ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάκτηση Πληροφορίας
Ακαδημαϊκό Έτος 2023-2024

<<Μηχανή Αναζήτησης πληροφορίας από επιστημονικά άρθρα>>

ΑΝΑΦΟΡΑ

ΜΑΙΟΣ 2024

Νικολέτα Κακούρη ΑΜ: 4369

Θεοδώρα Ξούλου ΑΜ: 4452

1.ΕΙΣΑΓΩΓΗ

Στην παρούσα εργασία σχεδιάστηκε και υλοποιήθηκε ένα σύστημα αναζήτησης πληροφορίας σχετικά με επιστημονικά άρθρα, χρησιμοποιώντας τη βιβλιοθήκη Lucene. Στόχος είναι η δημιουργία μίας μηχανής αναζήτησης πληροφοριών που επιτρέπει την αποδοτική αναζήτηση και παρουσίαση των αποτελεσμάτων με βάση τη συνάφειά τους με το ερώτημα του χρήστη.

2.ΣΥΛΛΟΓΗ ΕΓΓΡΑΦΩΝ

Τα έγγραφα που ανακτώνται από την μηχανή αναζήτησης είναι επιστημονικά άρθρα από τη συλλογή NIPS Papers 1987-2017, διαθέσιμα στο Kaggle. Η συλλογή αποτελείται από 370 άρθρα στα οποία έχουμε κάνει εκκαθάριση χωρίς κάποιο συγκεκριμένο κριτήριο και έχουμε επιλέξει τα πεδία που θα χρησιμοποιηθούν . Για κάθε επιστημονικό άρθρο οι πληροφορίες που έχουμε άρα και τα πεδία στα οποία η μηχανή αναζήτησης θα κάνει αναζήτηση είναι τα εξής:

- “title”(τίτλος του άρθρου)
- “year” (χρονιά δημοσίευσης του άρθρου)
- “abstract”(περίληψη του άρθρου)
- “full text”(πλήρες κείμενο του άρθρου)

3.ΚΑΤΑΣΚΕΥΗ ΕΥΡΕΤΗΡΙΟΥ

Όπως αναφέρθηκε η ανάκτηση των πληροφοριών που γίνεται σε αυτή την εφαρμογή αναζήτησης στηρίζεται στην βιβλιοθήκη Lucene.

Πριν την ευρετηριοποίηση των δεδομένων, που είναι το αμέσως επόμενο βήμα της υλοποίησης, απαιτείται η ανάλυση και η προεπεξεργασία του κειμένου χρησιμοποιώντας τα APIs της βιβλιοθήκης.

Εκμεταλλευτήκαμε τις λειτουργίες της Lucene (απαλοιφή stop words, uppercases, stemming κλπ.) για την προετοιμασία του κειμένου. Συγκεκριμένα χρησιμοποιήθηκε η ανάλυση της Lucene (CustomAnalyzer) που περιλαμβάνει τα εξής βήματα επεξεργασίας κειμένου:

1) **Αφαίρεση Stop Words**
   Με τη χρήση του StopFilter αφαιρεί κοινές λέξεις ώστε το κείμενο να γίνεται πιο συμπυκνωμένο και εστιασμένο στο σημαντικό περιεχόμενο.
2) **Stemming**
   Με τη χρήση του PorterStemFilter μειώνει τις λέξεις στη ρίζα τους άρα και την ποικιλομορφία των λεκτικών μορφών και διευκολύνει την αναζήτηση.

3) **Επέκταση Συνωνύμων-Ακρωνύμων**
   Με τη χρήση του SynonymGraphFilter προσθέτει συνώνυμες λέξεις ώστε να αντιστοιχίζονται κείμενα με παρόμοιο νόημα αλλά διαφορετική διατύπωση.
4) **Απαλοιφή διακριτικών**
   Με τη χρήση του ASCIIFoldingFilter αφαιρούνται τα διακριτικά από τους χαρακτήρες διασφαλίζοντας ότι οι λέξεις με διακριτικά θεωρούνται το ίδιο με τις αντίστοιχες χωρίς διακρτικά.
5) **Μετατροπή λεκτικών μονάδων σε πεζά γράμματα**
   Με τη χρήση του LowerCaseFilter διασφαλίζεται ότι διαφορετικές μορφές της ίδιας λέξης κανονικοποιούνται σε μία κοινή μορφή.
6) **Διαχωρισμός λέξεων με βάση τα κενά διαστήματα**
   Με τη χρήση του WhitespaceTokenizer διασφαλίζεται ότι οι λέξεις διαχωρίζονται με βάση τα κενά διαστήματα.

