Πανεπιστήμιο Πειραιά

Τμήμα Ψηφιακών Συστημάτων

Ανάκτηση Πληροφοριών

Project 2020 - 2021

Παναγιώτης Καραμολέγκος

E17065

Περιεχόμενα

1. Εισαγωγή

1.1. Χαρακτηριστικά της Μηχανής Αναζήτησης

Για το Project του μαθήματος «Ανάκτηση Πληροφοριών» έχει κατασκευαστεί μία μικρή μηχανή αναζήτησης εγγράφων κειμένου (εγγράφων με κατάληξη .txt). Το Project αυτό, επεκτείνει τις εργαστηριακές διαλέξεις του προαναφερθέν μαθήματος και συνεπώς βασίζεται επάνω στον κώδικα τον οποίο παραδόθηκε, ώστε να προσθέσει επάνω σε αυτόν, νέες λειτουργικότητες. Η μηχανή αναζήτησης περιέχει:

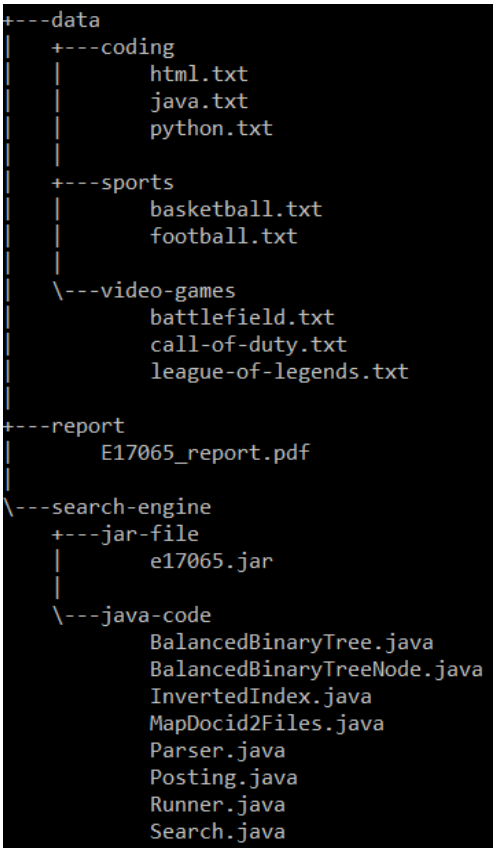
- Parser εγγράφων κειμένων που υλοποιεί διαίρεση σε σύμβολα και παράγει όρους (tokens) τα οποία τοποθετούνται εντός του λεξικού της δομής ευρετηρίου που έχει κατασκευαστή.
- Αντεστραμμένο ευρετήριο ως κατακερματισμένος πίνακας (λεξικό με λίστες καταχώρησης).
- **Αλγόριθμο ανάκτησης εγγράφων που να υποστηρίζει ερωτήματα Boole (and, or, not με περισσότερες από 2 λέξεις).**
- **Query parser που παίρνει ως είσοδο ένα ερώτημα σε εκφράσεις Boole (με ή χωρίς wildcard queries) και το διαχειρίζεται κατευθύνοντας το στην αναγκαία δομή για την επίλυσή του.**
- Εμφάνιση των αποτελεσμάτων της αναζήτησης στην οθόνη (την κονσόλα - τερματικό της οθόνης).
- Κανονικοποίηση των ερωτημάτων και τον λέξεων του λεξικού.
- **Αποδοτική υλοποίηση λεξικού (με χρήση μίας δομής Ισοζυγισμένου Δυαδικού Δέντρου).**
- **Υποστήριξη ανάκτηση ανεκτική σε σφάλματα με την χρήση ερωτημάτων με μπαλαντέρ (wildcard queries).**
- Μηχανισμό αποθήκευσης του ευρετηρίου στον δίσκο (όπως και απόκτησης αυτού, από τον δίσκο).

Το περιεχόμενο στην παραπάνω λίστα το οποίο είναι **μαυρισμένο**, είναι οι επιπλέων λειτουργικότητες που έχουν προστεθεί στον κώδικα των εργαστηριακών μαθημάτων.

1.2. Περιεχόμενα Παραδοτέου

Τα περιεχόμενα του .zip αρχείου το οποίο έχει παραδοθεί είναι αυτά που φαίνονται σε «tree» μορφή στην **Εικόνα 1**. Όπως παρουσιάζεται στην Εικόνα, υπάρχουν τρις φάκελοι οι οποίοι είναι οι εξής:

- data: Φάκελος ο οποίος είναι ένας ενδεικτικό κατάλογος με υποκαταλόγους με .txt αρχεία τα οποία περιέχουν κείμενο. Αυτός ο κατάλογος χρησιμοποιείται στα παραδείγματα του **Κεφαλαίου 2** ώστε να μπορεί να αναπαραχθεί η εκτέλεση ερωτημάτων.
- report: Φάκελος ο οποίος περιέχει την αναφορά την οποία διαβάζεται αυτήν την στιγμή. Αυτή η αναφορά περιέχει, αναλυτικά το τι έχει υλοποιηθεί (τι ακριβώς υποστηρίζει η μηχανή αναζήτησης), οδηγίες για την εκτέλεση του κώδικα, όπως και τεκμηρίωση αυτού του κώδικα.
- search-engine: Φάκελος ο οποίος περιέχει τον κώδικα της μηχανής αναζήτησης (υποφάκελος «java-code»), όπως επίσης και ένα εκτελέσιμο αρχείο .jar ως έναν δεύτερο, πιο μαζεμένο, τρόπο εκτέλεσης (υποφάκελος «jar-file»).



Εικόνα 1 – Tree μορφή του παραδοτέου

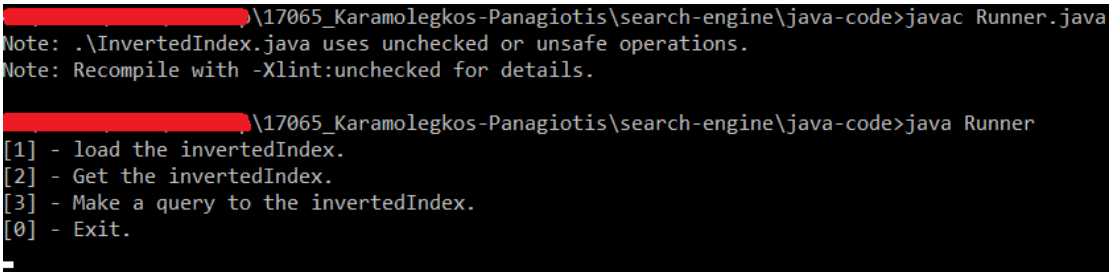
2. Εγχειρίδιο Χρήστη

2.1. Τρόπος Χρήσης της Μηχανής Αναζήτησης

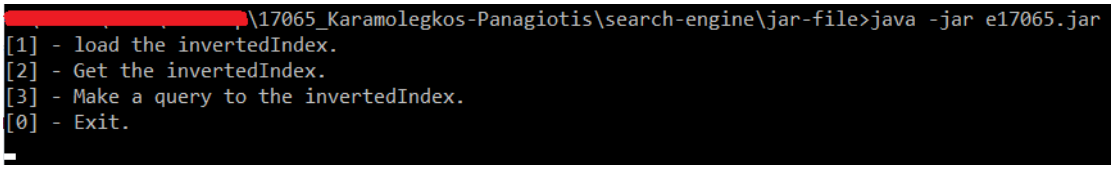
2.1.1. Εκτέλεση της Μηχανής Αναζήτησης

Η Μηχανή Αναζήτησης που έχει παραδοθεί, μπορεί να εκτελεστεί με δύο τρόπους:

- Εκτελώντας τον java κώδικα απευθείας όπως φαίνεται στην **Εικόνα 2**.
- Εκτελώντας το jar αρχείο όπως φαίνεται στην **Εικόνα 3** (Με αυτόν τον τρόπο αποφεύγεται η δημιουργία των .class αρχείων και για αυτό είναι ένας πιο «μαζεμένος» τρόπος εκτέλεσης).



