virusQuest

Μηχανή Αναζήτησης Για Άρθρα Σχετικά με τον COVID-19

ΜΥΥ003 : Ανάκτηση Πληροφορίας

Εργασία Εαρινού Εξαμήνου 2020 - 2021

Νεφέλη Ελευθερία Σέξτου AM: 4485
Θωμαή Τσιαούση AM: 4510

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

Περιεχόμενα

0. Εισαγωγή

Η μηχανής αναζήτησης VirusQuest επιστρέφει αποτελέσματα από τη συλλογή corpus που έχει συγκεντρωθεί, τα οποία θα πρέπει να ικανοποιούν την ανάγκη πληροφόρησης του χρήστη σχετικά με τον COVID-19. Επίσης, η μηχανή υποστηρίζει διάφορα είδη ερωτημάτων του χρήστη με βάση λέξεις κλειδιά και επιστρέφει αποτελέσματα παρουσιασμένα με βάση τη συνάφεια (ranking) τους.

Η βιβλιοθήκη που χρησιμοποιήθηκε για την υλοποίηση της VirusQuest είναι η Apache Lucene 8.8.1. Συγκεκριμένα, τα jars που χρησιμοποιήθηκαν:

- lucene-core-8.8.1.jar
- lucene-demo-8.8.1.jar
- lucene-queryparser-8.8.1.jar
- lucene-suggest-8.8.1.jar

Για τη δημιουργία του User Interface (GUI) χρησιμοποιήθηκε η βιβλιοθήκη swing της java, όπως και το εργαλείο WindowBuilder το οποίο παρέχει το Eclipse (IDE).

Τέλος, χρησιμοποιήθηκε η βιβλιοθήκη commons io 2.9.0 (commons-io-2.9.0.jar), για λειτουργίες σχετικές με το ιστορικό της μηχανής.

1. Συλλογή Άρθρων και Προεπεξεργασία

1.1 Συλλογή Άρθρων

Η συλλογή (corpus) που επιλέχθηκε να χρησιμοποιηθεί για την λειτουργία της μηχανής αναζήτησης, είναι ένα κομμάτι μιας έτοιμης συλλογής από την ιστοσελίδα kaggle[1], και αποτελείται από δεδομένα που έχουν συλλεχθεί από online πηγές. Η αρχική συλλογή (τύπου .csv) αποτελούνταν από περίπου 369.000 αρχεία, τα οποία είχαν δημοσιευθεί από τον Ιανουάριο του 2020 μέχρι και το Δεκέμβριο του 2020.

Από την παραπάνω συλλογή, επιλέχθηκε ένα δείγμα 700 αρχείων, το οποίο χρησιμοποιεί η VirusQuest για την αναζήτηση και εξαγωγή αποτελεσμάτων. Τα αρχεία καλύπτουν ένα ευρύ φάσμα θεμάτων, από επιστημονικά άρθρα και οικονομικά άρθρα μέχρι και αναρτήσεις στον τομέα της ψυχαγωγίας, που κάνουν αναφορά στην πανδημία, του ιού covid-19 (SARS-CoV-2). Οι συγγραφείς των άρθρων ποικίλουν, καθώς και η ημερομηνία δημοσίευσης τους εκτείνεται σε όλη τη διάρκεια του προηγούμενου έτους.

1.2 Προεπεξεργασία Δεδομένων

Το δείγμα των 700 δεδομένων που κρατήθηκαν σε ένα αρχείο .csv, μετατράπηκε μέσω του αρχείου *covid.py*, σε 700 διαφορετικά αρχεία τύπου .txt, ώστε να γίνει η ανάλυση και επεξεργασία των δεδομένων τους σε πρώτο στάδιο και τελικά η παρουσίαση τους στον χρήστη μετά την αναζήτηση.

Στην πρώτη σειρά του κάθε αρχείου βρίσκεται το όνομα του συγγραφέα (Author), αν υπάρχει, στη δεύτερη η κατηγορία στην οποία υπάγεται το άρθρο (Topic), στην τρίτη σειρά η ημερομηνία δημοσίευσης του (Date), στην τέταρτη ο τίτλος (Title), και από την πέμπτη μέχρι και το τέλος του αρχείου, το περιεχόμενο του άρθρου (Body).