Αποθηκεύουμε το ευρετήριο στον δίσκο ώστε να είναι μόνιμο και προσβάσιμο ακόμα και μετά την επανεκκίνηση του προγράμματος ή του υπολογιστή. Εισάγουμε τα δεδομένα (το αρχείο csv) χρησιμοποιώντας τη μέθοδο “normalizeText” για να διασφαλίσουμε ότι αποθηκεύονται σε κανονικοποιημένη μορφή ώστε να έχουμε λιγότερες διαφορετικές μορφές των ίδιων λέξεων.

Επεξήγηση Κώδικα

1.Εισαγωγή APIs

2.Αρχικά δημιουργούμε τον CustomAnalyzer ο οποίος κάνει προεπεξεργασία των δεδομένων χρησιμοποιώντας τα φίλτρα που αναφέρθηκαν παραπάνω.

3.Αρχικοποιούμε τον IndexWriter, καθορίζουμε ότι θα χρησιμοποιηθεί ο CustomAnalyzer και αποθηκεύουμε το ευρετήριο στον δίσκο.

4. Κανονικοποιούμε τα κείμενα χρησιμοποιώντα την normalizeText() αφαιρώντας τους διακριτικούς χαρακτήρες και αντικαθιστώντας τους με κενές συμβολοσειρές.

5.Δημιουργούμε μία μέθοδο που διαβάζει τα δεδομένα από το CSV αρχείο γραμμή - γραμμή και τα εισάγει στο ευρετήριο. Για να διαβάσει σωστά το αρχείο και τα πεδία που έχει:

5.1 Παραλείπει τη πρώτη γραμμή του αρχείου καθώς είναι η γραμμή κεφαλίδας που περιέχει τα ονόματα των στηλών και όχι τα δεδομένα που θέλουμε να ευρετηριάσουμε.

5.2 Για κάθε γραμμή εντοπίζει τις θέσεις των πρώτων τριών κομμάτων για να διαχωρίσει τα πεδία του αρχείου..

5.3 Κάνει τους απαραίτητους ελέγχους για να εξάγει τα τέσσερα πεδία:

5.4 Στην περίπτωση που υπάρχουν τουλάχιστον δύο κόμματα εξάγει τα πεδία «title» και «year» και τα αποθηκεύει. Τα πεδία «abstratct» και «full text» είναι κενά.

5.5 Στην περίπτωση που υπάρχει και τρίτο κόμμα εξάγονται και τα πεδία «abstract» και «full text».
5.6 Αν δεν υπάρχει τρίτο κόμμα, επειδή παρατηρήσαμε ότι το πεδίο «abstract» είναι τις περισσότερες φορές άδειο, το υπόλοιπο της γραμμής θεωρείται ως «full text».
5.7 Για κάθε γραμμή που διαβάζει δημιουργείται ένα νέο έγγραφο όπου προστίθενται τα πεδία και το έγγραφο αυτό εισάγεται στο ευρετήριο.

6. Δημιουργούμε μία μέθοδο που ελέγχει αν τα αρχεία ενός φακέλου είναι αρχείο CSV και αν ναι τα εισάγει στο ευρετήριο.

Για την παραπάνω διαδικασία χρησιμοποιήθηκαν τα εξής API της Lucene: IndexWriterConfig, IndexWriter, Document, Directory, FSDirectory, TextField, LowerCaseFilter, StopFilter, PorterStemFilter, SynonymGraphFilter, ASCIIFoldingFilter.

4. ΑΝΑΖΗΤΗΣΗ

Στη συνέχεια πραγματοποιείται η αναζήτηση στο ευρετήριο και η επιστροφή των αποτελεσμάτων. Η συγκεκριμένη υλοποίηση υποστηρίζει τρία είδη αναζήτησης.

- Η απλή αναζήτηση χρησιμοποιείται όταν ο χρήστης δεν έχει περαιτέρω πληροφορίες για τα επιστημονικά άρθρα όπως ολόκληρο τον τίτλο ή τη χρονιά δημοσίευσης τους οπότε τα αναζητά με κάποια λέξη – κλειδί που πιθανώς περιέχεται είτε στον τίτλο, είτε στην περίληψη του άρθρου είτε στο κείμενο του άρθρου.

Επεξήγηση Κώδικα

1. Δημιουργούμε μία μέθοδο που παίρνει ως είσοδο ένα ερώτημα αναζήτησης.
2. Αναλύει το ερώτημα χρησιμοποιώντας τον QueryParser.
3. Εκτελεί την αναζήτηση στο ευρετήριο.
4. Εξάγει όλα τα αποτελέσματα.
5. Δημιουργείται μία λίστα με τα αποτελέσματα.
6. Το ερώτημα καταγράφεται στο ιστορικό αναζήτησης.
7. Οι λέξεις – κλειδιά από το ερώτημα αναζήτησης επισημαίνονται με έντονα γράμματα στα αποτελέσματα που εξάγονται ώστε να είναι πιο εύκολα ορατές στον χρήστη.