Εικόνα 2 – Εκτέλεση του java κώδικα



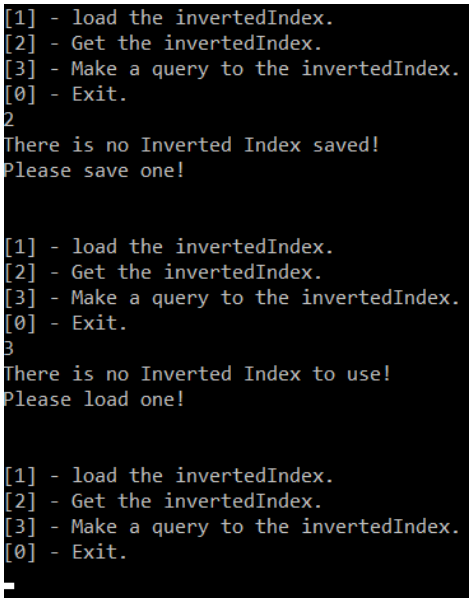
Εικόνα 3 – Εκτέλεση του jar αρχείου.

Είναι σημαντικό να σημειωθεί ότι για την ομαλή λειτουργία της Μηχανής Αναζήτησης ο χρήστης προτείνεται να χρησιμοποιεί **μόνιμα μόνο έναν** από τους προαναφερθέν τρόπους ώστε να μην τύχει να αποθηκεύσει κάποιο ευρετήριο και στην συνέχεια να μην το βρίσκει.

Όπως φαίνεται και από τους δύο τρόπους εκτέλεσης, ο χρήστης καταλήγει σε μία οθόνη, στην οποία έχει 4ης επιλογές:

- Απόκτηση ενός ευρετηρίου μέσο κάποιου path και αποθήκευση αυτού για μελλοντική χρήση – Επιλογή 1.
- Απόκτηση ενός ήδη αποθηκευμένου ευρετηρίου από κάποια προηγούμενη χρήση της κατασκευασμένης εφαρμογής – Επιλογή 2.
- Εκτέλεση κάποιου ερωτήματος στο αντεστραμμένο ευρετήριο – Επιλογή 3.
- Έξοδος από την εφαρμογή – Επιλογή 0.

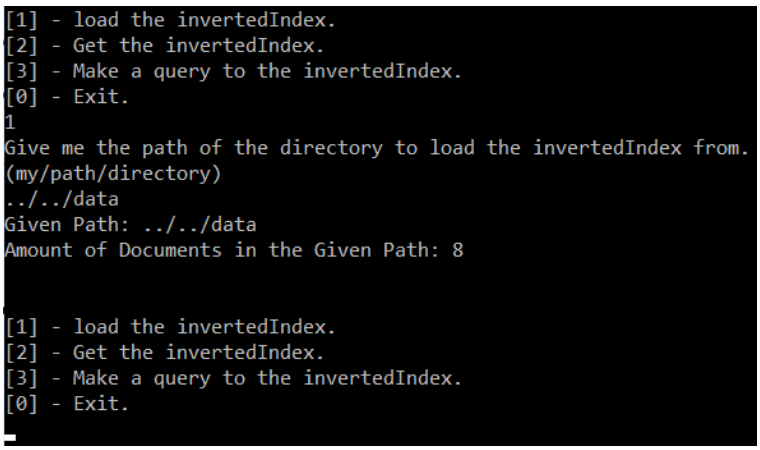
Ο χρήστης μπορεί να παίρνει αυτές τις επιλογές, μέχρις ότου να κάνει την επιλογή 0 και να βγει από την εφαρμογή. Στην περίπτωση που επιλέξει τις επιλογές 2 και 3 χωρίς από πριν να έχει αποθηκεύσει κάποιο λεξικό τότε του εμφανίζονται ανάλογα μηνύματα που τον ενημερώνουν αναλόγως (**Εικόνα 4**).



Εικόνα 4 – Ενημερώσεις χρήστη σχετικά με την λανθασμένη χρήση της εφαρμογής

2.1.2. Δημιουργία και αποθήκευση ενός ευρετηρίου στον δίσκο

Για την Δημιουργία και την αποθήκευση ενός ευρετηρίου στον δίσκο, πρέπει ο χρήστης να ακολουθήσει την επιλογή 1. Σε αυτήν την επιλογή, ο χρήστης θα κληθεί να δώσει ένα path για ένα directory από το οποίο θέλει να κατασκευάσει το λεξικό και το ευρετήριό του. Στην συνέχεια, θα εμφανιστεί στον χρήστη το μονοπάτι που έδωσε καθώς επίσης και το πλήθος των αρχείων που διαβάστηκαν ως επιβεβαίωση (**Εικόνα 5**).

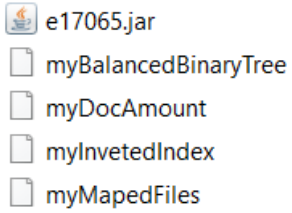


Εικόνα 5 – Εισαγωγή ενός μονοπατιού και η απάντηση της εφαρμογής

Μετά από την εισαγωγή του μονοπατιού μας, η εφαρμογή κατασκευάζει 4 αρχεία στο σημείο από το οποίο εκτελείτε (στην αναφορά αυτήν θα χρησιμοποιείται το jar για την εκτέλεση της εφαρμογής) για την αποθήκευση των αναγκαίων δεδομένων για να φορτώσει μελλοντικά το κατασκευασμένο ευρετήριο από τον δίσκο (**Εικόνα 6**). Τα 4 αρχεία αυτά είναι τα εξής:

- myBalanceBinaryTree: Αρχείο για την αποθήκευση ενός ισοζυγισμένου δυαδικού δέντρου, το οποίο χρησιμοποιείται για την γρήγορη εύρεση ερωτημάτων με χρήση μπαλαντέρ
- myDocAmount: Το πλήθος το αρχείων .txt τα οποία διαβάστηκαν

- myInvertedIndex: Το αντεστραμμένο ευρετήριο το οποίο χρησιμοποιείται για την γρήγορη εύρεση των λιστών καταχώρησης των όρων του λεξικού.
- myMapedFiles: Η σύνδεση των αρχείων που έχουν δοθεί με το id τους και το μονοπάτι στο οποίο βρίσκονται.



