να δημιουργηθεί ο γράφος γνώσης πρέπει πρώτα να χωριστούν οι προτάσεις στα συστατικά τους μέρη και μέρη του λόγου, δηλαδή να πραγματοποιηθεί Dependency Parsing και Part of Speech Tagging. Για τον σκοπό αυτό χρησιμοποιήθηκε

το εργαλείο Stanza. Το εργαλείο εγκαθίσταται με την εντολή pip install stanza και το κάνουμε import στο περιβάλλον PyCharm. Χρησιμοποιούμε στη συνέχεια τον νευρωνικό αγωγό (neural pipeline) του εργαλείου, συγκεκριμενοποιώντας τους επεξεργαστές (Processors) ανάλογα με την εργασία που θέλουμε να εκτελέσουμε. Έστω ότι έχουμε την πρόταση "I love art" .
Για το Dependency Parsing η ανάλυση εξάρτησης εκτελείται από τον DepparseProcessor και καλείται με το όνομα depparse. Θα χρησιμοποιήσουμε τον αγωγό ως εξής:

```
s = "I love art"
nlp = stanza.Pipeline(lang='en', processors='tokenize,mwt,pos,lemma,depparse')
doc = nlp(s)
```

και το αποτέλεσμα του Dependency Parsing θα είναι :

```
id: 1   word: I head id: 2  head: love  deprel: nsubj
id: 2   word: love  head id: 0  head: root  deprel: root
id: 3   word: art   head id: 2  head: love  deprel: obj
```

Η λέξη love είναι η ρίζα του δέντρου, δηλαδή το ρήμα, η λέξη I το subject και έχει ως κεφαλή τη λέξη love, ενώ η λέξη art είναι το object και έχει ως κεφαλή τη λέξη love.
Παρόμοια και για το Part of Speech Tagging, η ανάλυση των μερών του λόγου εκτελείται από τον POSProcessor και καλείται με το όνομα pos. Θα χρησιμοποιήσουμε τον αγωγό ως εξής:

```
nlp = stanza.Pipeline(lang='en', processors='tokenize,mwt,pos')
doc = nlp(s)
```

και το αποτέλεσμα του Part of Speech Tagging θα είναι :

```
word: I upos: PRON  xpos: PRP   feats: Case=Nom|Number=Sing|Person=1|PronType=Prs
word: love  upos: VERB  xpos: VBP   feats: Mood=Ind|Tense=Pres|VerbForm=Fin
word: art   upos: NOUN  xpos: NN    feats: Number=Sing
```

Η λέξη I έχει Universal POS tag PRON, δηλαδή pronoun, και χαρακτηριστικά όπως Number, Person, PronType κ.ο.κ. Κάθε λέξη έχει να δικά της χαρακτηριστικά ανάλογα σε ποιο μέρος του λόγου ανήκει.

Αυτή τη διαδικασία ακολουθήσαμε για όλες τις προτάσεις των δεδομένων μας.

5.3.3. Μοντελοποίηση του προβλήματος

Ανεξάρτητα από τα κειμενικά δεδομένα που έχουμε στη διάθεσή μας, θέλουμε το μοντέλο μας να μπορεί να ανταποκριθεί σχεδόν σε όλες τις περιπτώσεις κειμένων και προτάσεων. Η δομή ενός κειμένου φυσικής γλώσσας αποτελείται συνήθως από έναν τίτλο, τις παραγράφους του κειμένου, τις προτάσεις τις κάθε παραγράφου και τις λέξεις που συνθέτουν την εκάστοτε πρόταση. Με βάση αυτή την λογική έγινε και η μοντελοποίηση του προβλήματός μας.

Ο γράφος περιέχει τις εξής οντότητες:

1. Word: Αντιπροσωπεύει τις λέξεις που υπάρχουν σε ένα κείμενο. Κάθε λέξη με τις ιδιότητες της υπάρχει μόνο μία φορά στον γράφο και ανάλογα σε ποια πρόταση εμφανίζεται, συνδέεται με αυτή με την κατάλληλη σχέση. Στον γράφο μας υπάρχουν συνολικά 34 λέξεις.
2. Sentence: Αντιπροσωπεύει τις προτάσεις που υπάρχουν στο κείμενο.
3. Paragraph: Αντιπροσωπεύει τις παραγράφους του κειμένου.
4. Text: Αντιπροσωπεύει το κείμενο.

Οι σχέσεις που υπάρχουν μεταξύ των οντοτήτων είναι οι εξής:

1. TEXT_HAS_PARAGRAPH: Συνδέει μια οντότητα Text με μία οντότητα Paragraph, δηλαδή ένα κείμενο με τις παραγράφους του.
2. PARAGRAPH_HAS_SENTENCE: Συνδέει μια οντότητα Paragraph με μία οντότητα Sentence, δηλαδή μία παράγραφο με τις προτάσεις της.
3. IC_HAS_VERB: Συνδέει μια οντότητα Sentence με μία οντότητα Word και πιο συγκεκριμένα μια πρόταση με την λέξη που αποτελεί το verb της.
4. IC_HAS_MODAL_VERB: Συνδέει μια οντότητα Sentence με μία οντότητα Word και πιο συγκεκριμένα μια πρόταση με την λέξη που αποτελεί το modal verb της.
5. IC_HAS_SUBJECT: Συνδέει μια οντότητα Sentence με μία οντότητα Word και πιο συγκεκριμένα μια πρόταση με την λέξη που αποτελεί το subject της.
6. IC_HAS_OBJECT Συνδέει μια οντότητα Sentence με μία οντότητα Word και πιο συγκεκριμένα μια πρόταση με την λέξη που αποτελεί το object της.

Όλες οι παραπάνω οντότητες και οι σχέσεις φαίνονται στην Εικόνα 5.2.

5.3.4. Ιδιότητες οντοτήτων και σχέσεων

Όλοι οι κόμβοι και οι σχέσεις του γράφου έχουν ένα μοναδικό id. Κάθε οντότητα έχει τις ιδιότητες της ανάλογα του τι αντιπροσωπεύει:

1. Οι οντότητες Text έχουν τις ιδιότητες:
   author: ο συγγραφέας του κειμένου,
   title: ο τίτλος του κειμένου.

2. Οι οντότητες Sentence έχουν τις ιδιότητες:
   number: ο αριθμός της πρότασης,
   paragr: ο αριθμός της παραγράφου που ανήκει η πρόταση.
3. Οι οντότητες Paragraph έχουν τις ιδιότητες:
   number: ο αριθμός της παραγράφου.

