

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής



Εργασία Μαθήματος
Επεξεργασία Φυσικής Γλώσσας

<i>Αριθμός εργασίας – Τίτλος εργασίας</i>	<i>Τελική Εργασία</i>
Όνομα φοιτητή	Κωνσταντίνος Καλογερόπουλος
Αρ. Μητρώου	Π19057
Ημερομηνία παράδοσης	31 Ιουλίου, 2022



Εκφώνηση εργασίας

Β.2 Λύση σε άλλη γλώσσα προγραμματισμού

Εναλλακτικά μπορείτε να χρησιμοποιήσετε άλλες γλώσσες προγραμματισμού ή libraries τον κώδικα και την λειτουργικότητα των οποίων πρέπει να τεκμηριώσετε πειστικά και κατανοητά σύμφωνα με τα παρακάτω :

Θέμα 1^ο (20 μονάδες)

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

Θέμα 2^ο (20 μονάδες)

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

Θέμα 3^ο (30 μονάδες)

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

Θέμα 4^ο (30 μονάδες)

Πρόγραμμα στην γλώσσα προγραμματισμού της επιλογής σας, για την ενημέρωση και πραγματοποίηση ερωταποκρίσεων σε Βάση Γνώσης που έχετε αναπτύξει για αυτό τον σκοπό. Οι ερωτήσεις και απαντήσεις θα δίδονται σε φυσική γλώσσα.



ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1	Θέμα 1	4
1.1	Εισαγωγή.....	4
1.2	Περιγραφή Προγράμματος	4
1.3	Εκτέλεση Προγράμματος	5
2	Θέμα 2	6
2.1	Εισαγωγή.....	6
2.2	Περιγραφή Προγράμματος	6
2.3	Εκτέλεση Προγράμματος	8
3	Θέμα 3	9
3.1	Εισαγωγή.....	9
3.2	Περιγραφή Προγράμματος	9
3.3	Εκτέλεση Προγράμματος	10
4	Θέμα 4	11
4.1	Εισαγωγή.....	11
4.2	Περιγραφή Προγράμματος	11
4.3	Εκτέλεση Προγράμματος	13
5	Βιβλιογραφικές Πηγές.....	14

1 Θέμα 1

1.1 Εισαγωγή

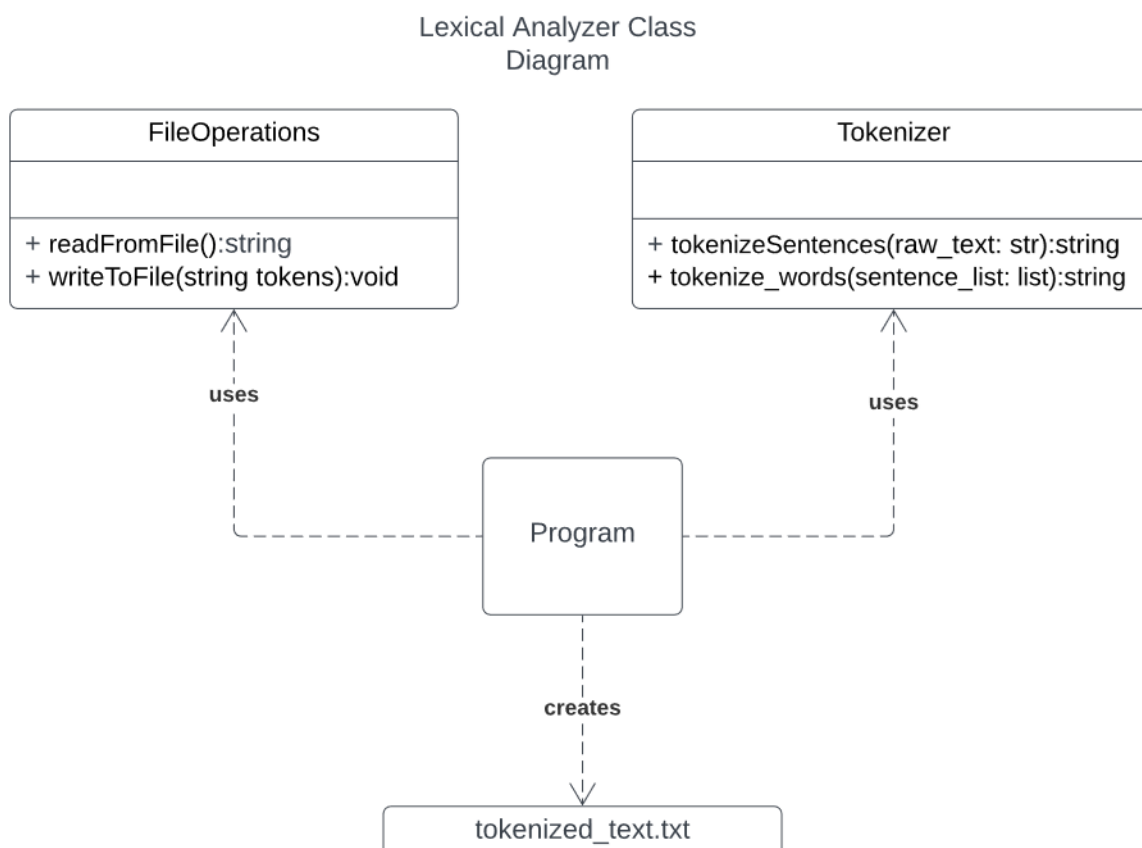
Η λεκτική ανάλυση (*lexical analysis*) είναι η διαδικασία που μετατρέπει μια ακολουθία από χαρακτήρες σε μια ακολουθία από λεκτικές μονάδες (tokens). Ένα πρόγραμμα ή συνάρτηση που κάνει λεκτική ανάλυση ονομάζεται λεκτικός αναλυτής (*lexical analyzer*, *lexer* ή *scanner*).