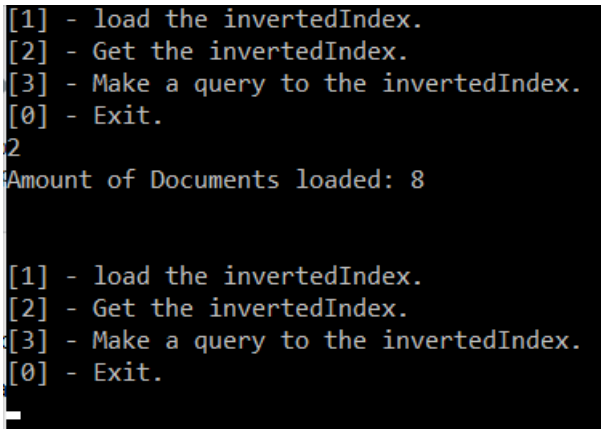
Εικόνα 6 – Αποθηκευμένα αρχεία από την εφαρμογή

Στην περίπτωση που ο χρήστης θέλει να αποθηκεύσει ένα νέο ευρετήριο και να διαγράψει το παλιό, τότε αρκεί να ξανατρέξει την επιλογή 1.

2.1.3. Φόρτωση ενός ευρετηρίου από τον δίσκο

Για την φόρτωση ενός ήδη υπάρχοντος ευρετηρίου από τον δίσκο (το οποίο εξηγήθηκε στο **Κεφάλαιο 2.1.2.**), ο χρήστης καλείται από την υλοποιημένη εφαρμογή να πάρει την επιλογή 2. Επιλέγοντάς της, η εφαρμογή θα φορτώσει τα αποθηκευμένα αρχεία και ως επιβεβαίωση στον χρήστη θα εμφανίσει τα γνωστά αρχεία στα οποία μπορεί να δώσει απαντήσεις σε δοθέντα ερωτήματα (**Εικόνα 7**).

Προσοχή! Η επιλογή 2 δεν θα λειτουργήσει στην περίπτωση που έχουν διαγραφτεί τα αρχεία που αποθηκεύει η εφαρμογή και θα φερθεί σαν να μην έχει εκτελεστεί η επιλογή 1 ποτέ!



Εικόνα 7 – Φόρτωση ευρετηρίου από τον δίσκο

2.1.4. Εκτέλεση ερωτημάτων στην Μηχανή Αναζήτησης

Μέσο της επιλογής 3, ο χρήστης μπορεί να δώσει Boole ερωτήματα στην Μηχανή Αναζήτησης και να λάβει τα μονοπάτια των εγγράφων στα οποία βρίσκεται η απάντησή του. Παρακάτω εξηγείτε ο τρόπος με τον οποίο η μηχανή αναζήτησης καταλαβαίνει τα ερωτήματα του χρήστη:

- Απαγορεύεται η χρήση παρενθέσεων (Θα «μπερδέψουν» τον αλγόριθμο που έχει υλοποιηθεί και πιθανώς θα καταλήξει σε λάθος αποτελέσματα).
- Οι λογικοί τελεστές AND, OR και NOT πρέπει να συντάσσονται με κεφαλαία γράμματα (Αλλιώς, θα «μπερδέψουν» τον αλγόριθμο που έχει υλοποιηθεί και πιθανώς θα καταλήξει σε λάθος αποτελέσματα).

- Οι λέξεις προς εύρεση μπορούν να γραφτούν με κεφαλαία και με μικρά γράμματα.
- Μπορούν να γραφτούν όσο μεγάλα ερωτήματα είναι αναγκαία από τον χρήστη.
- Απαγορεύεται η χρήση διπλής άρνησης στα ερωτήματα του χρήστη (Θα «μπερδέψουν» τον αλγόριθμο που έχει υλοποιηθεί και πιθανώς θα καταλήξει σε λάθος αποτελέσματα).
- Μπορούν να βρεθούν απαντήσεις σε ερωτήματα με μπαλαντέρ των μορφών: **X**, **X***, ***X**, ***X***, **X*Y**(Σε άλλες περιπτώσεις ο αλγόριθμος θα «μπερδευτεί» και πιθανώς θα καταλήξει σε λάθος αποτελέσματα).
- Η προτεραιότητά των τελεστών των ερωτημάτων είναι «Από τα αριστερά, στα δεξιά»

Παρακάτω φαίνονται η σύνταξη μερικών σωστών ερωτημάτων:

- NOT car
- cAr
- Fire OR NOT water
- fire OR water OR earth
- me AND you OR him AND his OR NOT friend
- c*r
- *re OR *no
- NOT *ye* AND su*
- Yes AND *nO* OR NOT true AND false

2.2. Παραδείγματα Χρήσης της Μηχανής Αναζήτησης

Στον κατάλογο «data» ο οποίος εμφανίζεται με τους υποκαταλόγους του στην **Εικόνα 1**, περιέχονται τα εξής αρχεία με το αντίστοιχο περιεχόμενο:

- html.txt:

"HTML (Hypertext Markup Language) is the code that is used to structure a web page and its content. For example, content could be structured within a set of paragraphs, a list of bulleted points, or using images and data tables."

- java.txt:

"Java is a programming language and computing platform first released by Sun Microsystems in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. Java is fast, secure, and reliable. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!"

- python.txt:

"Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects."

- basketball.txt:

"Basketball is a team sport in which two teams, most commonly of five players each, opposing one another on a rectangular court, compete with the primary objective of shooting a

basketball (approximately 9.4 inches (24 cm) in diameter) through the defender's hoop (a basket 18 inches (46 cm) in diameter mounted 10 feet (3.048 m) high to a backboard at each end of the court) while preventing the opposing team from shooting through their own hoop. A field goal is worth two points, unless made from behind the three-point line, when it is worth three. After a foul, timed play stops and the player fouled or designated to shoot a technical foul is given one, two or three one-point free throws. The team with the most points at the end of the game wins, but if regulation play expires with the score tied, an additional period of play (overtime) is mandated."

- football.txt:

"Football is a family of team sports that involve, to varying degrees, kicking a ball to score a goal. Unqualified, the word football normally means the form of football that is the most popular where the word is used. Sports commonly called football include association football (known as soccer in some countries); gridiron football (specifically American football or Canadian football); Australian rules football; rugby football (either rugby union or rugby league); and Gaelic football. These various forms of football share to varying extent common origins and are known as football codes."

- battlefield.txt:

"Battlefield is a series of first-person shooter video games developed by Swedish company EA DICE and is published by American company Electronic Arts. It started out on Microsoft Windows and OS X with Battlefield 1942, which was released in 2002. The Battlefield series has been played by more than 50 million players worldwide as of 2012, across 11 games and 12 expansion packs released since its inception in 2002."

- call-of-duty.txt:

"Call of Duty is a first-person shooter video game franchise published by Activision. Starting out in 2003, it first focused on games set in World War II. Over time, the series has seen games set in the midst of the Cold War, futuristic worlds, and outer space. The games were first developed by Infinity Ward, then also by Treyarch and Sledgehammer Games. Several spin-off and handheld games were made by other developers. The most recent title, Call of Duty: Black Ops Cold War, was released on November 13, 2020."

- league-of-legends.txt:

"League of Legends (LoL), commonly referred to as League, is a 2009 multiplayer online battle arena video game developed and published by Riot Games. Inspired by Defense of the Ancients, a custom map for Warcraft III, Riot's founders sought to develop a stand-alone game in the same genre. Since its release in October 2009, the game has been free-to-play and is monetized through purchasable character customization. The game is available for Microsoft Windows and macOS."