4. Οι οντότητες Word έχουν τις ιδιότητες:
   upos: το Universal POS tag της λέξης, δηλαδή σε ποιο μέρος του λόγου ανήκει,
   w: η λέξη,
   wp: ο πληθυντικός της λέξης αν υπάρχει.
   Σε περίπτωση που το upos μιας λέξης είναι verb υπάρχουν οι εξής επιπλέον ιδιότητες στις οποίες αποθηκεύονται οι χρόνοι του ρήματος:
   continuous: συνεχής χρόνος,
   past: παρελθοντικός χρόνος,
   presentparticiple: μετοχή ενεστώτα,
   simplepresent1st: 1ο πρόσωπο ενεστώτα,
   simplepresent3rd: 3ο πρόσωπο ενεστώτα,
   pastparticiple: παρελθοντική μετοχή.

Έχουμε αποθηκεύσει μόνο τα πρόσωπα στα οποία υπάρχουν αλλαγές στις καταλήξεις των ρημάτων, ουσιαστικά στο 1ο και στο 3ο του simple present.
Για παράδειγμα, η λέξη “float” έχει τις εξής ιδιότητες:

Node properties
<id>: 25
continuous: "floating"
past: "floated"
pastparticiple: "floated"
presentparticiple: "floating"
simplepresent1st: "float"
simplepresent3rd: "floats"
upos: "verb"
w: "float"

Όπως οι οντότητες, έτσι και κάθε σχέση έχει τις δικές της ιδιότητες:

1. Οι σχέσεις TEXT_HAS_PARAGRAPH έχουν την ιδιότητα:
   paranum: ο αριθμός της παραγράφου.
2. Οι σχέσεις PARAGRAPH_HAS_SENTENCE δεν έχουν κάποια ιδιότητα.

3. Οι σχέσεις IC_HAS_VERB και IC_HAS_MODAL_VERB έχουν τις ιδιότητες:
   number: ο αριθμός του ρήματος, ενικός ή πληθυντικός,
   person: το πρόσωπο του ρήματος,
   sentence: η πρόταση που ανήκει το ρήμα,
   tense: ο χρόνος του ρήματος.
4. Οι σχέσεις IC_HAS_OBLECT και IC_HAS_SUBJECT έχουν τις ιδιότητες:
   sentence: ο αριθμός της πρότασης,
   det: το προσδιοριστικό της λέξης
   number: παίρνει την τιμή plural αν η λέξη βρίσκεται στον πληθυντικό.

Για παράδειγμα η σχέση IC_HAS_VERB της 7ης πρότασης της 1ης παραγράφου έχει τις εξής ιδιότητες :

Relationship properties
<id>: 27
number: "singular"
person: "3rd"
sentence: "7"
tense: "infinitive"

5.3.5. Δημιουργία του γράφου

Αφού έχουμε μοντελοποιήσει το πρόβλημά μας, προχωράμε στη δημιουργία του γράφου στο περιβάλλον Neo4j AuraDB Free. Η μοντελοποίηση έγινε λαμβάνοντας υπόψη το περιβάλλον και τις δυνατότητές του. Με τη γλώσσα Cypher δημιουργήθηκαν οι κόμβοι και οι σχέσεις μεταξύ τους.

Με την εντολή CREATE δημιουργούνται οι κόμβοι και οι σχέσεις του γράφου μαζί με τις ιδιότητες τους. Για παράδειγμα, αν θέλουμε να δημιουργήσουμε έναν κόμβο που αντιπροσωπεύει την δεύτερη πρόταση της πρώτης παραγράφου, δηλαδή μιας οντότητας Sentence με τις ιδιότητες number = 2 και paragr = 1, η εντολή θα είναι η εξής:

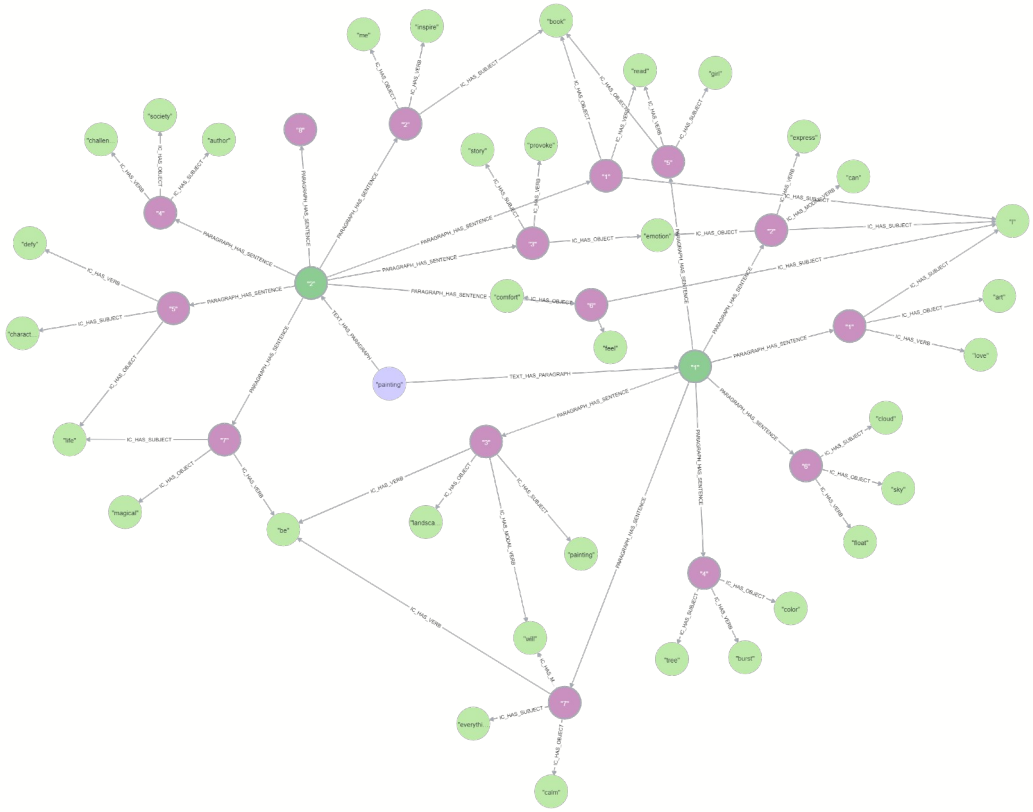
```
CREATE (s:Sentence {number:'2', paragr: '1'}).
```

Αν θέλουμε να δημιουργήσουμε μία σχέση ανάμεσα σε δύο οντότητες, για παράδειγμα τη σχέση ότι η πρώτη πρόταση της πρώτης παραγράφου έχει υποκείμενο τη λέξη I η εντολή θα είναι η εξής:

```
MATCH(k{w:'I'}), (s:Sentence) WHERE s.number='1' AND s.paragr ='1'
CREATE (s)-[r:IC_HAS_SUBJECT{sentence:'1'}]->(w).
```

Βρίσκουμε τον κόμβο με την ιδιότητα w = I και τον κόμβο οντότητας Sentence με ιδιότητες number:'2', paragr: '1' και δημιουργούμε τη σχέση [r:IC_HAS_SUBJECT] με ιδιότητα sentence:'1' .