- Όταν ο χρήστης διαθέτει τις πληροφορίες μπορεί να κάνει αναζήτηση βάσει ενός συγκεκριμένου πεδίου συμπληρώνοντάς το ώστε να του εμφανιστεί το επιθυμητό αποτέλεσμα. Είναι πιο σύνηθες να συμβαίνει με την συμπλήρωση του τίτλου ή της χρονιάς δημοσίευσης του άρθρου.

Επεξήγηση Κώδικα

1. Δημιουργούμε μία μέθοδο που κάνει αναζήτηση βάσει συγκεκριμένου πεδίου χρησιμοποιώντας τον QueryParser.
2. Εκτελεί την αναζήτηση στο ευρετήριο.
3. Εξάγει όλα τα αποτελέσματα.
4. Δημιουργείται μία λίστα με τα αποτελέσματα.
5. Το ερώτημα καταγράφεται στο ιστορικό αναζήτησης.
6. Οι λέξεις – κλειδιά από το ερώτημα αναζήτησης επισημαίνονται με έντονα γράμματα στα αποτελέσματα που εξάγονται ώστε να είναι πιο εύκολα ορατές στον χρήστη.

- Μία επιπλέον δυνατότητα είναι να πραγματοποιεί αναζήτηση άρθρων σε ένα συγκεκριμένο χρονικό εύρος. Στην υλοποίηση αυτή ο χρήστης δίνει ως είσοδο δύο χρονιές και επιστρέφονται όσα άρθρα έχουν δημοσιευτεί σε αυτό το χρονικό διάστημα.

Επεξήγηση Κώδικα

1. Η μέθοδος που δημιουργούμε εκτελεί μία αναζήτηση για ένα συγκεκριμένο πεδίο με καθορισμένο άνω και κάτω όριο.
2. Ανακτά όλα τα αποτελέσματα της αναζήτησης.
3. Δημιουργεί μία λίστα με αυτά.
4. Καταγράφει το ερώτημα αναζήτησης στο ιστορικό.
5. Οι λέξεις – κλειδιά από το ερώτημα αναζήτησης επισημαίνονται με έντονα γράμματα στα αποτελέσματα που εξάγονται ώστε να είναι πιο εύκολα ορατές στον χρήστη.

- Σε οποιαδήποτε φάση της αναζήτησης ο χρήστης μπορεί να δει τις προηγούμενες αναζητήσεις του καθώς κρατείται ιστορικό αναζήτησης.