Παραπάνω, το περιεχόμενο του κάθε εγγράφου, περιέχει χρωματισμένες λέξεις για να δείξει κοινούς όρους μεταξύ των εγγράφων. Καθώς επίσης και μερικές υπογραμμισμένες λέξεις οι οποίες θα χρησιμεύσουν στην εξήγηση της χρήσης μπαλαντέρ στην συνέχεια.

2.2.1. Απλά ερώτημα

Στις **Εικόνες 8 – 14** φαίνονται οι απαντήσεις σε απλά ερωτήματα σχετικά με τις χρωματισμένες λέξεις στο περιεχόμενο που έχει ήδη αναφερθεί των εγγράφων.

```
[1] - load the invertedIndex.
[2] - Get the invertedIndex.
[3] - Make a query to the invertedIndex.
[0] - Exit.
3
Give me a query
language
Results:
DocID: 1 | It can be found here: ..\..\data\coding\html.txt
DocID: 2 | It can be found here: ..\..\data\coding\java.txt
DocID: 3 | It can be found here: ..\..\data\coding\python.txt
```

Εικόνα 8 – Ερώτημα «language»

```
Give me a query
programming
Results:
DocID: 2 | It can be found here: ..\..\data\coding\java.txt
DocID: 3 | It can be found here: ..\..\data\coding\python.txt
```

Εικόνα 9 – Ερώτημα «programming»

```
Give me a query
team
Results:
DocID: 4 | It can be found here: ..\..\data\sports\basketball.txt
DocID: 5 | It can be found here: ..\..\data\sports\football.txt
```

Εικόνα 10 – Ερώτημα «team»

```
Give me a query
football
Results:
DocID: 5 | It can be found here: ..\..\data\sports\football.txt
```

Εικόνα 11 – Ερώτημα «football»

```
Give me a query
basketball
Results:
DocID: 4 | It can be found here: ..\..\data\sports\basketball.txt
```

Εικόνα 12 – Ερώτημα «basketball»

```
Give me a query
game
Results:
DocID: 2 | It can be found here: ..\..\data\coding\java.txt
DocID: 4 | It can be found here: ..\..\data\sports\basketball.txt
DocID: 7 | It can be found here: ..\..\data\video-games\call-of-duty.txt
DocID: 8 | It can be found here: ..\..\data\video-games\league-of-legends.txt
```

Εικόνα 13 – Ερώτημα «game»

```
Give me a query
games
Results:
DocID: 6 | It can be found here: ..\..\data\video-games\battlefield.txt
DocID: 7 | It can be found here: ..\..\data\video-games\call-of-duty.txt
DocID: 8 | It can be found here: ..\..\data\video-games\league-of-legends.txt
```

Εικόνα 14 – Ερώτημα «games»

Όπως φαίνεται και στις σχετικές Εικόνες, η Μηχανή Αναζήτησης απαντάει στα ερωτήματα του χρήστη, δείχνοντας το ID του εγγράφου καθώς επίσης και το μονοπάτι προς αυτό, από το μέρος στο οποίο εκτελείτε η υλοποιημένη Μηχανή.

2.2.2. Ερωτήματα με Μπαλαντέρ

Στις **Εικόνες 15** και **16** φαίνονται οι απαντήσεις σε ερωτήματα με Μπαλαντέρ σχετικά με τις χρωματισμένες λέξεις στο περιεχόμενο που έχει ήδη αναφερθεί των εγγράφων.

```
Give me a query
*ball
Results:
DocID: 4 | It can be found here: ..\..\data\sports\basketball.txt
DocID: 5 | It can be found here: ..\..\data\sports\football.txt
```

*Εικόνα 15 – Ερώτημα «*ball»*

```
game*
Results:
DocID: 2 | It can be found here: ..\..\data\coding\java.txt
DocID: 4 | It can be found here: ..\..\data\sports\basketball.txt
DocID: 6 | It can be found here: ..\..\data\video-games\battlefield.txt
DocID: 7 | It can be found here: ..\..\data\video-games\call-of-duty.txt
DocID: 8 | It can be found here: ..\..\data\video-games\league-of-legends.txt
```

Εικόνα 16 – Ερώτημα «game»*

2.2.3. Ερωτήματα με τελεστή NOT

Ο τελεστής «NOT» θα επιστρέψει τα αντίθετα έγγραφα, από αυτά που θα επέστρεφε η μηχανή δίχως την ύπαρξή του. Σχετικό παράδειγμα φαίνεται στην **Εικόνα 17**.

```
Give me a query
NOT *ball
Results:
DocID: 1 | It can be found here: ..\..\data\coding\html.txt
DocID: 2 | It can be found here: ..\..\data\coding\java.txt
DocID: 3 | It can be found here: ..\..\data\coding\python.txt
DocID: 6 | It can be found here: ..\..\data\video-games\battlefield.txt
DocID: 7 | It can be found here: ..\..\data\video-games\call-of-duty.txt
DocID: 8 | It can be found here: ..\..\data\video-games\league-of-legends.txt
```

*Εικόνα 17 – Ερώτημα «NOT *ball»*

Συνεπώς, αντί να επιστρέψει στο ερώτημα «NOT *ball» τα έγγραφα που περιέχουν λέξεις που τελειώνουν σε «ball» (όπως οι λέξεις «football» και «basketball»), θα επιστρέψει έγγραφα τα οποία **ΔΕΝ** περιέχουν τέτοιες λέξεις. Παρόμοια χρήση μπορεί να γίνει και για ερωτήματα χωρίς μπαλαντέρ, όπου θα επιστραφούν έγγραφα χωρίς την εισαγόμενη λέξη.

2.2.4. Ερωτήματα με τελεστές AND και OR

Ο τελεστής «AND» θα επιστρέψει μόνο τα κοινά έγγραφα από το αριστερό του αποτέλεσμα και το δεξιό του αποτέλεσμα (**Εικόνα 18**). Από την άλλη, ο τελεστής «OR» θα επιστρέψει τα κοινά αλλά και τα μη κοινά έγγραφα των δύο αυτών αποτελεσμάτων (**Εικόνα 19**).

Με παρόμοιο τρόπο μπορούν αυτοί οι δύο τελεστές να συνοδευτούν και με ερωτήματα με χρήση Μπαλαντέρ, όπως επίσης και με ερωτήματα τα οποία περιέχουν την χρήση του τελεστή «NOT».

```
Give me a query
games AND game
Results:
DocID: 7 | It can be found here: ..\..\data\video-games\call-of-duty.txt
DocID: 8 | It can be found here: ..\..\data\video-games\league-of-legends.txt
```

Εικόνα 18 – Ερώτημα «games AND game»

```
games OR game
Results:
DocID: 2 | It can be found here: ..\..\data\coding\java.txt
DocID: 4 | It can be found here: ..\..\data\sports\basketball.txt
DocID: 6 | It can be found here: ..\..\data\video-games\battlefield.txt
DocID: 7 | It can be found here: ..\..\data\video-games\call-of-duty.txt
DocID: 8 | It can be found here: ..\..\data\video-games\league-of-legends.txt
```

Εικόνα 19 – Ερώτημα «games OR game»

2.2.5. Ερωτήματα με πάνω από δύο λέξεις

Η Μηχανή Αναζήτησης, μπορεί να υποστηρίξει και ερωτήματα με πάνω από 2 λέξεις. Ένα παράδειγμα μίας τέτοιας χρήσης φαίνεται στην **Εικόνα 20**, όπου εμφανίζονται τα έγγραφα τα οποία περιέχουν τις λέξεις game ή games αλλά όχι λέξεις οι οποίες να καταλήγουν σε «ball».