Όμοια δημιουργούμε όλους τους κόμβους και τις σχέσεις τους σύμφωνα με το μοντέλο μας παραπάνω και ο γράφος γνώσης είναι αυτός που φαίνεται στην Εικόνα 5.3, μαζί με τις οντότητες του και τις σχέσεις που υπάρχουν μεταξύ τους.



Εικόνα 5.3. Ο γράφος γνώσης του προβλήματός μας.

5.3.6. Σύνδεση του Γράφου στην Python και προσθήκη επιπλέον δεδομένων

Η σύνδεση του γράφου με την Python γινεται πολύ απλά. Πρώτα εγκαθιστούμε την Neo4j στο περιβάλλον με την εντολή pip install neo4j και μετά εισάγουμε το στοιχείο GraphDatabase με την εντολή

```
from neo4j import GraphDatabase
```

Η σύνδεση με τον driver του γράφου και η έναρξη μιας συνεδρίας γίνεται με τις παρακάτω εντολές:

```
graphdb=GraphDatabase.driver(uri="neo4j+s://737e5ae4.databases.neo4j.io",
auth=("neo4j", "password"))
session = graphdb.session()
```

Την δεύτερη παράγραφο του κειμένου μας την προσθέσαμε με την χρήση της Python και τις κατάλληλες εντολές ανάλογα τις οντότητες που θέλουμε να δημιουργήσουμε. Παρακάτω βλέπουμε ενδεικτικά μερικές από αυτές:

```
s1 = session.run("CREATE (p:Paragraph {number:'2'})")
s2 = session.run("MATCH(t:Text{title:'painting'}), (p:Paragraph{number:'2'})
CREATE(t)-[r:TEXT_HAS_PARAGRAPH{paranum:'2'}]->(p)")
s3 = session.run("WITH ['1','2','3','4','5','6','7','8'] AS numbers FOREACH (value IN
numbers | CREATE (:Sentence{number:value, paragr:'2'})) ")
```

Ξεκινάμε μία συνεδρία με το s1 = session.run και μέσα στην παρένθεση συντάσσουμε την εντολή Cypher.

5.3.7 Αναπαραγωγή του φυσικού κειμένου

Αφού έχουμε δομήσει τη γνώση μας σε γράφο, σειρά έχει η συγγραφή του κώδικα για την αναπαραγωγή φυσικού κειμένου. Στο πρόγραμμά μας δημιουργήσαμε ένα ξεχωριστό Python File με το όνομα sentence.
Το πρόγραμμα τμηματοποιήθηκε σε τόσες συναρτήσεις ώστε να υπάρχει μελλοντικά η δυνατότητα ευελιξίας του και καθώς προσθέτονται περισσότερες οντότητες και πιο σύνθετες σχέσεις και προτάσεις να είναι πιο εύκολος ο εμπλουτισμός του χωρίς να χρειάζεται μεγάλη τροποποίηση στη δομή του.

Ο αλγόριθμος αποτελείται από τις παρακάτω συναρτήσεις:

1. def run_cypher_query(query, parameters=None): με αυτή την συνάρτηση τρέχουμε τα Cypher queries στον Neo4j γράφο μας. Αποτελεί έναν διαφορετικό τρόπο από αυτόν που είδαμε παραπάνω, χωρις καμία ουσιαστική διαφορα, τον οποίο υλοποιήσαμε απλά για να εξετάσουμε και αυτή την εναλλακτική.
2. allsent(): Με τη συνάρτηση allsent() επιστρέφονται ουσιαστικά όλες οι προτάσεις που υπάρχουν στην γράφο, και πιο συγκεκριμένα τα p.number as paragraph, s.number as snumber, k.w as word, type(r) as relation, r.tense as tense, r.person as person, r.number as wnumber, k.wp as number, δηλαδή ο αριθμός της παραγράφου που ανήκει η πρόταση, ο αριθμός της πρότασης, η λέξη με την κάθε σχέση που τη συνδέει με την πρόταση , ο χρόνος και το πρόσωπο του κάθε

ρήματος, ο αριθμός κάθε λέξης (ενικός ή πληθυντικός) και τέλος σε περίπτωση που μια λέξη βρίσκεται στον πληθυντικό, η αντίστοιχη μορφή της. Όλα τα δεδομένα επιστρέφονται με τη μορφή ενός DataFrame:

0	1	7	calm	...	None	None	None
1	1	7	will	...	singular	None	None
2	1	7	be	...	singular	None	None
3	1	7	everything	...	None	None	None
4	1	6	float	...	plural	None	None
5	1	6	sky	...	None	the	skies
6	1	6	cloud	...	plural	None	clouds
7	1	5	read	...	singular	None	None
8	1	5	book	...	None	a	books
9	1	5	girl	...	None	a	girls
10	1	4	burst	...	plural	None	None
11	1	4	color	...	plural	None	colors
12	1	4	tree	...	plural	None	trees
13	1	3	will	...	singular	None	None
14	1	3	be	...	singular	None	None
15	1	3	landscape	...	None	a	landscapes
16	1	3	painting	...	None	None	paintings
17	1	2	can	...	singular	None	None
18	1	2	express	...	singular	None	None
19	1	2	emotion	...	plural	None	emotions
20	1	2	I	...	None	None	None
21	1	1	art	...	singular	None	None
22	1	1	love	...	singular	None	None
23	1	1	I	...	None	None	None
24	2	7	be	...	singular	None	None
25	2	7	magical	...	singular	None	None
26	2	7	life	...	singular	None	lives
27	2	4	challenge	...	plural	None	None