Σκοπός του πρώτου θέματος είναι η δημιουργία ενός λεκτικού αναλυτή που διαβάζει μια μικρή ιστορία και παράγει μια λίστα από προτάσεις όπου κάθε πρόταση περιέχει μια λίστα από λέξεις.

1.2 Περιγραφή Προγράμματος

Το πρόγραμμα αναπτύχθηκε σε python3.10.5. Ακόμα, έγινε χρήση της βιβλιοθήκης NLTK (Natural Language Toolkit). Το Εργαλείο Φυσικής Γλώσσας, ή πιο συχνά NLTK, είναι μια σειρά από βιβλιοθήκες και προγράμματα για συμβολική και στατιστική επεξεργασία φυσικής γλώσσας για τα Αγγλικά.

Το πρόγραμμα αποτελείται από δύο κλάσεις όπως φαίνεται από το παρακάτω σχεδιάγραμμα:



Εικόνα 1 Lexical Analyzer Class Diagram



Αρχικά, η κλάση `FileOperations`, περιέχει τις μεθόδους για το άνοιγμα και την αποθήκευση ενός αρχείου. Πιο συγκεκριμένα:

- `readFromFile():string`

Η μέθοδος `readFromFile()` καλείται από το πρόγραμμα για να ανοίξει και να διαβάσει το αρχείο. Επιστρέφει μια μεταβλητή τύπου `string`, που έχει το περιεχόμενο του αρχείου.

- `writeToFile(tokens):void`

Η μέθοδος `writeToFile(text)` δημιουργεί ένα αρχείο `tokenized_text.txt`, μέσα στο οποίο αποθηκεύει τις μονάδες (`tokens`) που προέκυψαν από την λειτουργία του λεκτικού αναλυτή.

Έπειτα, υπάρχει η κλάση `Tokenizer`, η οποία ενσωματώνει τις μεθόδους της βιβλιοθήκης `NLTK`, για τους σκοπούς της δημιουργίας μιας λίστας που περιέχει λέξεις και μιας λίστας που περιέχει προτάσεις. Πιο αναλυτικά:

- `tokenizeSentences(raw_text: str):string`

Η μέθοδος `tokenizeSentences(raw_text: str)` χωρίζει το κείμενο σε προτάσεις, τις οποίες επιστρέφει σε μία λίστα.

- `tokenize_words(sentence_list: list):string`

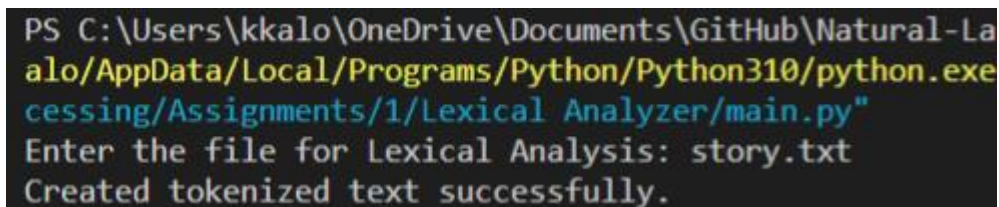
Η μέθοδος `tokenize_words(sentence_list: list)` λαμβάνει μια λίστα προτάσεων και με βάση αυτή, δημιουργεί μια λίστα λέξεων για κάθε πρόταση, την οποία και επιστρέφει.

1.3 Εκτέλεση Προγράμματος

Το πρόγραμμα βρίσκεται στον φάκελο `/Assignments/1/Lexical Analyzer/`.

Οδηγίες εκτέλεσης:

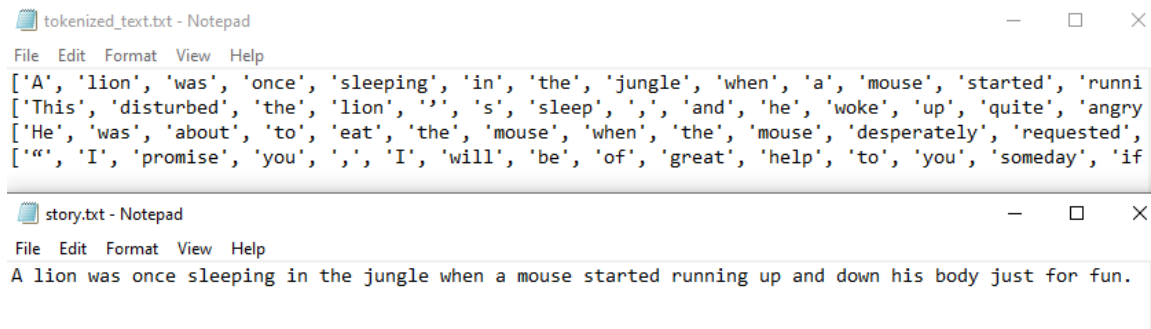
Αρχικά, πρέπει να έχει γίνει `install` η βιβλιοθήκη `NLTK` (`pip install nltk`). Έπειτα, επιλέγουμε το αρχείο `main.py` και το εκτελούμε με κάποια `python version` της επιλογής μας (καλύτερα `python version 10+`).



```
PS C:\Users\khalo\OneDrive\Documents\GitHub\Natural-Language-Processing\Assignments\1\Lexical Analyzer/
C:\Users\khalo\AppData\Local\Programs\Python\Python310\python.exe
Enter the file for Lexical Analysis: story.txt
Created tokenized text successfully.
```