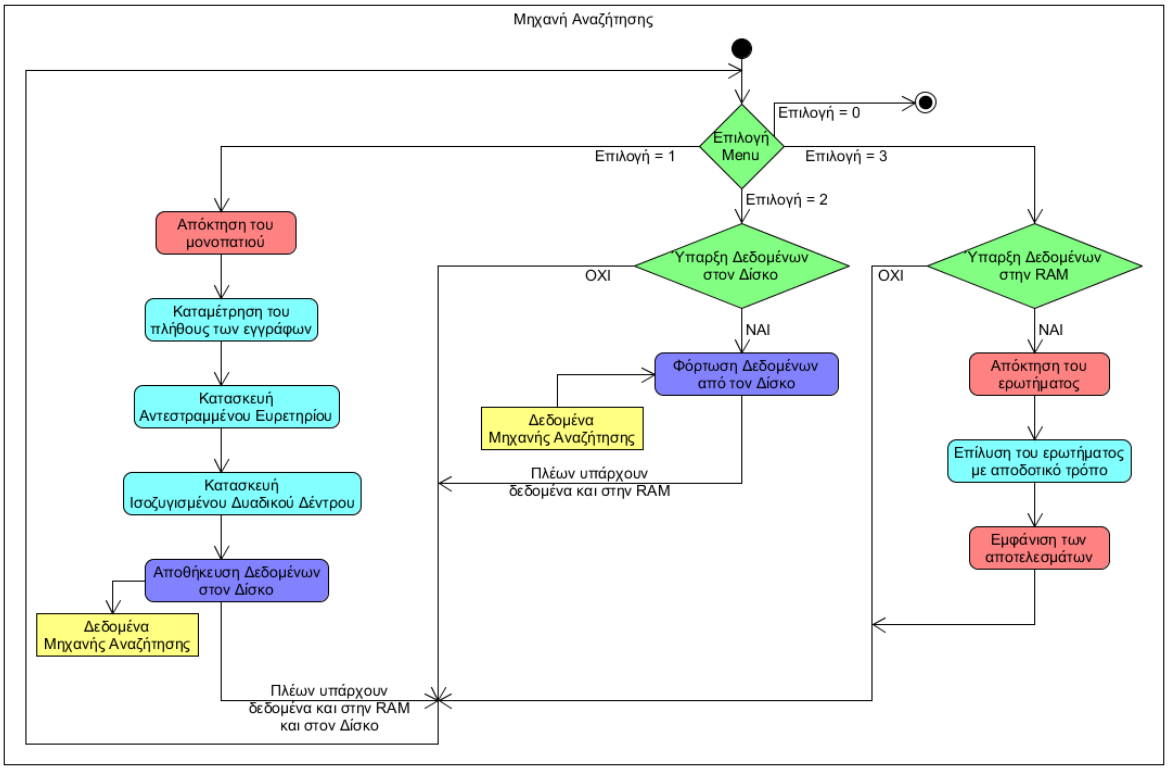
```
Give me a query
games OR game AND NOT *ball
Results:
DocID: 2 | It can be found here: ..\..\data\coding\java.txt
DocID: 6 | It can be found here: ..\..\data\video-games\battlefield.txt
DocID: 7 | It can be found here: ..\..\data\video-games\call-of-duty.txt
DocID: 8 | It can be found here: ..\..\data\video-games\league-of-legends.txt
```

*Εικόνα 20 – Ερώτημα «games OR game AND NOT *ball»*

3. Τεχνικό Εγχειρίδιο

3.1. Διαδικασία Εκτέλεσης

Η διαδικασία της λειτουργίας της Μηχανής Αναζήτησης εμφανίζεται στην **Εικόνα 21** συνολικά. Καθώς επίσης, στην **Εικόνα 22**, φαίνεται σε μεγαλύτερη ανάλυση η διαδικασία της απάντησης σε ένα δοθέν ερώτημα.



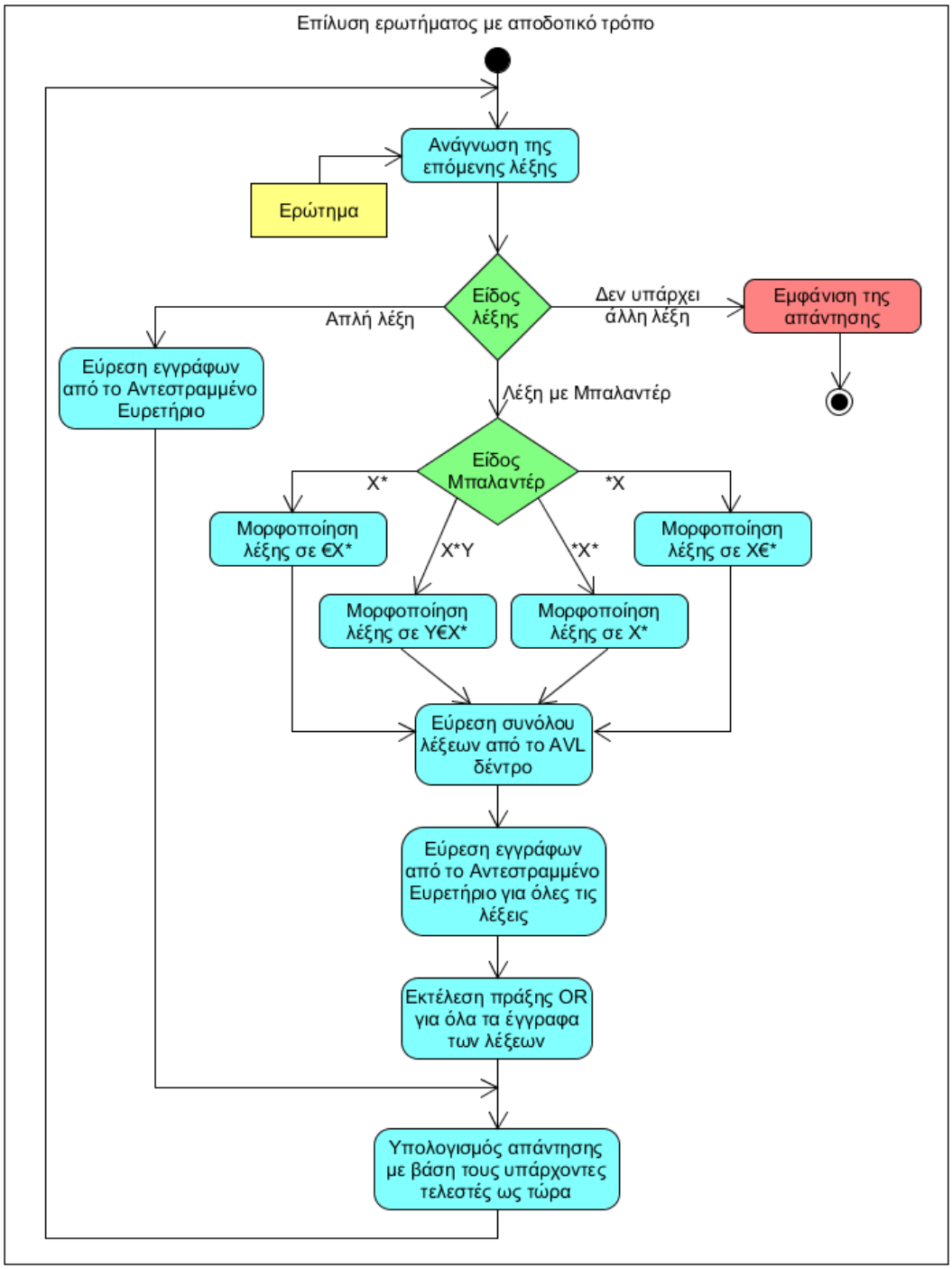
Εικόνα 21 – Διάγραμμα εκτέλεσης της Μηχανής Αναζήτησης