3. senten(df, p, s): Η συνάρτηση senten παίρνει ως παραμέτρους ένα DataFrame df ,και συγκεκριμένα αυτό που περιέχει όλες τις προτάσεις , τον αριθμό μιας συγκεκριμένης παραγράφου p και τον αριθμό μιας συγκεκριμένης πρότασης s, και επιστρέφει ένα DataFrame με τα στοιχεία της συγκεκριμένης πρότασης.

	paragraph	snumber	word	relation	...	person	wnumber	det	number
4	1	6	float	IC_HAS_VERB	...	3rd	plural	None	None
5	1	6	sky	IC_HAS_OBJECT	...	None	None	the	skies
6	1	6	cloud	IC_HAS_SUBJECT	...	None	plural	None	clouds

4. word(b): Η συνάρτηση αυτή δέχεται ως παράμετρο μια συγκεκριμένη λέξη και επιστρέφει όλες τις ιδιότητές της.

	presentparticiple	simplepresent3rd	...	upos	pastparticiple
0	floating	floats	...	verb	floated

5. wordsent(s,r): Η συνάρτηση αυτή δέχεται ως παράμετρο ένα Dataframe, και πιο συγκεκριμένα ένα dataframe που περιέχει τα στοιχεία μιας συγκεκριμένης πρότασης και μια συμβολοσειρά (string) που αντιπροσωπεύει μία σχέση στον γράφο, για παράδειγμα IC_HAS_VERB. Επιστρέφει την λέξη που αντιστοιχεί στην εκάστοτε σχέση, δηλαδή το ρήμα, το αντικείμενο της πρότασης κ.ο.κ.

6. wtense(s, r): Η συνάρτηση αυτή δέχεται ως παράμετρο ένα Dataframe, και πιο συγκεκριμένα ένα dataframe που περιέχει τα στοιχεία μιας συγκεκριμένης πρότασης και μια συμβολοσειρά (string) που αντιπροσωπεύει μία σχέση στον γράφο, για παράδειγμα IC_HAS_VERB και επιστρέφει τον χρόνο - tense της λέξης, π.χ. του ρήματος.
7. wperson(s, r): Όμοια με τις δύο παραπάνω συναρτήσεις και η συγκεκριμένη δέχεται ως παράμετρο ένα Dataframe, το dataframe που περιέχει τα στοιχεία μιας συγκεκριμένης πρότασης και μια συμβολοσειρά (string) που αντιπροσωπεύει μία σχέση στον γράφο, για παράδειγμα IC_HAS_VERB και επιστρέφει το πρόσωπο-person στο οποίο βρίσκεται η λέξη. Τα αποτελέσματα των συναρτήσεων wordsent, wtense και wperson είναι της μορφής:

```
['float']
['simplepresent1st']
['3rd']
```

8. righttense(t, p, df): Η συνάρτηση αυτή δέχεται ως παραμέτρους τον χρόνο t, το πρόσωπο p και ένα dataframe με τις ιδιότητες μιας συγκεκριμένης λέξης και επιστρέφει τον σωστό χρόνο , π.χ. ενός ρήματος.
9. rightnu(df,r):Οι παράμετροι της συνάρτησης αυτής είναι το dataframe που περιέχει τα στοιχεία μιας συγκεκριμένης πρότασης και μια συμβολοσειρά που αντιπροσωπεύει μία σχέση στον γράφο, για παράδειγμα IC_HAS_OBJECT και επιστρέφει την λέξη στον σωστό αριθμό, σε περίπτωση που αυτή βρίσκεται στον πληθυντικό στην συγκεκριμένη πρόταση.
10. det (df,r): Η συνάρτηση αυτή δέχεται ως είσοδο το dataframe που περιέχει τα στοιχεία μιας συγκεκριμένης πρότασης και μια συμβολοσειρά που αντιπροσωπεύει μία σχέση στον γράφο και επιστρέφει το προσδιοριστικό - determiner της λέξης αν αυτό υπάρχει.

Τέλος, οι δύο συναρτήσεις που χρησιμοποιούν τις παραπάνω για να επιστρέψουν τις λέξεις στην κατάλληλη πτώση, αριθμό κτλ είναι οι εξής:

1. wordsentenv (df,r): Η συνάρτηση αυτή δέχεται ως παράμετρο το το dataframe που περιέχει τα στοιχεία μιας συγκεκριμένης πρότασης και ένα string που αντιπροσωπεύει μία σχέση στον γράφο τύπου IC_HAS_VERB ή IC_HAS_MODAL_VERB και επιστρέφει το verb ή το modal verb στον σωστό του χρόνο, αυτόν στον οποίο βρίσκεται στην πρόταση. Στην συνάρτηση καλείται αρχικά η συνάρτηση wordsent ώστε να επιστραφεί αρχικά η λέξη στη γενική της μορφή, μετά ελέγχεται αν η λέξη υπάρχει και δεν είναι η θέση στο dataframe

κενή (στην περίπτωση που δεν υπάρχει modal verb παραδείγματος χάρη) και με την χρήση των συναρτήσεων wtense, wperson, word και τέλος righttense, τις οποίες αναλύσαμε παραπάνω, επιστρέφεται η λέξη στον σωστό της χρόνο, αυτόν της πρότασης.

2. wordsentenw(df,r): Η συνάρτηση αυτή δέχεται ως παράμετρο το dataframe που περιέχει τα στοιχεία μιας συγκεκριμένης πρότασης και ένα string που αντιπροσωπεύει μία σχέση στον γράφο τύπου IC_HAS_OBJECT ή IC_HAS_SUBJECT. Καλείται η συνάρτηση rightnu και επιστρέφεται η λέξη στον σωστό της αριθμό.

Για την εμφάνιση της πρότασης ορίζουμε 4 μεταβλητές με τις σχέσεις που έχουμε, robj = "IC_HAS_OBJECT" για το αντικείμενο, rsubj= "IC_HAS_SUBJECT" για το υποκείμενο, rverb = "IC_HAS_VERB" για το ρήμα και rmodverb = "IC_HAS_MODAL_VERB" για το modal verb. Στη συνέχεια καλούμε τη συνάρτηση o = senten(allsent(), p="1", s="6") και στα ορίσματα p και s θέτουμε τον αριθμό της παραγράφου και της πρότασης που θέλουμε. Έπειτα καλούμε τις κατάλληλες συναρτήσεις ώστε να μας επιστραφούν τα στοιχεία της πρότασης:

1. verb = wordsentenv(o,rverb)
2. modal_verb = wordsentenv(o,rmodverb)
3. subject = wordsentenw(o,rsubj)
4. object = wordsentenw(o,robj)
5. detero = det(o, robj)
6. deters = det(o, rsubj)

Τέλος, όπως έχουμε ήδη αναφέρει, η δομή μιας απλής πρότασης είναι Υποκείμενο – Ρήμα- Αντικείμενο, επομένως εμφανίζουμε τα συστατικά μέρη στη σωστή σειρά με μια εκτύπωση print(deters,subject,modal_verb,verb,detero,object). Το αποτέλεσμα για την έκτη πρόταση της πρώτης παραγράφου θα είναι το ακόλουθο:

```
clouds  float the sky
```

Ολόκληρος ο κώδικας υπάρχει διαθέσιμος στο Παράρτημα.