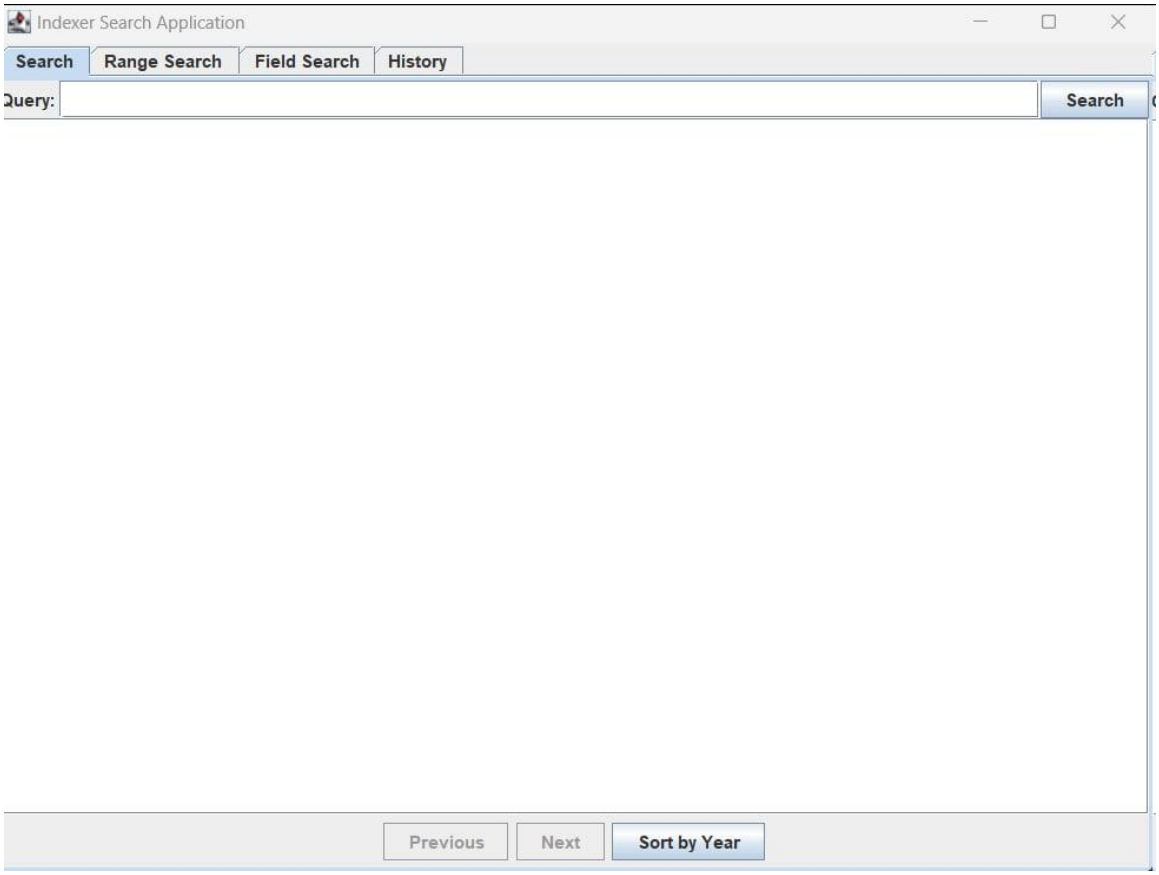
Επεξήγηση Κώδικα

1. Δημιουργούμε μία μέθοδο η οποία καταγράφει κάθε τύπο αναζήτησης σε ένα αρχείο ιστορικού (search_history.txt) σημειώνοντας την ημερομηνία και την ώρα που πραγματοποιήθηκε η αναζήτηση.

Για την παραπάνω διαδικασία χρησιμοποιήθηκαν τα εξής API της Lucene: StandardAnalyzer, DirectoryReader, IndexReader, IndexSearcher, QueryParser, Query, TopDocs, ScoreDoc και FSDirectory.

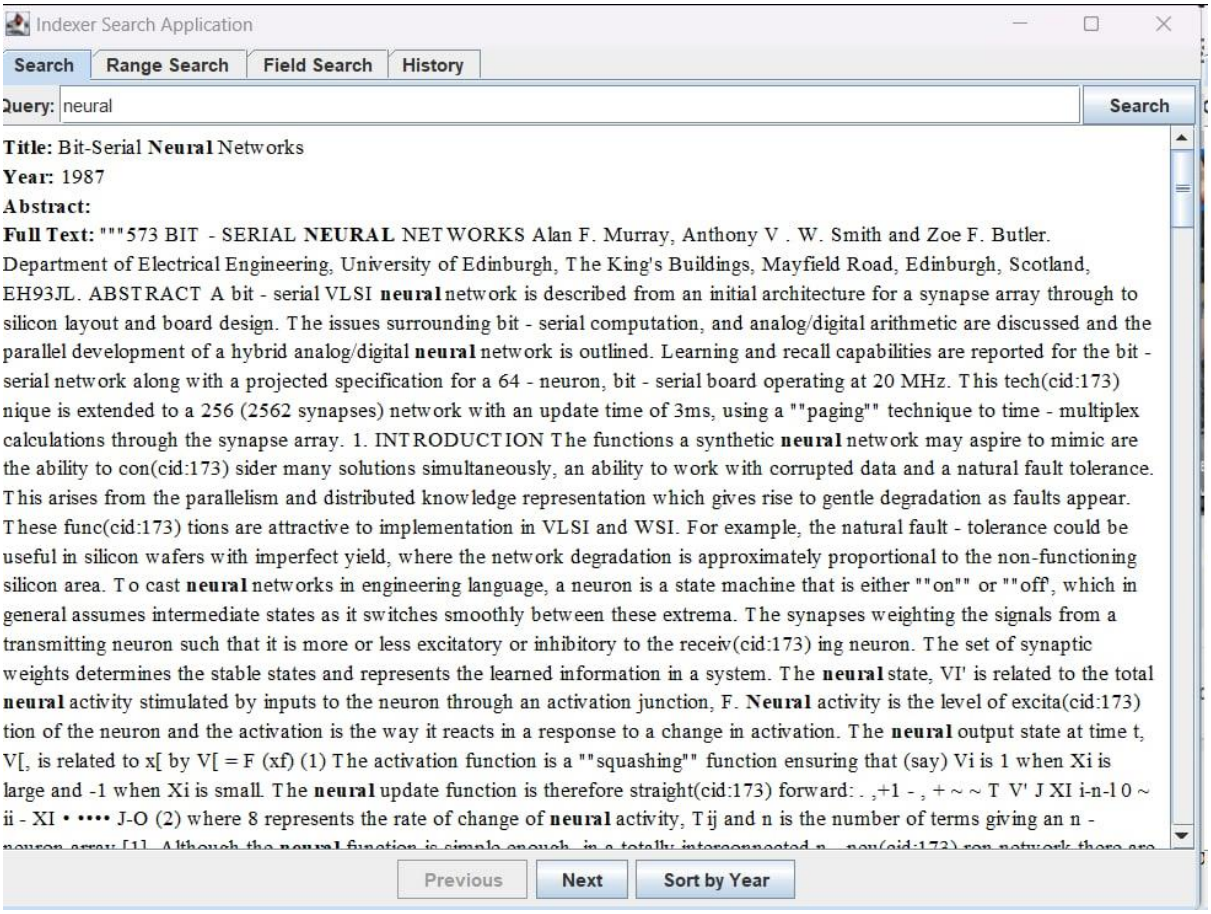
5. ΠΑΡΟΥΣΙΑΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Τελειώνοντας τη διαδικασία της αναζήτησης περνάμε στη διαδικασία της παρουσίασης. Ο χρήστης επιλέγει τον τρόπο με τον οποίο θέλει να υλοποιήσει την αναζήτησή του ανάλογα με τις πληροφορίες που διαθέτει. Σε οποιαδήποτε περίπτωση πατάει το κουμπί “Search” και αναμένει να δει τα αποτελέσματα της ερώτησής του. Έπειτα επιστρέφονται στην οθόνη του χρήστη τα συναφή με το ερώτημα αποτελέσματα.