Όπως φαίνεται και στην **Εικόνα 22**, για την διαχείριση ερωτημάτων μπαλαντέρ γίνεται μορφοποίηση των δοθέντων ερωτημάτων με την βοήθεια ενός ιδικού χαρακτήρα (€), ο οποίος χρησιμοποιείτε και κατά την εισαγωγή των όρων στην κατασκευή του Ισοζυγισμένου Δυαδικού Δέντρου. Η υλοποίηση αυτή βασίζεται στον αλγόριθμο του βιβλίου «Introduction to Information Retrieval» σελ. 48 – 49.

Επίσης, το Ισοζυγισμένο Δυαδικό Δέντρο που υλοποιείται είναι της μορφής που περιγράφεται στο προαναφερθέν βιβλίο στις σελ. 45 – 47. Ακόμα, η εφαρμογή που έχει παραδοθεί εκμεταλλεύεται την μορφή αντεστραμμένων ευρετηρίων στην σελ. 6.

Η Κανονικοποίηση των όρων που εισάγονται από το document parser και το query parser είναι η εισαγωγή όλων των όρων ως πεζοί χαρακτήρες και χωρίς σημεία στίξης.

Στο **Κεφάλαιο 3.3** θα εξηγηθεί αναλυτικότερα η επιλογή της χρήσης Ισοζυγισμένου Δυαδικού Δέντρου μαζί με έναν κατακερματισμένο πίνακα για αντεστραμμένο ευρετήριο.

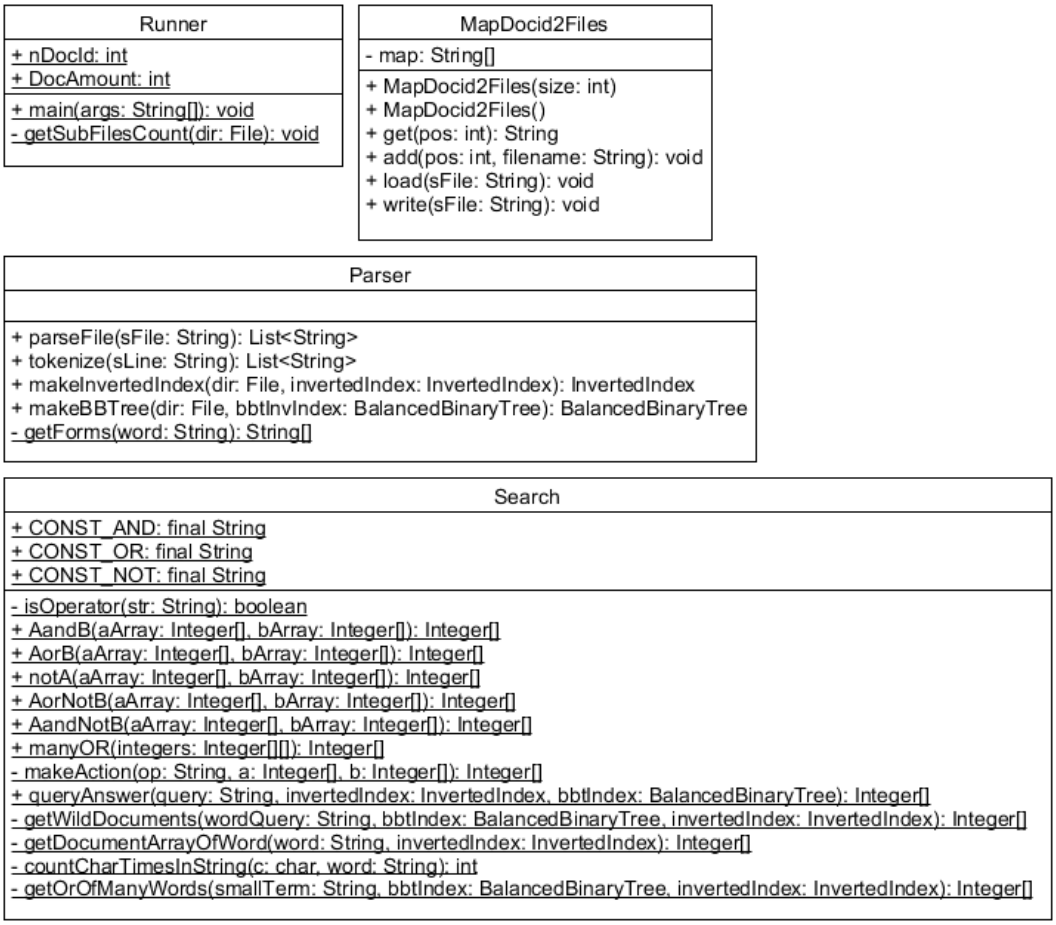


Εικόνα 22 – Διαδικασία απάντησης σε ένα δοθέν ερώτημα

3.2. Υλοποιημένες Κλάσης

Στις **Εικόνες 23 – 25** φαίνονται όλες οι Java κλάσεις οι οποίες παραδόθηκαν εντός του .zip αρχείου.

Η κλάση Runner, περιέχει την main μέθοδο και κατευθύνει τον κώδικα στις επιλογές που παίρνει ο χρήστης. Περιέχει μία μέθοδο για την καταμέτρηση του πλήθους .txt αρχείων εντός ενός καταλόγου. Όπως επίσης κατέχει και δύο βοηθητικά πεδία για την δημιουργία ενός νέου ευρετηρίου.



Εικόνα 23 – Runner, MapDocid2Files, Parser, Search

Η κλάση MapDocid2Files περιέχει έναν πίνακα από String τιμές οι οποίες για κάθε θέση (γραμμή - DocID) του πίνακα περιέχουν το αντίστοιχο μονοπάτι για το αντίστοιχο έγγραφο. Περιέχει μεθόδους για την δημιουργία αυτού του πίνακα, την προσθήκη ενός νέου εγγράφου μέσα του, την απόκτηση ενός μονοπατιού κάποιου εγγράφου, καθώς επίσης και μεθόδους για την αποθήκευσή του και την απόκτησή του πίνακα από τον σκληρό δίσκο.