6. Επίλογος

6.1. Σύνοψη και συμπεράσματα

Για την εκπόνηση της διπλωματικής εργασίας χρησιμοποιήσαμε περιορισμένο αριθμό κειμενικών δεδομένων τα οποία συγγράψαμε για αυτό το σκοπό. Τα δεδομένα είναι στην Αγγλική γλώσσα και αποτελούνται από απλές προτάσεις της μορφής Υποκείμενο - Ρήμα - Αντικείμενο , μορφή που αντιστοιχίζεται εύκολα με αυτήν που απαιτείται για την αναπαράσταση γνώσης με γράφους γνώσης. Η αναπαράσταση αυτή είναι κόμβοι που αντιπροσωπεύουν οντότητες από ένα πεδίο ενδιαφέροντος και ακμές οι οποίες αντιπροσωπεύουν τις πιθανές διαφορετικές σχέσεις μεταξύ των οντοτήτων αυτών. Παρέχεται έτσι ένας ευέλικτος και διαισθητικός τρόπος δόμησης ο οποίος όμως βασίζεται σε υπάρχοντα πρότυπα και τεχνικές και είναι εύκολα κατανοητός τόσο από τον άνθρωπο όσο και από τις μηχανές. Με τη χρήση του περιβάλλοντος Neo4j Aura DB Free και τη γλώσσα Cypher δημιουργήθηκε ο γράφος γνώσης που αντιστοιχεί στα κειμενικά δεδομένα, αφού πρώτα πραγματοποιήθηκε συντακτική και σημασιολογική ανάλυση σε αυτά με το πακέτο Stanza και αποφασίστηκαν οι οντότητες και οι σχέσεις που υπάρχουν μεταξύ τους λαμβάνοντας ως δεδομένο το γεγονός ότι το μοντέλο πρέπει να μπορεί να ανταποκριθεί σχεδόν σε όλες τις περιπτώσεις κειμένων και προτάσεων. Μετά τη δημιουργία του γράφου, γίνεται σύνδεσή του με το περιβάλλον PyCharm και τη γλώσσα Python. Με τις κατάλληλες συναρτήσεις και επεξεργασία των δεδομένων μας αναπαράγουμε φυσικό κείμενο από την δομημένη σε γράφο γνώσης γνώση.

Η αναπαράσταση αυτή βοήθησε πάρα πολύ τόσο στην κατανόηση όσο και στην μοντελοποίηση του προβλήματος αφού μπορεί να γίνει κατανοητή από τον οποιονδήποτε, είτε έχει επιστημονικές γνώσεις, είτε όχι. Στην περίπτωσή μας τα δεδομένα ήταν μικρά, σε περιπτώσεις που ο όγκος τους είναι πολύ μεγαλύτερος και πολύ πιο σύνθετος κάτι τέτοιο γίνεται ακόμα πιο εμφανές.

6.2. Όρια και περιορισμοί της έρευνας

Ένας ακόμα αρχικός στόχος της διπλωματικής ήταν να επιχειρηθεί η ανάποδη εργασία, δηλαδή από το φυσικό κείμενο να προκύψει δομημένη αναπαράσταση γνώσης. Κάτι τέτοιο δεν επιχειρήθηκε λόγω περιορισμού του χρόνου καθώς και της τεχνολογικής ισχύς που υπήρχε στην διάθεση μας. Τέλος, ένας άλλος περιορισμός αποτελεί το δεδομένο ότι στην αγγλική γλώσσα υπάρχουν λέξεις οι οποίες έχουν πολλές σημασίες, χρησιμοποιούνται ως διαφορετικά μέρη του λόγου ανάλογα την θέση, τη σημασία κ.ο.κ. που έχουν στην πρόταση, για τον λόγο αυτό χρησιμοποιήθηκαν απλά κειμενικά δεδομένα,ώστε να ελαχιστοποιηθεί όσο γίνεται ο παράγοντας αυτός. Αυτό δεν σημαίνει ότι μελλοντικά δεν υπαρχει η δυνατότητα διευθέτησης αυτού του θέματος και εμπλουτισμού του γράφου γνώσης, κάτι που αναλύεται περισσότερο ακριβώς παρακάτω.

Τέλος, όσον αφορά το βιβλιογραφικό κομμάτι, αν και υπήρχαν πολυπληθής πηγές και τα δεδομένα μας ήταν στα αγγλικά επομένως η έρευνα δεν ήταν δύσκολη, σε αρκετά άρθρα ή/και βιβλία που κάλυπταν το θέμα τόσο θεωρητικά όσο και προγραμματιστικά δεν υπήρχε πλήρης πρόσβαση στο κείμενο των δημοσιεύσεων αυτών.

6.3.Μελλοντικές Επεκτάσεις

Όπως έχουμε ήδη αναφέρει, η χρήση των γράφων γνώσης για αναπαράσταση γνώσης σε πάρα πολλούς τομείς αλλά και για διαφορετικούς σκοπούς όπως ακαδημαϊκούς, επιρρηματικούς κτλ έχει ανθίσει τα τελευταία χρόνια, αν και υπήρχε σαν έννοια εδώ και δεκαετίες. Στο πλαίσιο της δικής μας διπλωματικής και στην χρήση τους για εργασίες επεξεργασίας φυσικής γλώσσας υπάρχουν αρκετά περιθώρια βελτίωσης. Αρχικά μπορεί να επεκταθεί ο γράφος γνώσης με περισσότερα ήδη προτάσεων , όπως σύνθετες προτάσεις (Compound Sentences) αλλά και περίπλοκες (Complex Sentences) οι οποίες περιέχουν περισσότερα μέρη του λόγου και είναι πιο σύνθετες συντακτικά. Τέλος, όπως αναφέραμε και προηγουμένως, μια πολύ σημαντική μελλοντική εργασία που μπορεί να πραγματοποιηθεί ως συνέχεια της διπλωματικής αυτής είναι από το φυσικό κείμενο να προκύψει δομημένη αναπαράσταση γνώσης. Τόσο ο εμπλουτισμός όσο και η παραγωγή δομημένης αναπαράστασης, μπορεί να συνεισφέρει και να οδηγήσει και σε άλλες διεργασίες, όπως η παραγωγή της περίληψης ενός κειμένου με εντοπισμό των σημαντικότερων οντοτήτων και σχέσεων, ενδεχομένως την μετάφραση ενός κειμένου αλλά και την απάντηση ερωτημάτων όσον αφορά τα κειμενικά δεδομένα.