Εικόνα 2 Output προγράμματος

Το πρόγραμμα ζητάει να του δώσουμε ένα αρχείο `txt` το οποίο περιέχει μια μικρή ιστορία και στην συνέχεια πραγματοποιεί την λεκτική ανάλυση. Το τελικό αποτέλεσμα το αποθηκεύει σε ένα αρχείο `txt` με όνομα `tokenized_text`. Παρακάτω ακολουθεί μια εικόνα που αποτελεί ένα δείγμα του τελικού αρχείου που παράγεται σε σύγκριση με το αρχικό.



Εικόνα 3 Output αρχικής(κάτω) και τελικής(πάνω) ιστορίας μετά την λεκτική ανάλυση.

2 Θέμα 2

2.1 Εισαγωγή

Ο συντακτικός αναλυτής υλοποιεί τη συντακτική ανάλυση μιας συγκεκριμένης γλώσσας. Αρχικά, δέχεται ως είσοδο ένα πρόγραμμα με τη μορφή ακολουθίας λεκτικών μονάδων. Έπειτα, ελέγχει αν το πρόγραμμα είναι σύμφωνο με τη γραμματική της γλώσσας που υλοποιεί (αν ανήκει στη συγκεκριμένη γλώσσα). Σε περίπτωση συντακτικού λάθους ενημερώνει τον χρήστη. Τέλος, παράγει το συντακτικό δέντρο που αντιστοιχεί στην ακολουθία εισόδου.

Σκοπός του δεύτερου θέματος είναι η ανάπτυξη ενός συντακτικού αναλυτή, ο οποίος με βάση τους κανόνες συντακτικής ανάλυσης της πρότυπης λύσης σε Prolog, θα παράγει το συντακτικό δέντρο της πρότασης.

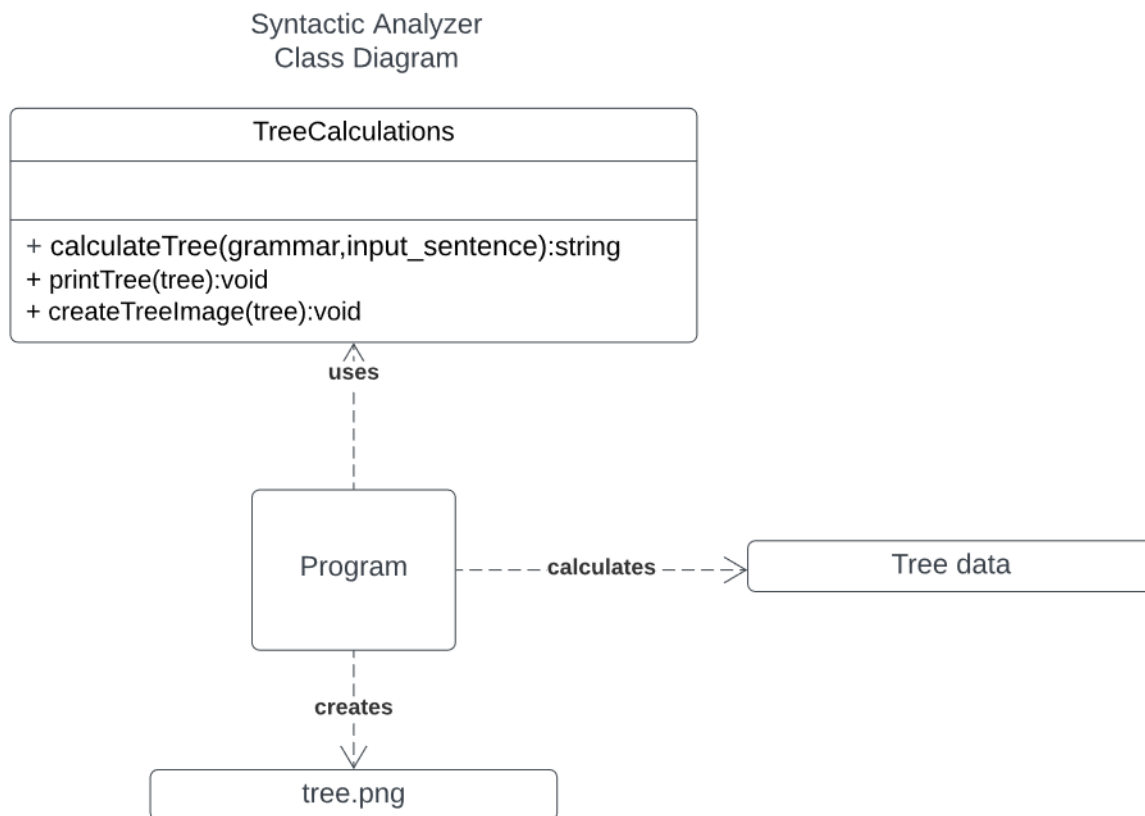
2.2 Περιγραφή Προγράμματος

Το πρόγραμμα αναπτύχθηκε σε python3.10.5. Ακόμα, έγινε χρήση της βιβλιοθήκης NLTK (Natural Language Toolkit) (Αναφέρθηκε στο 1^ο θέμα πιο αναλυτικά).