1.Κεντρικό παράθυρο αναζήτησης

- Στην περίπτωση του 1ου τρόπου αναζήτησης ο χρήστης εισάγει τη λέξη – κλειδί στο πεδίο που ονομάζεται «Query:» και πατάει το κουμπί αναζήτησης.



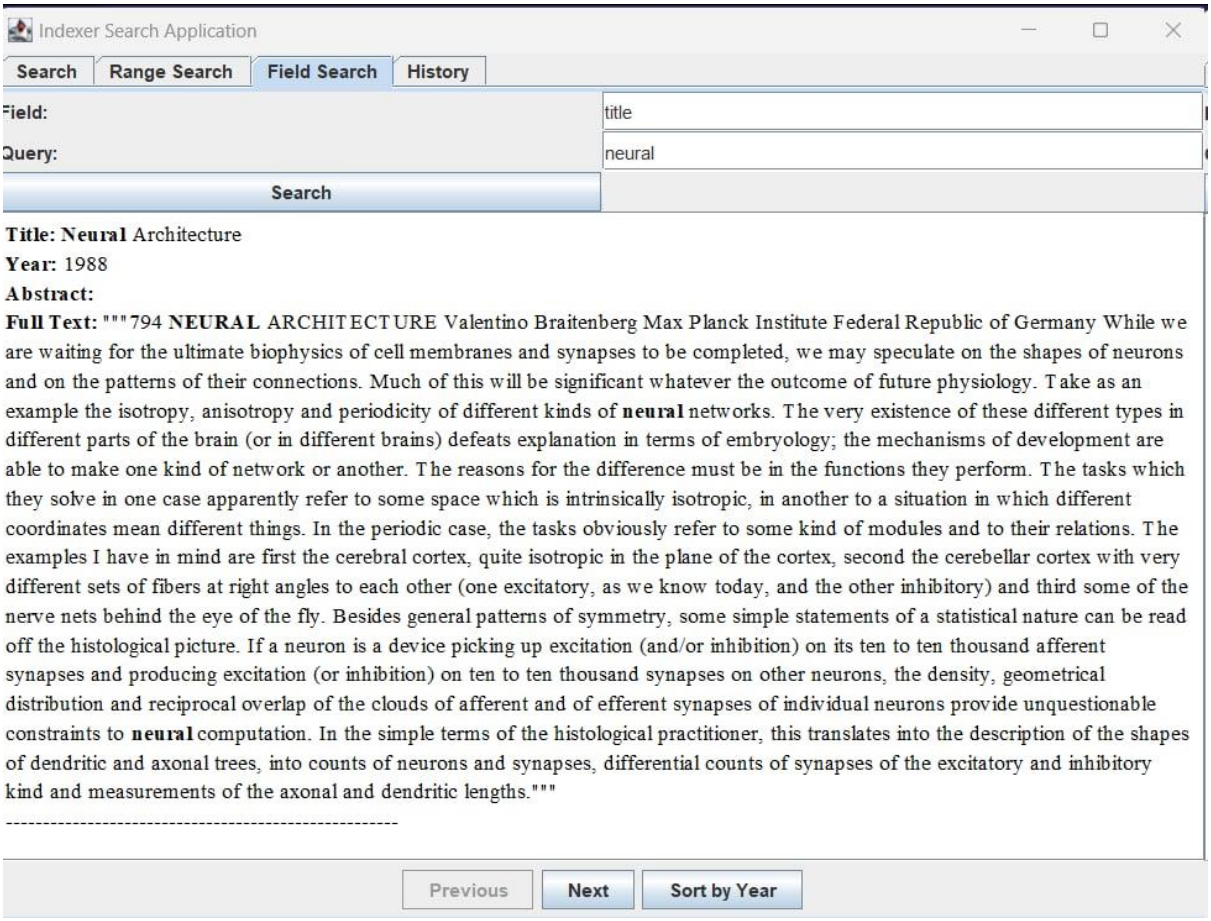
2. Εμφάνιση αποτελεσμάτων για την αναζήτηση "neural"