**Σημείωση:* Χρησιμοποιήθηκε η κωδικοποίηση Western European (Windows).

[1] Covid-19 Public Media Dataset by Anacode | Kaggle

2. Ανάλυση και Ευρετηριοποίηση

2.1 Γενικά

Κατά την ανάλυση, χωρίζουμε το περιεχόμενο των αρχείων .txt που βρίσκονται στη συλλογή, σε επιμέρους πεδία τα οποία είναι τα:

i. Author: ο συγγραφέας του άρθρου,
ii. Topic: το θέμα / κατηγορία που υπάγεται,
iii. Date: η ημερομηνία δημοσίευσης του,
iv. Title: ο τίτλος του άρθρου και
v. Body: το περιεχόμενο.

Για καθένα από τα παραπάνω πεδία, δημιουργείται ένα ευρετήριο, στο οποίο αποθηκεύονται οι όροι του κάθε πεδίου (stored term vectors), αφού έχει γίνει ανάλυση.

2.2 Ανάλυση

Η κλάση στην οποία υλοποιείται η ανάλυση της συλλογής δεδομένων (corpus) είναι η *indexMaker.java*. Πρώτα, αρχικοποιούνται επτά (private) πεδία, τα int n (αριθμός αρχείων στη συλλογή), File dirFile, Directory dir. τα IndexWriter idxWriter, IndexWriterConfig config, FieldType metadateFieldSet και FieldType contentFieldSet.

Στον constructor δέχεται ως όρισμα τον αριθμό n και το path στο οποίο θα αποθηκευτεί το ευρετήριο. Το dir, μέσω του dirFile που έχει λάβει το κατάλληλο dirpath, ανοίγει το Directory στο οποίο θα γίνει η αποθήκευση. Ο idxWriter είναι αυτός που δημιουργεί και διατηρεί τα indexes, ενώ το config, αποφασίζει αν δημιουργείται ένα νέο index ή αν αυτό «ανοίγει» από κάποιο προϋπάρχων. Επίσης, τα πεδία metadateFieldSet και contentFieldSet, περιγράφουν τις ρυθμίσεις για τα μεταδεδομένα και περιεχόμενα του κάθε Document που έχει γίνει Index (για παράδειγμα αν γίνονται stored ή tokenized).

Η ανάλυση γίνεται με χρήση του StandardAnalyzer, που παρέχει η Lucene 8.8.1, Ο οποίος αναγνωρίζει συγκεκριμένα είδη tokens, μετατρέπει τη γραμματοσειρά των όρων σε lowercase, αφαιρεί stop words, χρησιμοποιώντας μια διαμορφώσιμη λίστα.

2.3 Δημιουργία Ευρετηρίου

Στη μέθοδο indexer() ξεκινά η διαδικασία ευρετηριοποίησης για καθένα από τα διαφορετικά πεδία Author, Topic, Date, Title και Body που έχουν φτιαχτεί, διαβάζοντας το περιεχόμενο του κάθε αρχείου text file που βρίσκεται στο corpus και αναλύοντας τα με τον StandardAnalyzer. Τα indexes των πεδίων μαζί με τα properties του καθενός, αποθηκεύονται σε μία δομή Document doc, και τελικά το doc αποθηκεύεται στο ευρετήριο μέσω του idxWriter, ώστε να γίνει αργότερα η αναζήτηση.

Με άλλα λόγια:

indexer():

i. Δεν δέχεται κάποιο όρισμα και είναι void.
ii. Εκτελεί τη διαδικασία του indexing.
iii. Για n ίσο με τον αριθμό των .txt αρχείων που βρίσκονται στη συλλογή, παίρνει το path, αρχικοποιεί τα πεδία, και κάνει indexing διαβάζοντάς τα κατάλληλα από κάθε γραμμή του αρχείου.
iv. Αποθηκεύει τα indexes των πεδίων στο doc.
v. Ο idxWriter αποθηκεύει το doc στο ευρετήριο.
vi. Κλείνει τον IndexWriter και το Directory από το οποίο γινόταν η ανάγνωση.