Επιπλέον, έγινε χρήση της βιβλιοθήκης Pillow. Η Pillow (Python Imaging Library) είναι μια δωρεάν βιβλιοθήκη ανοιχτής πηγής για τη γλώσσα προγραμματισμού Python που προσθέτει υποστήριξη για το άνοιγμα, το χειρισμό και την αποθήκευση πολλών διαφορετικών μορφών αρχείων εικόνας.

Επίσης, έγινε χρήση του διερμηνέα (interpreter) Ghostscript. Πιο συγκεκριμένα, χρησιμοποιείται για τις γλώσσες περιγραφής σελίδας PostScript και Portable Document Format (PDF) της Adobe Systems. Πιο αναλυτικά βήματα για την εγκατάσταση του, θα αναφερθούν στο output του προγράμματος.

Το πρόγραμμα αποτελείται από μία κλάση όπως φαίνεται από το παρακάτω σχεδιάγραμμα:



Εικόνα 4 Syntactic Analyzer Class Diagram

Η κλάση `TreeCalculations` λαμβάνει τα δεδομένα για το δέντρο και ενσωματώνει τις κατάλληλες λειτουργίες για την δημιουργία του. Πιο συγκεκριμένα, έχει τις ακόλουθες μεθόδους:

- `calculateTree(grammar,input_sentence):string`

Λαμβάνει ως παραμέτρους τα δεδομένα της γραμματικής (μέσω του αρχείου `grammar.cfg`) και μια πρόταση προς έλεγχο. Επιστρέφει το παραγόμενο δέντρο σε μια `string` μεταβλητή.

- `printTree(tree):void`

Εκτυπώνει το δέντρο της πρότασης στην κονσόλα του προγράμματος.

- `createTreeImage(tree):void`

Σε αυτή την μέθοδο γίνεται χρήση των βιβλιοθηκών που αναφέρθηκαν στην αρχή της περιγραφής του προγράμματος. Πιο συγκεκριμένα, παράγει το δέντρο της πρότασης σε εικόνα της μορφής `PostScript(.ps)`. Έπειτα την μετατρέπει σε μορφή `.png` ώστε να είναι εύκολα προσβάσιμη για άνοιγμα.

2.3 Εκτέλεση Προγράμματος

Το πρόγραμμα βρίσκεται στον φάκελο /Assignments/2/Syntactic Analyzer/.

Οδηγίες εκτέλεσης:

Αρχικά, πρέπει να έχει γίνει install η βιβλιοθήκη NLTK (pip install nltk) και Pillow (Python Imaging Library) (pip install pillow). Έπειτα, πρέπει να γίνει εγκατάσταση[2] του διερμηνέα (interpreter) Ghostscript και να ρυθμιστεί το path του μέσα στην μέθοδο createImageTree() (συγκεκριμένα στο EpsImagePlugin.gs_windows_binary). Τέλος, επιλέγουμε το αρχείο main.py και το εκτελούμε.

```
PS C:\Users\kkalo\OneDrive\Documents\GitHub\Natural-Language-Processing\As
kkalo/AppData/Local/Programs/Python/Python310/python.exe "c:/Users/kkalo/O
rocessing/Assignments/2/Syntactic Analyzer/main.py"
Syntactic Tree is:
(s (np (det the) (n dog)) (vp (v chased) (np (det the) (n cat))))

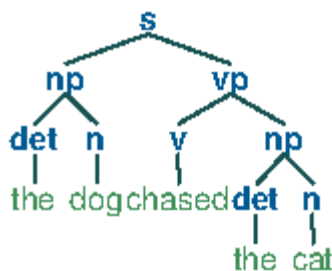
Creating syntactic tree image file..

Converting file to PNG so we can view it..

Done! Locate the file as 'tree.png' in the working directory.
```

Εικόνα 5 Output προγράμματος

Παρατηρείται ότι πρώτα πρώτα το πρόγραμμα παράγει και εκτυπώνει το συντακτικό δέντρο με βάση τη γραμματική και την πρόταση που του δόθηκε. Στην συνέχεια δημιουργεί το δέντρο σε εικόνα μορφής PostScript(.ps) και τέλος το μετατρέπει σε εικόνα png όπου ακολουθεί παρακάτω:



Εικόνα 6 Συντακτικό δέντρο της δοσμένης πρότασης.