Όπως παρατηρούμε επιστρέφονται τα άρθρα τα οποία είναι συναφή με το ερώτημα του χρήστη (στη συγκεκριμένη περίπτωση περιέχουν τον όρο “neural”). Η λέξη – κλειδί είναι τονισμένη στα αποτελέσματα και υπάρχει η δυνατότητα, πατώντας το κουμπί «Next» να διατρέξει και τα υπόλοιπα αποτελέσματα.

Επεξήγηση Κώδικα

1. Δημιουργούμε την κλάση «SearchPanel» που περιλαμβάνει ένα πεδίο κειμένου «JTextField» για να εισάγει ο χρήστης το ερώτημα και ένα κουμπί αναζήτησης «JButton».
2. Δημιουργούμε το παράθυρο που θα εμφανίζονται τα αποτελέσματα «JTextPane» και υποστηρίζουμε τη δυνατότητα προσπέλασης των αποτελεσμάτων «JScrollPane» με τη προσθήκη κουμπιών πλοήγησης «Previous» και «Next».
3. Σε αυτό το σημείο προσπαθήσαμε να υλοποιήσουμε την εμφάνιση των αποτελεσμάτων ανά 10 ωστόσο δεν καταφέραμε την δυνατότητα αυτή.
4. Εισάγαμε ένα κουμπί ταξινόμησης «Sort by Year» για την ταξινόμηση των αποτελεσμάτων ανά έτος.
5. Δημιουργήσαμε την μέθοδο «performSearch» ώστε να πραγματοποιείται η αναζήτηση και να εκτυπώνονται τα αποτελέσματα στην οθόνη, συμπεριλαμβανομένης της περίπτωσης να μην υπάρχουν αποτελέσματα.
6. Δημιουργήσαμε την μέθοδο «updateResults» για να ενημερώνονται τα αποτελέσματα που εμφανίζονται στην οθόνη.
7. Δημιουργήσαμε την μέθοδο «sortResults» για να ταξινομούνται τα αποτελέσματα με βάση τη χρονιά δημοσίευσης τους.
8. Δημιουργήσαμε τη μέθοδο «extractYear» ώστε να ξεχωρίζουμε το έτος από κάθε άρθρο και να κάνουμε την ταξινόμησή τους.

- Στην περίπτωση του 2ου τρόπου αναζήτησης πατάμε το κουμπί “Field Search” και εισάγουμε στο «Field» το πεδίο στο οποίο θέλουμε να ψάξουμε τον συγκεκριμένο όρο, τον οποίο αναγράφουμε στο πεδίο «Query».



F 3.Εμφάνιση αποτελεσμάτων για την αναζήτηση "title-neural"

Επιστρέφονται τα άρθρα τα οποία περιέχουν στον τίτλο τους τον όρο «». Επιπλέον υπάρχει και εδώ η δυνατότητα προσπέλασης των υπόλοιπων αποτελεσμάτων.