Βιβλιογραφία

Aidan, H., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutiérrez, C., ... & Zimmermann, A. (2020). Knowledge Graphs (Doctoral dissertation, Mines Saint-Etienne).

Al-Aswadi, F. N., Chan, H. Y., & Gan, K. H. (2020). Automatic ontology construction from text: a review from shallow to deep learning trend. *Artificial Intelligence Review*, *53*(6), 3901-3928.

Asim, M. N., Wasim, M., Khan, M. U. G., Mahmood, W., & Abbasi, H. M. (2018). A survey of ontology learning techniques and applications. *Database*, *2018*.

Balakrishnama, S., & Ganapathiraju, A. (1998). Linear discriminant analysis-a brief tutorial. *Institute for Signal and information Processing*, *18*(1998), 1-8.

Browarnik, A., & Maimon, O. (2015). Ontology learning from text: why the ontology learning layer cake is not viable. *International Journal of Signs and Semiotic Systems (IJSSS)*, *4*(2), 1-14.

Buitelaar, P., Cimiano, P., & Magnini, B. (2005). Ontology learning from text: An overview. *Ontology learning from text: Methods, evaluation and applications*, *123*.

Chaudhri, V., Baru, C., Chittar, N., Dong, X., Genesereth, M., Hendler, J., ... & Wang, K. (2022). Knowledge Graphs: Introduction, History and, Perspectives. *AI Magazine*, *43*(1), 17-29.

Chollet,F. Character-level recurrent sequence-to-sequence model , 29/09/2017, Διαθέσιμο: https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html (20/7/23)

Chowdhary, K. R. (2020). *Fundamentals of artificial intelligence* (pp. 603-49). New Delhi:: Springer India.

De Marneffe, M. C., Manning, C. D., Nivre, J., & Zeman, D. (2021). Universal dependencies. *Computational linguistics*, *47*(2), 255-308.

Ehrlinger, L., & Wöß, W. (2016). Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, *48*(1-4), 2

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179-211.

Fellbaum, Christiane (2005). WordNet and wordnets. In: Brown, Keith et al. (eds.), Encyclopedia of Language and Linguistics, Second Edition, Oxford: Elsevier, 665-670.

Fensel, D., Şimşek, U., Angele, K., Huaman, E., Kärle, E., Panasiuk, O., ... & Wahler, A. (2020). *Knowledge graphs*. Springer International Publishing.

Jakus, G., Milutinović, V., Omerović, S., Tomažič, S. (2013). Knowledge Representation. In: Concepts, Ontologies, and Knowledge Representation. SpringerBriefs in Computer Science. Springer, New York, NY.

Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, *31*(3), 685-695.

Jurafsky, D. & Martin, J. H.,(2023) , An Introduction to Natural language Processing, Computational Linguistics, and Speech Recognition (https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf)

Mahesh, B. (2020). Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, *9*(1), 381-386.

Maniraj, V., & Sivakumar, R. (2010). Ontology languages-a review. *International Journal of Computer Theory and Engineering*, *2*(6), 887.

Opdahl, A. L. (2020). Knowledge Graphs and Natural-Languag e Processing. In *Big Data in Emergency Management: Exploitation Techniques for Social and Mobile Data* (pp. 75-91). Springer, Cham.

Paulheim, H. (2017). Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, *8*(3), 489-508.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton and Christopher D. Manning. 2020. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In Association for Computational Linguistics (ACL) System Demonstrations. 2020.

Tarwani, K. M., & Edem, S. (2017). Survey on recurrent neural network in natural language processing. *Int. J. Eng. Trends Technol*, *48*(6), 301-304.

Tiwari, S., Al-Aswadi, F. N., & Gaurav, D. (2021). Recent trends in knowledge graphs: theory and practice. *Soft Computing*, *25*(13), 8337-8355.

Zhang, C., & Lu, Y. (2021). Study on artificial intelligence: The state of the art and future prospects. *Journal of Industrial Information Integration*, *23*, 100224.

Ελληνική Βιβλιογραφία

Γεωργούλη, Κ.(2015). Τεχνητή Νοημοσύνη

Καμπουρλάζος, Β., & Παπακώστας, Γ. (2015). Εισαγωγή στην Υπολογιστική Νοημοσύνη. *Αθήνα: ΣΕΑΒ*.

Παναγιωτακόπουλος, Χ., Τσαλίδης, Χ., Γάκης, Π., & Κόκκινος, Θ. (2022). Υπολογιστική γλωσσολογία.

Σακελλαρίου, Η., Βασιλειάδης, Ν., Κεφαλάς, Π., & Σταμάτης, Δ. (2015). Τεχνικές Λογικού Προγραμματισμού για Επίλυση Προβλημάτων.