3 Θέμα 3

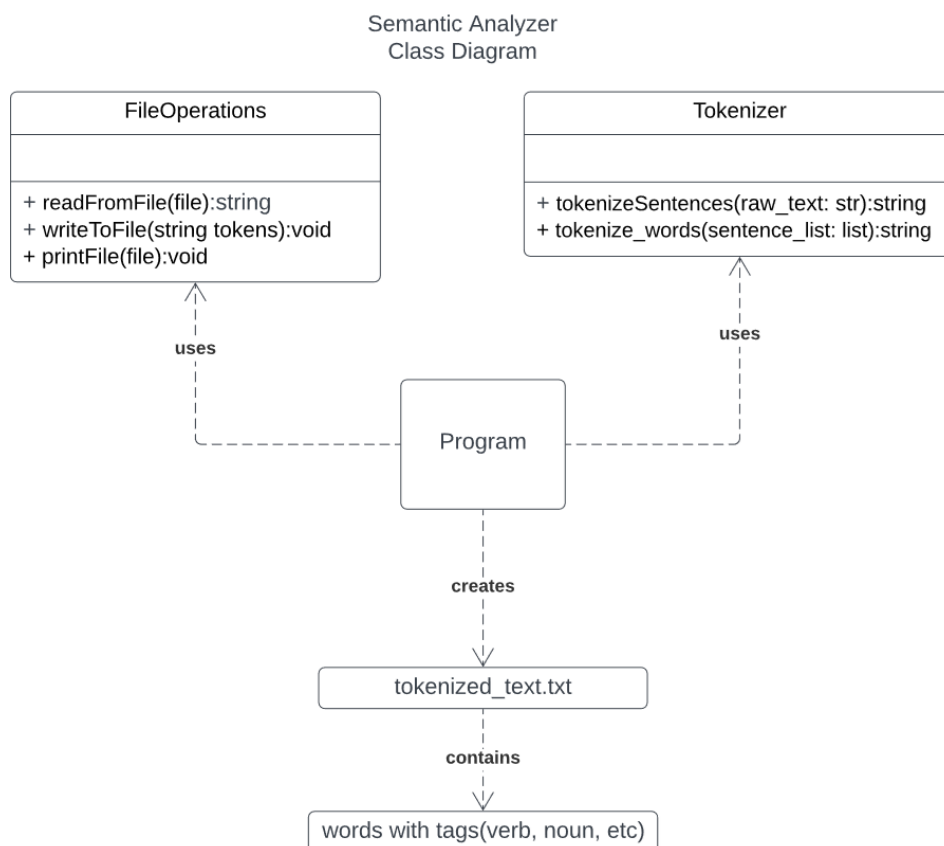
3.1 Εισαγωγή

Με απλά λόγια, η σημασιολογική ανάλυση είναι η διαδικασία εξαγωγής νοήματος από το κείμενο. Επιτρέπει στους υπολογιστές να κατανοούν και να ερμηνεύουν προτάσεις, παραγράφους ή ολόκληρα έγγραφα, αναλύοντας τη γραμματική τους δομή και προσδιορίζοντας τις σχέσεις μεταξύ μεμονωμένων λέξεων σε ένα συγκεκριμένο πλαίσιο (όπως σε κάποια πρόταση). Χρησιμοποιείται συχνά στην αγορά για την εξαγωγή συγκεντρωτικών δεδομένων με βάση τις κριτικές των καταναλωτών για κάποιο προϊόν ή υπηρεσία.

Σκοπός του τρίτου θέματος είναι η ανάπτυξη ενός σημασιολογικού αναλυτή, ο οποίος με βάση τους κανόνες σημασιολογικής ανάλυσης της πρότυπης λύσης σε Prolog, θα παράγει το σημεινόμενα της πρότασης όπως είναι οι σχέσεις μεταξύ ρημάτων, ουσιαστικών, επιθέτων κ.λπ.

3.2 Περιγραφή Προγράμματος

Το πρόγραμμα αναπτύχθηκε σε python3.10.5. Επίσης, έγινε χρήση της βιβλιοθήκης NLTK (Natural Language Toolkit). Ακολουθεί το διάγραμμα κλάσεων:



Εικόνα 7 Semantic Analyzer Class Diagram



Αρχικά, η κλάση `FileOperations`, περιέχει τις μεθόδους για το άνοιγμα και την αποθήκευση ενός αρχείου. Πιο συγκεκριμένα:

- `readFromFile():string`

Η μέθοδος `readFromFile()` καλείται από το πρόγραμμα για να ανοίξει και να διαβάσει το αρχείο. Επιστρέφει μια μεταβλητή τύπου `string`, που έχει το περιεχόμενο του αρχείου.

- `writeToFile(tokens):void`

Η μέθοδος `writeToFile(text)` δημιουργεί ένα αρχείο `tokenized_text.txt`, μέσα στο οποίο αποθηκεύει τις μονάδες (`tokens`) που προέκυψαν από την λειτουργία του λεκτικού αναλυτή.

- `printFile(file):void`

Η μέθοδος αυτή, τυπώνει τα δεδομένα ενός αρχείου τύπου `text`.

Έπειτα, υπάρχει η κλάση `Tokenizer`, η οποία ενσωματώνει τις μεθόδους της βιβλιοθήκης `NLTK`, για τους σκοπούς της δημιουργίας μιας λίστας που περιέχει λέξεις και μιας λίστας που περιέχει προτάσεις. Πιο αναλυτικά:

- `tokenizeSentences(raw_text: str):string`

Η μέθοδος `tokenizeSentences(raw_text: str)` χωρίζει το κείμενο σε προτάσεις, τις οποίες επιστρέφει σε μία λίστα.