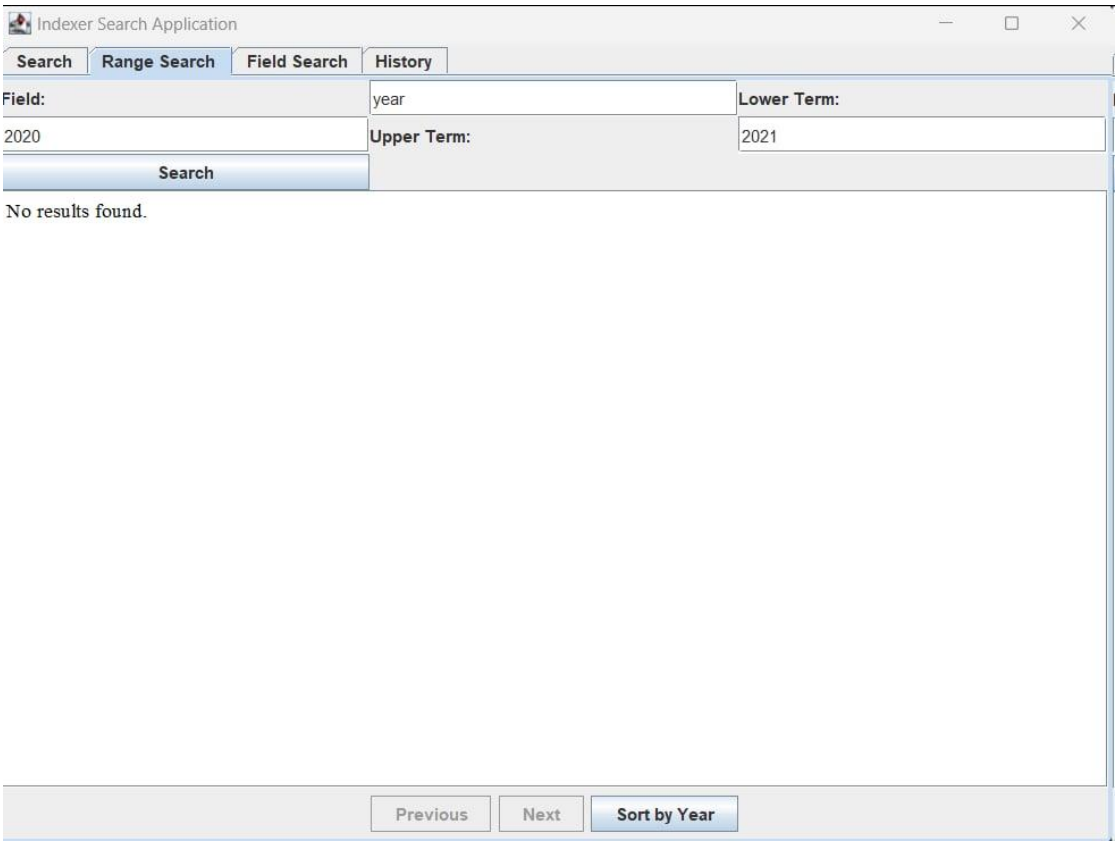
Επεξήγηση Κώδικα

1. Δημιουργούμε την κλάση «FieldSearhPanel» που περιλαμβάνει δύο πεδία κειμένου «JTextField», ένα για το πεδίο αναζήτησης και ένα για το ερώτημα αναζήτησης και ένα κουμπί αναζήτησης «JButton».

Αυτή είναι και η μόνη διαφορά στην υλοποίηση του κώδικα σε σχέση με την κλάση «SearchPanel» που εξηγήσαμε παραπάνω.

- Στην περίπτωση του 3ου τρόπου αναζήτησης εισάγουμε στο «Field» το «year» καθώς και το εύρος των χρονολογιών ανάλογα: την πιο πρόσφατη χρονολογία στο «Upper Term » και την πιο παλιά στο «Lower Term».

  Στο συγκεκριμένο παράδειγμα που παρατίθεται έχουμε εισάγει χρονολογίες μεταξύ των οποίων δεν υπάρχουν δημοσιευμένα άρθρα. Στην περίπτωση που δεν υπάρχουν αποτελέσματα συναφή με το ερώτημα του χρήστη η μηχανή αναζήτησης επιστρέφει «No results found».