Παράρτημα Α

Κώδικας Προγράμματος

A. Pyhton File “Main”

```
import stanza
from neo4j import GraphDatabase
graphdb= GraphDatabase.driver(uri="XXXX ", auth=("XXX", "XXXX"))
session = graphdb.session()


# Adding data to the graph .....................................................


#s1 = session.run("CREATE (p:Paragraph {number:'2'})")
#s2 = session.run("MATCH(t:Text{title:'painting'}), (p:Paragraph{number:'2'}) CREATE(t)-
[r:TEXT_HAS_PARAGRAPH{paranum:'2'}]->(p)")
#s3 = session.run("WITH ['1','2','3','4','5','6','7','8'] AS numbers FOREACH (value IN numbers |
CREATE (:Sentence{number:value, paragr:'2'})) ")
#s4 = session.run("MATCH (k:Word{w:'I'}), (s:Sentence{number:'1', paragr:'2'}) CREATE (s)-
[r:IC_HAS_SUBJECT{sentence:'1', paragr:'2'}]->(k)")
#s5 = session.run("MATCH (p:Paragraph{number:'2'}),(s:Sentence{paragr:'2'}) CREATE (p)-
[r:PARAGRAPH_HAS_SENTENCE]->(s)")
#s6 = session.run (" MATCH(w:Word), (s:Sentence) WHERE s.paragr = '2' AND s.number= '1'
AND  w.w= 'read' CREATE (s)-[r:IC_HAS_VERB{sentence: '1',tense: 'simple present',
number:'singular', person: 'first'}]->(w)")
#s7 = session.run ("MATCH(w:Word), (s:Sentence) WHERE w.w = 'book' AND s.paragr = '2'
AND  s.number='1' CREATE(s)-[r:IC_HAS_OBJECT{sentence:'1', number:'plural'}]->(w)")
#s8 = session.run ("MATCH (k:Word{w:'book'}), (s:Sentence{number:'2', paragr:'2'}) CREATE
(s)-[r:IC_HAS_SUBJECT{sentence:'2', paragr:'2', number:'plural'}]->(k)")
#s9 = session.run("MATCH (s:Sentence{number:'2', paragr:'2'}) CREATE
(k:Word{w:'inspire',simplepresent1st:'inspire', simplepresent3rd:' inspires',past:'inspired',
continuous:'inspiring', pastparticiple:'inspired',presentparticiple:'inspiring'})<-
[r:IC_HAS_VERB{sentence: '2',tense: 'simple present', number:'plural', person: 'third'}]-(s)")
#s10 = session.run ("MATCH (s:Sentence{number:'2', paragr:'2'}) CREATE
(k:Word{Case:'Acc',Number:'Sing', Person:'1',PronType:'Prs', w:'me',upos:'PRON'})<-
[r:IC_HAS_OBJECT{sentence:'2'}]-(s)")
#s11 = session.run("MATCH (s:Sentence{number:'3', paragr:'2'}) CREATE
(k:Word{upos:'NOUN', w:'story'}) <-[r:IC_HAS_SUBJECT{sentence:'3', number:'plural'}]-(s)")
```

```
#s12 = session.run("MATCH (s:Sentence{number:'3', paragr:'2'}) CREATE
(k:Word{w:'provoke',simplepresent1st:'provoke', simplepresent3rd:' provokes',past:'provoked',
continuous:'provoking', pastparticiple:'provoked',presentparticiple:'provoking'})<-
[r:IC_HAS_VERB{sentence: '3',tense: 'present continuous', number:'plural', person: 'third'}]-(s)")
#s13 = session.run("MATCH (s:Sentence{number:'3', paragr:'2'}), (k:Word{w:'emotion'})
CREATE (s)-[r:IC_HAS_OBJECT{sentence:'3', number:'plural'}]->(k)")
#s14 = session.run("MATCH (s:Sentence{number:'3', paragr:'2'}) CREATE
(k:Word{upos:'NOUN', w:'author'}) <-[:IC_HAS_SUBJECT{sentence:'2', paragr:'2',
number:'plural'}] - (s) - [:IC_HAS_OBJECT{sentence:'3', number:'sing'}] ->
(g:Word{upos:'NOUN', w:'society'})")
#s15 = session.run("MATCH (k:Word{w:'author'}) <- [r]- (s:Sentence{number:'3', paragr:'2'}) -
[r1] -> (g:Word{w:'society'}) DELETE r, r1 ")
#s16 = session.run("MATCH (k:Word{w:'author'}),(s:Sentence{number:'4', paragr:'2'}) ,
(g:Word{w:'society'}) CREATE (k) <-[:IC_HAS_SUBJECT{sentence:'4', paragr:'2',
number:'plural'}]-(s)-[:IC_HAS_OBJECT{sentence:'4',paragr:'2', number:'sing'}] -> (g)")
#s17 = session.run("MATCH (s:Sentence{number:'4', paragr:'2'}) CREATE
(k:Word{w:'challenge',simplepresent1st:'challenge',
simplepresent3rd:'challenges',past:'challenged', continuous:'challenging',
pastparticiple:'challenged',presentparticiple:'challenging'}) <-[:IC_HAS_VERB{sentence:
'4',tense: 'present continuous', number:'plural', person: 'third'}]-(s)")
#s18 = session.run ("MATCH (s:Sentence{number:'5', paragr:'2'}) CREATE
(k:Word{upos:'NOUN', w:'character'}) <-[:IC_HAS_SUBJECT{sentence:'5', paragr:'2',
number:'plural'}] - (s) - [:IC_HAS_OBJECT{sentence:'5', paragr:'2', number:'sing'}] ->
(g:Word{upos:'NOUN', w:'life'})")
#s19 = session.run("MATCH (s:Sentence{number:'5', paragr:'2'}) CREATE
(k:Word{w:'defy',simplepresent1st:'defy', simplepresent3rd:' defies',past:'defied',
continuous:'defying', pastparticiple:'defied',presentparticiple:'defying'}) <-
[:IC_HAS_VERB{sentence:'5',tense: 'present continuous', number:'plural', person: 'third'}]-(s)")
#s20 = session.run("MATCH (k:Word{w:'I'}), (s:Sentence{number:'6', paragr:'2'}) CREATE (k)
<-[:IC_HAS_SUBJECT{sentence:'6', paragr:'2', number:'singular'}] - (s) -
[:IC_HAS_OBJECT{sentence:'6', paragr:'2', number:'sing'}] -> (g:Word{upos:'NOUN',
w:'comfort'}) ")
#s21 = session.run("MATCH (s:Sentence{number:'6', paragr:'2'}) CREATE
(k:Word{w:'feel',simplepresent1st:'feel', simplepresent3rd:' feels',past:'felt', continuous:'feeling',
pastparticiple:'felt',presentparticiple:'feeling'}) <- [:IC_HAS_VERB{sentence:'6',tense: 'simple
present', number:'singular', person:'first'}]-(s)")
#s22 = session.run("MATCH (k:Word{w:'life'}), (s:Sentence{number:'7', paragr:'2'}) CREATE
(k)<-[:IC_HAS_SUBJECT{sentence:'7', paragr:'2', number:'singular'}] - (s)-
[:IC_HAS_OBJECT{sentence:'7', paragr:'2', number:'singular'}] -> (g:Word{upos:'ADJ',
w:'magical'})")
```

```
#s23 = session.run("MATCH (k:Word{w:'be'}), (s:Sentence{number:'7', paragr:'2'}) CREATE (s)-[:IC_HAS_VERB{sentence:'7',tense: 'simple present', numb

# NLP .....
s = "Clouds float the sky"
nlp = stanza.Pipeline(lang='en', processors='tokenize,mwt,pos')
doc = nlp(s)
print(*[f'word: {word.text}\tupos: {word.upos}\txpos: {word.xpos}\tfeats: {word.feats if word.feats else "_"}' for sent in doc.sentences for word in sent.words], sep='\n')
#dicts = doc.to_dict() # dicts is List[List[Dict]], representing each token / word in each sentence in the document
# .....

# NLP Contsituency parsing .....

nlp = stanza.Pipeline(lang='en', processors='tokenize,mwt,pos,lemma,depparse')
doc = nlp(s)
print(*[f'id: {word.id}\tword: {word.text}\thead id: {word.head}\thead: {sent.words[word.head-1].text if word.head > 0 else "root"}\tdeprel: {word.deprel}' for sent in doc.sentences for word in sent.words], sep='\n')
# .....
```