3. Αναζήτηση

3.1 Είδη Ερωτημάτων Που Υποστηρίζονται

- Ερώτημα απλής λέξης κλειδιού - *παράδειγμα*: **coronavirus**
- Boolean ερώτημα - *παράδειγμα*: **coronavirus AND vaccine** ή **government OR virus**
- Ερωτήματα στα οποία χρησιμοποιούνται οι τελεστές **+ , - , * , ~**
  Για παράδειγμα:
  +virus +vaccine: να περιέχεται η λέξη virus και η λέξη vaccine.
  +virus -vaccine: να περιέχεται η λέξη virus αλλά να μη περιέχεται η λέξη vaccine.
  gov*: να περιέχονται λέξεις που ξεκινούν με το πρόθεμα gov.
  irus~: λέξεις που μοιάζουν με το irus.
- Αναζήτηση με βάση ένα από τα πεδία : author, topic, date, title, body.
  Παράδειγμα: **title: Vaccine**
- Σύνθετα ερωτήματα όπως:
  author:Paul LeBlanc body:irus~
  author:Paul LeBlanc AND NOT title:Fauci
  author:Paul LeBlanc -title:Fauci
  (virus OR coronavirus) AND US
  (virus OR coronavirus) AND title:US
  body:"use masks"~5 εμφάνιση των use και masks με απόσταση μέχρι 5 θέσεων μεταξύ τους.

Σημείωση*: τις ημερομηνίες δεν τις εντοπίζει εάν δεν έχουν γραφεί ολογράφως με τη μορφή **μήνας/μέρα/έτος
Για παράδειγμα: **2/15/2020**

3.2 Ιστορικό Και Εναλλακτικά Ερωτήματα

Το ιστορικό αναζητήσεων διατηρείται σε ένα αρχείο κειμένου history.txt που βρίσκεται εντός του φακέλου του project, εάν όμως δεν υπάρχει δημιουργείται.

Το ιστορικό μπορεί ο χρήστης ανά πασά στιγμή να το εμφανίσει όπως ακριβώς είναι γραμμένο στο αρχείο με το πάτημα του κουμπιού **History** στη διεπαφή χρήστη καθώς και να το διαγράψει με το πάτημα του κουμπιού **CLR Hist.** Μετά από κάθε αναζήτηση εμφανίζονται στο κάτω δεξιά μέρος του User Interface προτεινόμενα ερωτήματα σχετικά με προηγούμενες αναζητήσεις του χρήστη, με την προϋπόθεση το ιστορικό να μην είναι άδειο.

Ο έλεγχος ορθογραφικών λαθών λειτουργεί επίσης ως μια μορφή πρότασης ερωτημάτων καθώς παρουσιάζει λέξεις που είναι κοντά με το ερώτημα του χρήστη, συγκρίνοντας το με όρους του λεξικού που έχει δημιουργηθεί από τα άρθρα που έχουν ευρετηριοποιηθεί. Οι προτάσεις αυτές παρουσιάζονται πάνω δεξιά.

3.3 Αναζήτηση

Η κλάση που αφορά την αναζήτηση είναι η Searcher (*Searcher.java*). Τα private πεδία της αποτελούνται από δύο αντικείμενα κλάσης File, δύο αντικείμενα κλάσης Directory, ένα αντικείμενο τύπου IndexReader, πέντε αντικείμενα κλάσης IndexWriterConfig, ένα αντικείμενο κλάσης IndexSearcher και ένα SpellChecker.

Στον constructor πρέπει να περαστούν ως ορίσματα δύο paths. Το πρώτο είναι το path για το index που περιέχει τα ευρετηριοποιημένα άρθρα και το δεύτερο είναι το path για το index του λεξικού που θα χρησιμοποιηθεί για τον έλεγχο ορθογραφικών λαθών. Όλα τα προαναφερθέντα πεδία αρχικοποιούνται εντός του constructor.