- `tokenize_words(sentence_list: list):string`

Η μέθοδος `tokenize_words(sentence_list: list)` λαμβάνει μια λίστα προτάσεων και με βάση αυτή, δημιουργία μια λίστα λέξεων για κάθε πρόταση, την οποία επεξεργάζεται αναγνωρίζοντας τί είναι η κάθε λέξη (ουσιαστικό, ρήμα, αντικείμενο) και τοποθετεί δίπλα της την ανάλογη ιδιότητα που έχει μέσα στην πρόταση.

3.3 Εκτέλεση Προγράμματος

Το πρόγραμμα βρίσκεται στον φάκελο `/Assignments/3/Semantic Analyzer/`.

Οδηγίες εκτέλεσης:

Αρχικά, πρέπει να έχει γίνει `install` η βιβλιοθήκη `NLTK` (`pip install nltk`). Ακόμα, θα χρειαστεί να εγκατασταθεί η βιβλιοθήκη συμβόλων της `NLTK` (`Nltk.download("universal_tagset")`). Έπειτα, επιλέγουμε το αρχείο `main.py` και το εκτελούμε με κάποια `python version` της επιλογής μας (καλύτερα `python version 10+`).

```
yzers> & C:/Users/kka10/AppData/Local/Programs/Python/Python310/python.exe -c:/Users/k
Documents/GitHub/Natural-Language-Processing/Assignments/3/Semantic Analyzer/main.py"
Enter the file for Semantic Analysis: story.txt
Created tokenized text successfully.
[('A', 'DET'), ('lion', 'NOUN'), ('was', 'VERB'), ('once', 'ADV'), ('sleeping', 'VERB'), ('in', 'ADP'), ('the', 'DET'), ('jungle', 'NOUN'), ('when', 'ADV'), ('a', 'DET'), ('mouse', 'NOUN'), ('started', 'VERB'), ('running', 'VERB'), ('up', 'ADV'), ('and', 'CONJ'), ('down', 'ADP'), ('his', 'PRON'), ('body', 'NOUN'), ('just', 'ADV'), ('for', 'ADP'), ('fun', 'NOUN'), ('.', '.')] ]
```

Εικόνα 8 Output προγράμματος `Semantic Analyzer`

Όταν ξεκινάει η εκτέλεση του προγράμματος, ζητάει από τον χρήστη να εισάγει ένα κείμενο με προτάσεις. Στην συνέχεια, το πρόγραμμα το επεξεργάζεται και παράγει τα σημεινόμενα της κάθε πρότασης. Τέλος, τα αποθηκεύει στο αρχείο `tokenized_text.txt`, το οποίο προβάλλει στο `ouput` του προγράμματος ώστε να δει ο χρήστης το αποτέλεσμα της σημασιολογικής ανάλυσης.

4 Θέμα 4

4.1 Εισαγωγή

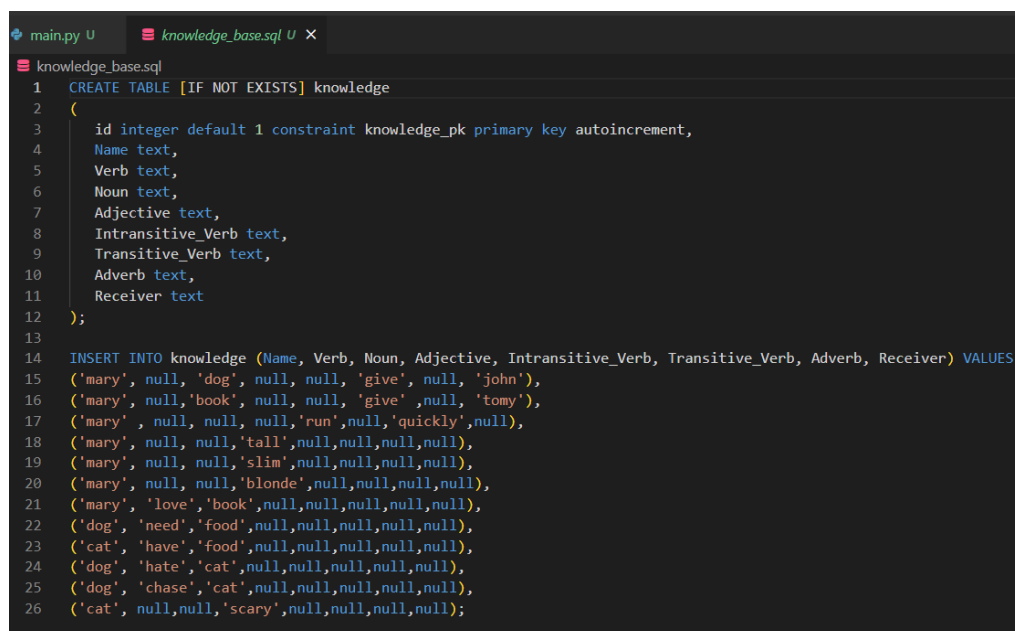
Η βάση γνώσεων είναι ένα ολοκληρωμένο αποθετήριο που περιέχει τις πληροφορίες που απαιτούνται για την κατανόηση των ερωτήσεων και την απάντηση τους. Ακόμα δίνει τη δυνατότητα να βελτιστοποιηθεί η επεξεργασία φυσικής γλώσσας και η απάντηση ερωτήσεων για να επιτύχουμε ακρίβεια που μοιάζει όσο τον δυνατό πιο φιλική με τον άνθρωπο.