B. Pyhton File “Sentence”

```
import stanza
import pandas as pd
from pandas import DataFrame
from neo4j import GraphDatabase
import numpy as np
graphdb = GraphDatabase.driver(uri="XXXX", auth=("XXX", "XXX"))

def run_cypher_query(query, parameters=None):
  with graphdb.session() as session:
      result = session.run(query, parameters)
      return result.data()

def allsent():  #returns all sentences
```

```
    COMMAND="MATCH (p:Paragraph)-[g]-(s:Sentence)-[r]->(k) RETURN p.number as paragraph, s.number as snumber, k.w as word, type(r) as relation, r.tense as tense, r.person as person, r.number as wnumber, r.det as det,k.wp as number"
    s = run_cypher_query(COMMAND)
    df = pd.DataFrame(s)
    return df


def senten(df, p, s):  #returns specific sentence
    df = df[(df["snumber"]==s) & (df["paragraph"]==p)].copy()
    return df
    # b = str(n)
    # g = (s[s["snumber"] == b])
    #return g


def allwords():
    COMMAND = "MATCH (w:Word) RETURN w"
    s = run_cypher_query(COMMAND)
    df = pd.DataFrame(s)
    df.set_option('display.max_columns', None)
    df.set_option('display.max_colwidth', None)
    allw = df.json_normalize(df['w'])
    return allw


def depenpar(s):
    s = s.to_string()
    nlp = stanza.Pipeline(lang='en', processors='tokenize,mwt,pos,lemma,depparse')
    doc = nlp(s)
    print(*[f'id: {word.id}\tword: {word.text}\thead id: {word.head}\thead: {sent.words[word.head-1].text if word.head > 0 else "root"}\tdeprel: {word.deprel}' for sent in doc.sentences for word in sent.words], sep='\n')


def word(b):
    word_param = b
    word_param = ', '.join(word_param).replace('[', '').replace(']', '')
    COMMAND = f"MATCH (k:Word {{w:'{word_param}'}}) RETURN k"
    s = run_cypher_query(COMMAND)
    df = pd.DataFrame(s)
```

```
    df = pd.json_normalize(df['k'])
    print(df)
    return df

def wordsent(s,r):  # s: data /frame, v: IC_HAS_VERB or IC_HAS_OBJECT etc
g = (s.loc[s['relation'] == r])
ww = (g.word).values
return ww   #returns the word , meaning the verb, object

def wtense(s, r):
    g = (s.loc[s['relation'] == r])
    #w = (g.relation).values
    if not (g.tense).values == None:
        return (g.tense).values   #returns tense

def wperson(s, r):
    g = (s.loc[s['relation'] == r])
    #w = (g.relation).values
    if (g.tense).values:
        return (g.person).values   #returns tense

def righttense(t, p, df):

        t_str = ''.join(t).replace(' ', '')
        p_str = ''.join(p).replace(' ', '')

        combined_variable = f"{t_str}{p_str}".lower()  # Combine the variables t and p into one
string and convert to lowercase

        # First, check for exact matches in DataFrame columns
        exact_matches = [col for col in df.columns if col.lower() == combined_variable]

        if exact_matches:
            # If an exact match is found, use it to retrieve the value from the DataFrame
            exact_match_column = exact_matches[0]
            value = df[exact_match_column].iloc[0]
```

```
            return value
        else:
            # If no exact match is found, search for substring matches in DataFrame columns
            substring_matches = [col for col in df.columns if col.lower() in combined_variable]

            if substring_matches:
                # If a substring match is found, use the first one to retrieve the value from the
DataFrame
                substring_match_column = substring_matches[0]
                value = df[substring_match_column].iloc[0]
                return value
            else:
                # If no exact or substring match is found, return None
                return None


def rightnu (df,r):
    g = (df.loc[df['relation'] == r ])
    wn = (g.wnumber).values
    if (wn == "plural"):
      return (g.number).values
    else:
      return (g.word).values

def det (df,r):
 g = (df.loc[df['relation'] == r])
 d = (g.det).values
 if not d:
     return ""
 else:
    d= ' '.join(d).strip()
    return d


def wordsentenv (df,r):
    u = wordsent(df, r)  # specific word, example the verb of the specific sentence
```

```
    print(u)
    u1 = np.any(u)
    if not u1 == False:
      t = wtense(df, r)  # tense of word
      print(t)
      p = wperson(df, r)  # in what person is the verb etc
      print(p)
      specword = word(u)  # dataframe with all the properties of a specific word
      verb = righttense(t, p, specword)
      return verb
    else:
        l = ""
        return l

def wordsentenw (df,r):
    u = rightnu(df, r)
    u = ' '.join(u).strip()
    return u




if __name__ == "__main__":

#----------------RELATIONS----------------------------

    robj = "IC_HAS_OBJECT"
    rsubj= "IC_HAS_SUBJECT"
    rverb = "IC_HAS_VERB"
    rmodverb = "IC_HAS_MODAL_VERB"

#-----------------------------------------------------
o = senten(allsent(), p="1", s="6") #specific sentence

#----------------VERB---------------------------------
```

```
verb = wordsentenv(o,rverb)

#---------------MODAL VERB-----------------------------

modal_verb = wordsentenv(o,rmodverb)

#---------------SUBJECT--------------------------------

subject = wordsentenw(o,rsubj)

#----------------OBJECT---------------------------------

object = wordsentenw(o,robj)

#---------------DETERMINER-----------------------------

detero = det(o, robj )
deters = det(o, rsubj )

#---------------Sentence-------------------------------

print(deters,subject,modal_verb,verb,detero,object)
```