Υπάρχουν τρεις μέθοδοι:

search:

i. Παίρνει ως όρισμα ένα String (είσοδος του χρήστη).
ii. Εδώ δημιουργείται ένας MultiQueryParser ώστε να μπορεί να γίνει αναζήτηση σε όλα τα πεδία.
iii. Δημιουργείται το Query και καλείται η search του indexSearcher ώστε να γίνει η αναζήτηση στο index και να επιστραφούν μέχρι 700 TopDocs, όσος είναι ο συνολικός αριθμός άρθρων του corpus.
iv. Η μέθοδος επιστρέφει τα TopDocs που επιστράφηκαν κατά την αναζήτηση.

spellSuggestions:

i. Παίρνει ως όρισμα ένα String (είσοδος του χρήστη).
ii. Καλείται η μέθοδος suggestSimilar του SpellChecker και επιστρέφονται δύο αποτελέσματα (προτάσεις).
iii. Ελέγχεται αν ο πίνακας στον οποίο επιστρέφονται τα αποτελέσματα δεν είναι κενός και αν έχει θετικό μήκος και αν αυτό είναι αληθές προστίθενται σε ένα ArrayList.
iv. Η μέθοδος επιστρέφει το ArrayList με τις προτάσεις.

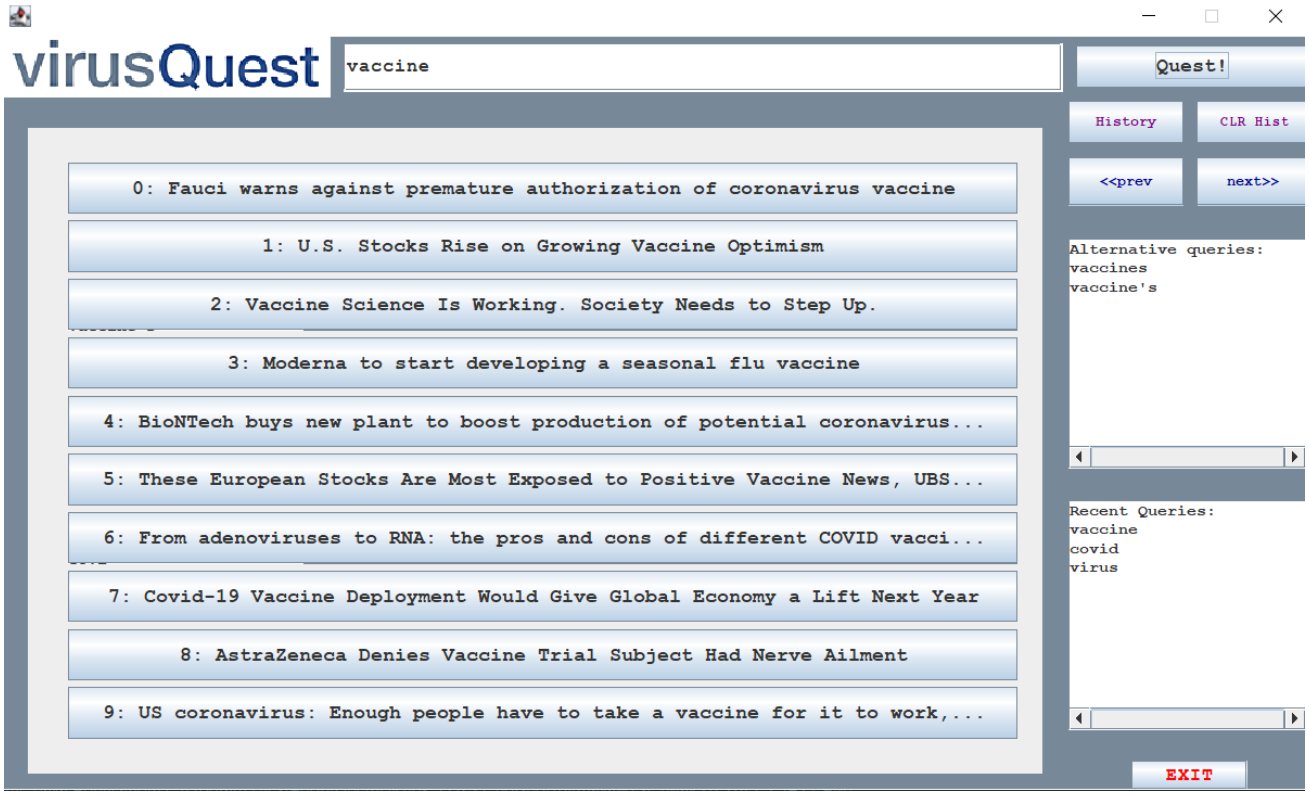
nextPrevField :