Σκοπός του τέταρτου θέματος είναι η ανάπτυξη ενός προγράμματος για την πραγματοποίηση ερωταποκρίσεων σε μία βάση γνώσης. Οι ερωτήσεις και οι απαντήσεις θα δίνονται σε φυσική γλώσσα.

4.2 Περιγραφή Προγράμματος

Το πρόγραμμα αναπτύχθηκε σε `python3.10.5`.

Η βάση γνώσης στην `python` αντικατοπτρίζεται σε μια βάση δεδομένων. Για αυτό, έγινε χρήση της `SQLite`. Έτσι, εκτελούμε τις παρακάτω εντολές στο `query tool` για να δημιουργηθεί ο πίνακας στην βάση και οι στήλες του.



```
main.py U knowledge_base.sql X
knowledge_base.sql
1 CREATE TABLE [IF NOT EXISTS] knowledge
2 (
3   id integer default 1 constraint knowledge_pk primary key autoincrement,
4   Name text,
5   Verb text,
6   Noun text,
7   Adjective text,
8   Intransitive_Verb text,
9   Transitive_Verb text,
10  Adverb text,
11  Receiver text
12 );
13
14 INSERT INTO knowledge (Name, Verb, Noun, Adjective, Intransitive_Verb, Transitive_Verb, Adverb, Receiver) VALUES
15 ('mary', null, 'dog', null, null, 'give', null, 'john'),
16 ('mary', null, 'book', null, null, 'give', null, 'tomy'),
17 ('mary', null, null, null, 'run', null, 'quickly', null),
18 ('mary', null, null, 'tall', null, null, null, null),
19 ('mary', null, null, 'slim', null, null, null, null),
20 ('mary', null, null, 'blonde', null, null, null, null),
21 ('mary', 'love', 'book', null, null, null, null, null),
22 ('dog', 'need', 'food', null, null, null, null, null),
23 ('cat', 'have', 'food', null, null, null, null, null),
24 ('dog', 'hate', 'cat', null, null, null, null, null),
25 ('dog', 'chase', 'cat', null, null, null, null, null),
26 ('cat', null, null, 'scary', null, null, null, null);
```

Εικόνα 9 Τα queries για την δημιουργία του πίνακα και των στηλών του.

Στην συνέχεια, στο αρχείο `semantics.fcfg` έχει δημιουργηθεί μια γραμματική και ένα λεξιλόγιο σύμφωνα με την πρότυπη λύση στην Prolog.

```
main.py U knowledge_base.sql U semantics.fcfg U X
≡ semantics.fcfg
1 % start S
2 S[SEM=('SELECT'+?q + 'FROM knowledge WHERE' + ?p)] -> Q[SEM=?q] P[SEM=?p]
3
4 P[SEM=(?v+ 'AND' '+?n)] -> V[SEM=?v] N[SEM=?n]
5 P[SEM=?v] -> V[SEM=?v]
6 P[SEM=?adj] -> ADJ[SEM=?adj]
7 P[SEM=?iv] -> IV[SEM=?iv]
8 P[SEM=(?iv+ 'AND' '+?adv)] -> IV[SEM=?iv] ADV[SEM=?adv]
9 P[SEM=?tv] -> TV[SEM=?tv]
10 P[SEM=(?tv+ 'AND' '+?n)] -> TV[SEM=?tv] N[SEM=?n]
11 P[SEM=(?tv+ 'AND' '+?r)] -> TV[SEM=?tv] R[SEM=?r]
12
13 Q[SEM='Name'] -> 'who'
14 Q[SEM='Noun'] -> 'what' | 'what is'
15 Q[SEM='Adverb'] -> 'how'
16 Q[SEM='Receiver'] -> 'who is'
17
18 R[SEM=("Receiver='john'")] -> 'to john'
19 R[SEM=("Receiver='mary'")] -> 'to mary'
20 R[SEM=("Receiver='tomy'")] -> 'to tomy'
21
22 N[SEM=("Noun='dog'")] -> 'dog' | 'dogs'
23 N[SEM=("Noun='food'")] -> 'food' | 'foods'
24 N[SEM=("Noun='cat'")] -> 'cat' | 'cats'
25 N[SEM=("Noun='book'")] -> 'book' | 'books'
```

Εικόνα 10 Δείγμα της γραμματικής και του λεξιλογίου που δημιουργήθηκε για τις ανάγκες του προγράμματος.

Έπειτα, ξεκίνησε η δημιουργία του προγράμματος. Έγινε χρήση της βιβλιοθήκης NLTK.