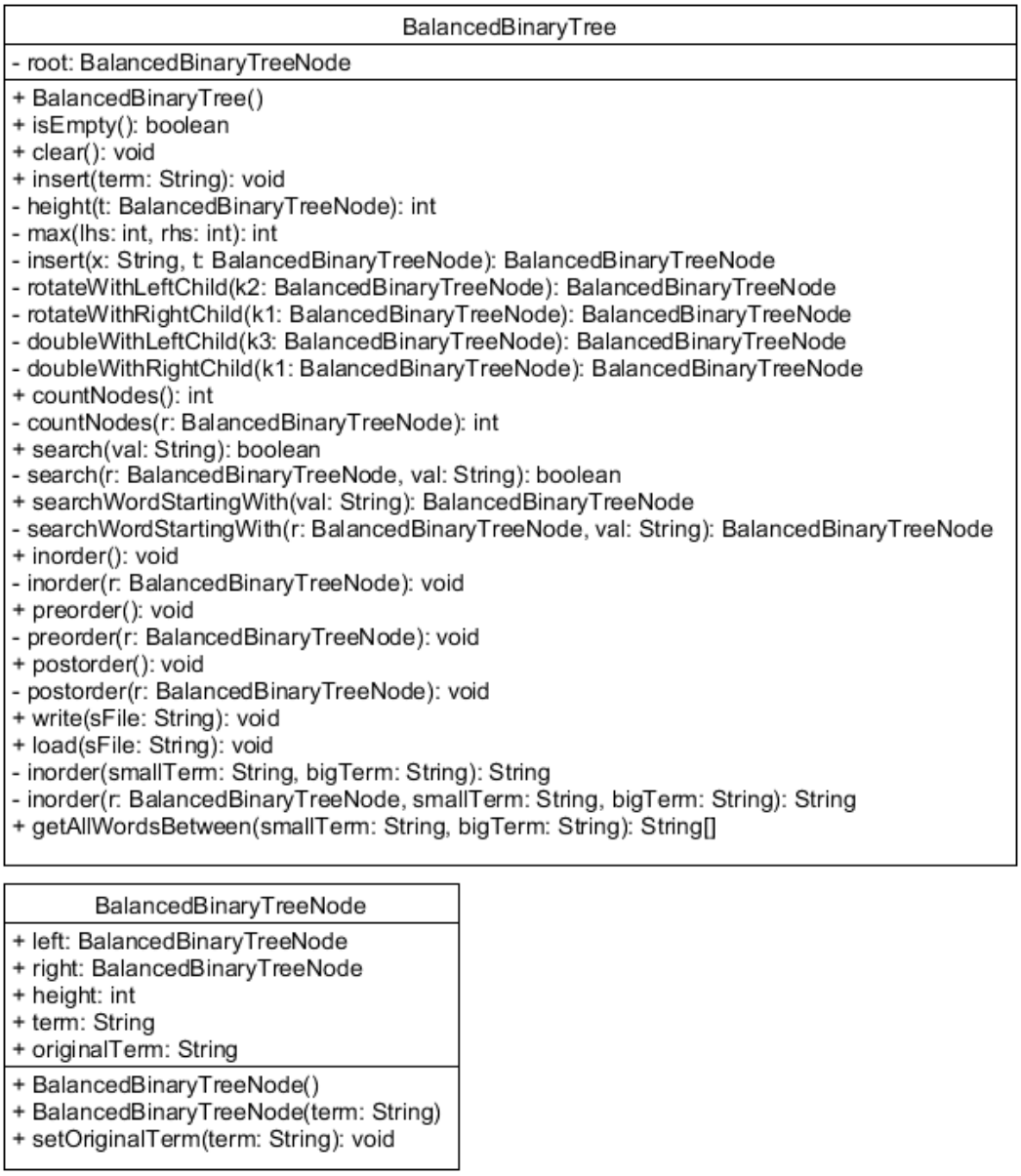
Η κλάση Parser περιέχει μεθόδους για την προσπέλαση καταλόγων και αρχείων και την κανονικοποιημένη δημιουργία όρων από τα έγγραφα τα οποία βρίσκει. Χρησιμοποιείται για το γέμισμα ενός Inverted Index και ενός Ισοζυγισμένου Δυαδικού Δέντρου μέσο ενός καταλόγου. Ακόμα για την διευκόλυνσή της στην κατασκευή του Δυαδικού Δέντρου χρησιμοποιείται η μέθοδος getForms η οποία παίρνει μία λέξη και την επιστρέφει σε όλες της τις μορφές με έναν ιδικό βοηθητικό χαρακτήρα (€). Πχ. Hello -> hello€, €hello, o€hell, lo€hel, llo€he, ello€h.

Η κλάση Search, χρησιμοποιείται κυρίως για την απάντηση σε ερωτήματα της Μηχανής Αναζήτησης. Περιέχει μέσα της μεθόδους που δέχονται πίνακες από ακεραίους (DocIDs) και επιστρέφει τις πράξεις τους μέσο των τελεστών «OR», «NOT», «AND». Η μέθοδος queryAnswer είναι αυτή που χρησιμοποιείται ως Parser για τα ερωτήματα τα οποία λαμβάνει η εφαρμογή. Διαβάζει την ερώτηση από τα αριστερά προς τα δεξιά και κατευθύνει κάθε λέξη που εμφανίζεται στο Δυαδικό Δέντρο (αν έχει μπαλαντέρ – μέσο της getWildDocuments) ή κατευθείαν στο Inverted Index. Η μέθοδος getWildDocuments θα ξεχωρίσει πιο είναι το είδος του ερωτήματος με Μπαλαντέρ και

θα βρει τον τρόπο με τον οποίο πρέπει να γραφτεί το ερώτημα προς το Δυαδικό Δέντρο. Στην συνέχεια, θα βρει την επόμενη λέξη **[2]** από την αναζητούμενη έκφραση και θα την χρησιμοποιήσει για να λάβει τα αποτελέσματα που θέλει να βρει ο χρήστης.

Πχ. Στην περίπτωση που θέλει να βρει τις λέξεις που ξεκινάνε με noη τότε θα αναζητήσει όλες τις x λέξεις όπου noη <= x < noο.

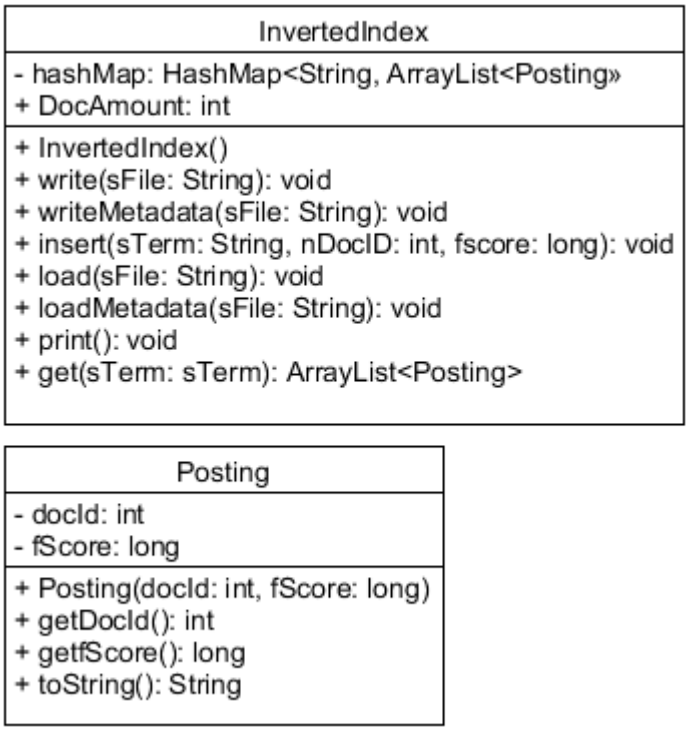
(Παραπάνω πληροφορία για την συγκεκριμένη διαδικασία φαίνεται στο **Κεφάλαιο 3.3**)