i. Παίρνει ως όρισμα, ένα αντικείμενο TopDocs, δύο int που θα χρησιμοποιηθούν σε επαναλήψεις για να κατευθυνθεί στο κατάλληλο μέρος του scoreDocs πεδίου του TopDocs και ένα String ώστε να μπορεί να δεχτεί το πεδίο.
ii. Αφού ληφθεί το docId δημιουργείται ένα Document, παίρνει το περιεχόμενο του πεδίου, το αποθηκεύει σε ένα String, το προσθέτει σε ένα ArrayList και γίνεται έλεγχος τερματισμού για το αν έχει φτάσει στο τέλος του topDocs ώστε να σταματήσει να προσθέτει σε αυτή τη περίπτωση.
iii. Η μέθοδος επιστρέφει ένα ArrayList που περιέχει το περιεχόμενο του δοσμένου πεδίου κάθε Document που εξήχθη από το topDocs.

closeSearcher :

i. Κλείνει τον IndexReader και τα Directories που είχαν ανοιχτεί για να υπάρχει πρόσβαση στα indexes.

4. Παρουσίαση Αποτελεσμάτων



4.1 Λειτουργίες

Quest: Εκτέλεση αναζήτησης, καλεί την μέθοδο searcher με την είσοδο του χρήστη και τοποθετεί τους τίτλους των άρθρων που επιστρέφει η αναζήτηση στα κουμπιά που μπορεί ο χρήστης να πατήσει για να εμφανίσει κάποιο άρθρο σε ξεχωριστό παράθυρο.

History: Εκτέλεση ανάγνωσης του history.txt και παρουσίασης του περιεχομένου του σε ξεχωριστό παράθυρο.

CLR Hist : Διαγραφή του περιεχομένου του history.txt.

next>>: Αν υπάρχουν επόμενα αποτελέσματα καλεί την nextPrevField (με ανανεωμένα τα πεδία start και end) για όλα τα πεδία και ανανεώνει τους τίτλους στα κουμπιά, εμφανίζοντας στο χρήστη τους τους επόμενους τίτλους.

<<prev: Αν υπάρχουν προηγούμενα αποτελέσματα καλεί την nextPrevField (με ανανεωμένα τα πεδία start και end) για όλα τα πεδία και ανανεώνει τους τίτλους στα κουμπιά, εμφανίζοντας στο χρήστη τους τους επόμενους τίτλους.

EXIT: Πριν κλείσει την εφαρμογή καλεί την closeSearcher του αντικειμένου Searcher που δημιουργήθηκε όταν δημιουργήθηκε το κεντρικό frame.

Επιπλέον Σημειώσεις:

- Πριν εμφανιστούν τα γραφικά εκτελείται μια φορά το indexing. Με αυτό τον τρόπο, η μόνη αλλαγή που χρειάζεται αν αλλάξει το corpus (με τους τίτλους να παραμένουν με την ίδια μορφή και τα paths ίδια) θα πρέπει να αλλάξει μόνο το όρισμα του πάνω ορίου των πιθανών επιστρεφόμενων TopDocs και το int όρισμα του στη δημιουργία του αντικειμένου indexMaker.
- Το αντικείμενο Searcher που χρησιμοποιείται καθ' όλη τη διάρκεια της αναζήτησης δημιουργείται μέσα στη mainframe.
- Όταν πατιέται ένα από τα κουμπιά τίτλων ανοίγει ένα νέο παράθυρο με το περιεχόμενο του άρθρου το οποίο έχει ληφθεί από τη κατάλληλη θέση των αντίστοιχων global ArrayList για κάθε πεδίο. Αν δεν υπάρχει τίτλος κάποιου άρθρου απλά ανοίγει ένα παράθυρο χωρίς περιεχόμενο.
- Στα δύο παράθυρα στα δεξιά εμφανίζονται οι προτάσεις εναλλακτικής ορθογραφικής διατύπωσης και οι τρεις πιο πρόσφατες αναζητήσεις του χρήστη αντίστοιχα.