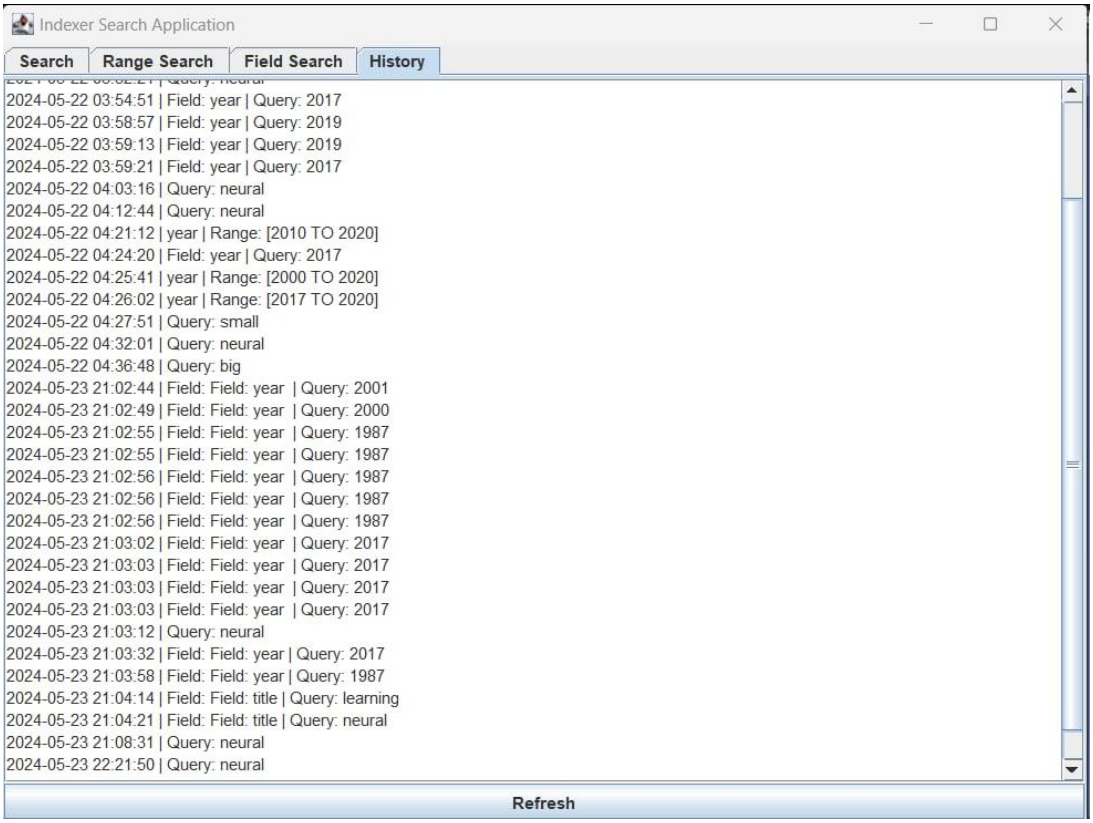
4.Εμφάνιση αποτελεσμάτων για αναζήτηση με χρονολογικό εύρος

Επεξήγηση Κώδικα

1. Δημιουργούμε την κλάση «RangeSearhPanel» που περιλαμβάνει τρία πεδία κειμένου «JTextField», ένα για το πεδίο αναζήτησης, ένα για τον κάτω όρο και έναν για τον άνω όρο και ένα κουμπί αναζήτησης «JButton».

Αυτή είναι και η μόνη διαφορά στην υλοποίηση του κώδικα σε σχέση με την κλάση «SearchPanel» που εξηγήσαμε παραπάνω.

- Στην περίπτωση που θέλουμε να περιηγηθούμε στο ιστορικό των αναζητήσεων μας πατάμε το κουμπί «History» και επιστρέφονται στην οθόνη μας όλα τα προηγούμενα ερωτήματα που είχαμε θέσει. Αν θέλουμε να ανανεώσουμε τη λίστα με τα αποτελέσματα που μας παρέχονται πατάμε το κουμπί «Refresh».



5.Εμφάνιση ιστορικού αναζήτησης

Επεξήγηση Κώδικα

1. Δημιουργούμε την κλάση «HistoryPanel» η οποία περιλαμβάνει ένα «JTextArea» για την εμφάνιση του ιστορικού αναζητήσεων.
2. Για να υποστηρίξουμε την δυνατότητα κύλισης των αποτελεσμάτων ώστε να μπορούμε να τα διαβάσουμε όλα το προσθέτουμε σε ένα «JScrollPane».
3. Δημιουργούμε ένα κουμπί «Refresh» για την ανανέωση του ιστορικού αναζητήσεων.
4. Δημιουργούμε μία μέθοδο «loadHistory» η οποία καλείται όταν δημιουργείται το συγκεκριμένο παράθυρο και η οποία ανακτά το ιστορικό των αναζητήσεων το οποίο έχουμε αποθηκεύσει σε ένα txt αρχείο και το εκτυπώνει στην οθόνη.

Σχεδιασμός Κώδικα

Στην υλοποίηση μας έχουμε δημιουργήσει 2 πακέτα.

- **indexer**: Περιέχει τις κλάσεις Indexer στην οποία γίνεται το indexing του αρχείου και δημιουργείται το ευρετήριο και τις Searcher, FieldSearcher, RangeSearcher και SearchHistoryLogger όπου η κάθε μία κάνει διαφορετικό τύπο αναζήτησης.
- **gui**: Περιέχει τις κλάσεις «SearchPanel», «FieldSearchPanel», «RangeSearchPanel» και «HistoryPanel» όπου υλοποιείται το γραφικό περιβάλλον της κάθε αναζήτησης.