Εικόνα 24 – Κλάσεις Ισοζυγισμένου Δυαδικού Δέντρου

Οι κλάσεις στην **Εικόνα 24**, έχουν υλοποιηθεί με την βοήθεια της αναφοράς **[1]** και έχουν εμπλουτιστεί για την εφαρμογή τους στην Μηχανή Αναζήτησης. Η κλάση BalancedBinaryTreeNode είναι οι κόμβοι του Δυαδικού Δέντρου, η οποίοι περιέχουν και την κανονική λέξη εκτός από την λέξη η οποία κατέχει και τον χαρακτήρα «€». Ενώ η κλάση BalancedBinaryTree είναι η κλάση που κατασκευάζει το Ισοζυγισμένο Δυαδικό Δέντρο. Σε αυτήν, έχουν προστεθεί κλάσεις για την αποθήκευση του Δέντρου

και την απόκτησή του από τον σκληρό δίσκο, όπως επίσης αι κλάσεις οι οποίες λαμβάνουν δύο αλφαριθμητικά και επιστρέφουν όλες τις εγγραφές τους δέντρου που βρίσκονται αλφαβητικά ανάμεσα σε αυτά τα δύο ορίσματα. Αυτό το αποτέλεσμα επιστρέφεται ως ένα αλφαριθμητικό με κενά ανάμεσα σε κάθε λέξη. Η συγκεκριμένη μέθοδος μπορεί να επιστρέψει κάποια λέξη δύο φορές ή και παραπάνω, λόγο των πολλών διαφορετικών εγγραφών που μπορεί να έχει με τον βοηθητικό χαρακτήρα «€». Αυτό αντιμετωπίζεται με την επιστροφή των κανονικών λέξεων μέσα στο αλφαριθμητικό παρά των καταγραμμένων και στην συνέχεια με απαλοιφή των διπλοτύπων (Παραπάνω σχετικά με την διαχείριση αυτήν στο **Κεφάλαιο 3.3**).



Εικόνα 25 – Κλάσεις για την Διαχείριση των Documents των εγγράφων

Η κλάση InvertedIndex χρησιμοποιεί την κλάση Posting για την κατασκευή Λιστών εντός ενός κατακερματισμένου πίνακα. Αυτές οι Λίστες περιέχουν όλα τα έγγραφα που εμφανίζεται η κάθε λέξη του καταλόγου που δίνεται από τον χρήστη (όπως αναφέρθηκε και στο **Κεφάλαιο 3.1**).

3.3. Τα βήματα επίλυσης ενός ερωτήματος με Μπαλαντέρ

Στην περίπτωση απόκτησης ενός απλού ερωτήματος τότε ο κώδικας απλά χρησιμοποιεί την κλάση InvertedIndex απευθείας για να βρει σε O(1) χρόνο την Λίστα την οποία έχει ανάγκη για κάθε λέξη, ώστε να ολοκληρώσει τις πράξεις που είναι αναγκαίες για την απάντηση του ερωτήματος. Σε αυτό το κεφάλαιο θα αναπτυχθεί η διαχείριση και ο λόγος της επιλογής της χρήσης ενός Ισοζυγισμένου Δυαδικού Δέντρου μαζί με έναν Πίνακα Κατακερματισμού σε αντίθεση με την υλοποίηση απευθείας σε AVL δέντρο του ευρετηρίου.

Θεωρώντας ότι το μέσο μήκος τον όρων του ευρετηρίου είναι k γράμματα. Τότε ένα AVL δέντρο θα έχει συνολικά k φορές μεγαλύτερο αριθμό κόμβων από ότι οι θέσεις του Κατακερματισμένου Πίνακα λόγο της χρήσης του ιδικού χαρακτήρα «€» στο Δέντρο.

Κρατώντας την κανονική λέξη κάθε κόμβου του Δυαδικού Δέντρου (κρατάμε δηλαδή και την lo€hel αλλά και την hello στον ίδιο κόμβο) μπορούμε να γνωρίζουμε ότι διπλασιάζοντας τις αποθηκευμένες λέξεις του Δέντρου, έχουμε δείκτες απευθείας στις Posting λίστες των αντίστοιχων λέξεων εντός του Κατακερματισμένου Πίνακα.

Συνεπώς δεν έχουμε την Ανάγκη να αποθηκεύουμε k φορές την κάθε Posting List εντός του Δέντρου αυτού, αλλά μόνο μία φορά εντός του Κατακερματισμένου Πίνακα. Συμπεραίνεται εύκολα λοιπόν, ότι για συλλογές εγγράφων με μέσο όρο μήκους των όρων των εγγράφων k>2 (κάτι το οποίο είναι πολύ συχνό – σχεδόν ισχύει πάντα) συμφέρει να χρησιμοποιούμε αυτές τις δύο δομές μαζί, παρά ξεχωριστά στην περίπτωση που θέλουμε να υποστηρίξουμε ερωτήματα με Μπαλαντέρ.

Κάθε λέξη εντός του Ισοζυγισμένου Δυαδικού Δέντρου, χρειάζεται συνολικά O(log(N*k)) - θεωρώντας ότι υπάρχουν N λέξεις εντός του καταλόγου που έχει δοθεί - για να βρεθεί (κάτι το οποίο μπορεί να βελτιωθεί ακόμα παραπάνω με την χρήση B-Δέντρων).

Κάνοντας το ερώτημα «*n*» (άρα θα αναζητηθούν οι εγγραφές «n*») μπορούν να επιστραφούν τα παρακάτω:

- nomenon€phe → phenomenon
- non€phenome → phenomenon
- n€phenomeno → phenomenon
- n€rai → rain
- nstructionist€co → constructionist
- nist€constructio → constructionist

Ο κώδικας που έχει παραδοθεί λαμβάνει O(m) χρόνο – θεωρώντας ότι βρίσκει m εγγραφές σε ένα ερώτημα – για να καθαρίσει τις διπλοεγγραφές. Όπως επίσης έλαβε O(log(N*k) + m) χρόνο για να βρει όλες τις λέξεις αφού βρίσκοντας την πρώτη, όλες οι υπόλοιπες λέξεις είναι δεξιά από αυτήν σε σειρά «in order» μέχρις ότου κάποια να μην ξεκινάει από το επιθυμητό όρο.

Συνεπώς θα καταλήξει στις παρακάτω λέξεις:

- phenomenon
- rain
- constructionist

Για κάθε λέξη χρειάζεται O(1) χρόνος για την απόκτηση της Λίστας τους.

Οπότε για την απόκτηση όλων των αναγκαίων λιστών για την επίλυση ενός ερωτήματος χρειάζεται συνολικά:

Χρόνος = O(log(N*k) + m) + O(m) + **mO(1)**

Το παραπάνω επειδή στην χειρότερη δεν θα είναι καμία λέξη διπλοεγγραφή!

Οπότε: Χρόνος = O(log(N*k) + m) + 2O(m) = O(log(N*k) + m) + O(m)

Άρα: Χρόνος = O(log(N*k) + m)

4. Αναφορές

[1]. https://www.sanfoundry.com/java-program-create-balanced-binary-tree-incoming-data/
[2]. https://stackoverflow.com/questions/2899301/how-do-i-increment-a-variable-to-the-next-or-previous-letter-in-the-alphabet