Πιο συγκεκριμένα, η κλάση `DbCreation` περιέχει την μέθοδο:

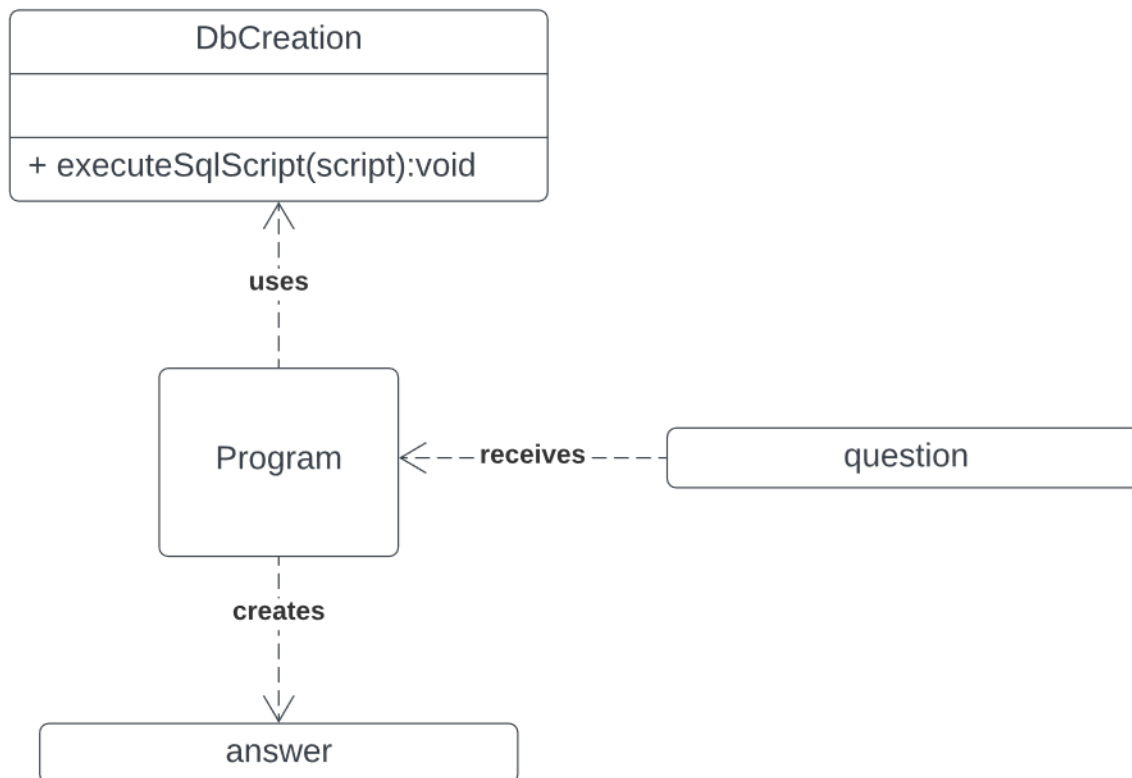
- `executeSqlScript(script):void`

Η μέθοδος αυτή εκτελεί το SQL αρχείο με τις sql εντολές για την αρχικοποίηση του πίνακα `knowledge` στην βάση δεδομένων.

Μετά, το πρόγραμμα χρησιμοποιεί τα δεδομένα στην βάση τα οποία επεξεργάζεται κάθε φορά μετά την ερώτηση του χρήστη σε αυτό. Στο τέλος, του επιστρέφει την απάντηση σε φυσική γλώσσα.

Ακολουθεί το διάγραμμα κλάσεων:

KnowledgeBase Application Class Diagram



Εικόνα 11 Knowledge Base Application Class Diagram

4.3 Εκτέλεση Προγράμματος

Το πρόγραμμα βρίσκεται στον φάκελο `/Assignments/4/ KnowledgeBaseApplication/`.

Οδηγίες εκτέλεσης:

Αρχικά, πρέπει να έχει γίνει install η βιβλιοθήκη NLTK (`pip install nltk`). Έπειτα, επιλέγουμε το αρχείο `main.py` και το εκτελούμε με κάποια `python version` της επιλογής μας (καλύτερα `python version10+`).

Αρχικά, ο χρήστης πραγματοποιεί μια ερώτηση στο πρόγραμμα. Στην συνέχεια, το πρόγραμμα ψάχνει στην βάση γνώσης εάν υπάρχει απάντηση. Στην περίπτωση που υπάρχει, του επιστρέφει το αποτέλεσμα.

Ακολουθούν εικόνες από την εκτέλεση του προγράμματος:



```
Examples of questions you can make to the NLP: 'who loves books', 'who gave dog', 'who hates cat'..
Enter your question: who loves books
mary
mary
mary
mary
mary
mary
mary
mary
mary
mary
mary
mary
mary
mary
mary
mary
mary
mary
```

Εικόνα 12 Output προγράμματος 1

```
Loading grammar..

Examples of questions you can make to the NLP: 'who loves books', 'who gave dog', 'who hates cat'..
Enter your question: who hates cat
dog
dog
dog
dog
dog
dog
dog
dog
dog
dog
dog
dog
dog
dog
dog
dog
dog
dog
dog
dog
```

Εικόνα 13 Output προγράμματος 2

5 Βιβλιογραφικές Πηγές

1. Σημειώσεις διαλέξεων
2. Link to Ghostscript download site, <https://ghostscript.com/releases/gsdnld.html>
3. SQLite, <https://sqliteonline.com/>
4. Βιβλιοθήκη nltk, <https://www.nltk.org/>