מרינה בוטביניק 317522605 , דור אלקבץ 2040

דוח מנוע אחזור – חלק ב'

- **להלן המחלקות שהוספנו בחלק ב':**

1. **מחלקת Searcher:**

מחלקה האחראית על חיפוש המסמכים הרלוונטיים לשאילתה או למקבץ שאילתות שהיא מקבלת. המחלקה תיגש לקבצי הפוסטינג שייצרנו בחלק א' ותוציא משם את המידע הרלוונטי לכל שאיתה.

פונקציות המחלקה:

- **Searcher(boolean stem, String path, boolean isSemantic,Parse parser)** – בנאי המחלקה אשר אחראי על אתחול כל המשתנים , אתחול הנתיב לנתיב הפוסטינג הנחוץ (עם Stem או שבלי) , מעלה את שני המילונים של ה-terms ושל המסמכים לזיכרון, ומאתחל את האורך הממוצע של כל המסמכים בקורפוס.
- **void fillEntities()**
הפונצקיה תקרא מהזיכרון את הפוסטינג בו כתוב עבור כל מסמך את 5 הישויות הנפוצות ביותר שלו וכמות הפעמים שהן הופיעו , ותעלה את זה לזרון בצורת מילון כך שנוכל להשתמש בו בהמשך.
- **void fillDictionaries()**
הפונקציה תעבור על כל המידע ששמרנו עבור כל מסמך בקבצי הפוסטינג של המסמכים, ותייבא את האינפורמציה על אורך המסמך וכמה פעמים המילה הכי נפוצה הופיע בו, ותאתחל את המשתנים d_docLength ו-d_docmaxTF בהתאם.
- **void search(LinkedHashMap<String,String> queries)**
הפונקציה המרכזית שמקבלת מילון של שאילתות עם המספר מזהה שלהם. הפונקציה תעבור על כל שאילתה. קודם כל היא תפרסר אותה לterms באותו אופן שפרסרנו את כל הקורפוס . לאחר מכן עבור כל term בשאילתה , הפונקציה תיגש על פי המילון לתיקייה המתאימה בפוסטינג ותקרא את האינפורמציה שצריך לאותו term (כלומר כמות הפעמים שהמילה הופיע בכל הקורפוס, וכמות הפעמים שהיא הופיע בכל מסמך). במקרה וביקשו טיפול סמנטי יתבצע טיפול סמנטי עבור השאילתה. אחרי שהיא אוספת את כל האינפורמציה הנ"ל הפונקציה תדרג את כל המסמכים הרלוונטיים בעזרת מחלקת Ranker עבור כל שאילתה. לבסוף, יווצר מילון עבור כל שאילתה נוכל לראות את 50 המסמכים הכי רלוונטיים , ועבור כל מסמך נוכל לראות את ה-5 ישויות הכי דומיננטיות שלו ביחד עם הדירוג.
- **HashMap<String, HashMap<String, LinkedHashMap<String, Double>>> getDocsAndEntitiesForQuery()**
פונקציה המחזירה את המילון הסופי של השאילות , המסמכים , והישויות.
- **writeQueriesResults()**
פונקציה האחראית על כתיבת התוצאות הסופיות עבור כל שאילתה לדיסק בפורמט המתאים לתוכנת ה-TREC EVAL.
- **HashMap<String,LinkedHashMap<String,Double>> getEntities (HashMap<String, Double> rankedDocs)**
פונקציה האחראית להביא מהזיכרון עבור כל רשימת המסמכים שהיא מקבלת, את חמשת הישויות הכי דומיננטיות לאותו מסמך. הפונקציה מנרמלת את התוצאות באופן הבא:
$Rank = \frac{tf}{maxtf}$ כך שטווח הערכים שנקבל עבור כל ישות הוא בין 0 ל-1.
- **String semantic(String query)**
פונקציה זו מקבלת שאילתה ותמצא עבור כל מילה בשאילתה את המילים הדומות סמנטית לאותה מילה. המילים שהכי דומות סמנטית (שהדירוג שלה קיבל מעל ל-0.9) יצורפו לסוף השאילתה כחלק ממנה. לבסוף השאילתה המעודכנת עם כל המילים החדשות תוחזר.
- **void setAvgLength()**
הפונקציה קוראת מהקובץ בו כתוב האורך הממוצע של כל המסמכים בקורפוס , מאתחלת את המשתנה avgLength כדי שנשתמש בו בהמשך לדירוג המסמכים.

- **HashMap<String, Integer> getLengthofDocs(HashSet<String> docsForQuery)**
הפונקציה מקבלת סט של מספרי מסמכים , ותחזיר מילון בו המפתח הוא מספר המסמך, והערך הוא אורך המסמך – כלומר כמות המילים במסמך כולל כפילויות.

2. **מחלקת Ranker:**
מחלקה זו אחראית על דירוג המסמכים לשאילתה לפי אלגוריתם BM25 עליו נפרט בהמשך.
פונקציות המחלקה הן :
- **Ranker()**
בנאי המחלקה, הבנאי מאתחל את אלגוריתם BM25.
- **HashMap<String,Double> rank(String[] terms,int allDocs,HashMap<String, Integer> idfs, HashMap<String, HashMap<String, Integer>> tf, HashSet<String> docsForQuery, HashMap<String, Integer> docLengths, double avgLength)**
הפוקנציה המרכזית שמדרגת את המסמכים. הפונקציה מקבלת מערך של המילים של השאילתה , כמות המסמכים בקורפוס, עבור כל מילה כמה היא הופיעה בקורפוס (idf) וכמה היא הופיע בכל מסמך רלוונטי שנרצה לדרג (tf) , רשימה של כל המסמכים הרלוונטים ואורכיהם , ואת האורך הממוצע למסמך בקורפוס.
עבור כל מסמך הפוקנציה תדרג את המסמך בהתאם לשאילתה עם האלגוריתם של BM25 ולבסוף תחזיר מילון עם 50 המסמכים הכי רלוונטיים ודירוגם.
- **HashMap<String, Double> getTop50(HashMap<String, Double> allRankings)**
הפונקציה מקבלת רשימה של מסמכים עם דירוגיהם , ממיינת אותם לפי הדירוג מהגבוה לנמוך , ומחזירה את 50 המסמכים הראשונים – כלומר בעלי הדירוג הכי גבוה.

3. **מחלקת BM25:**
מחלקה הממשת את RelevanceRanker, מטרת מחלקה זו היא לחשב את BM25.
החישוב מתבצע לפי הנוסחה הבאה :

$$rank(D,Q) = \sum_{qi=1}\left(\frac{f(q_i,D)*(k+1)}{(f(q_i,D)+k(1-b+b*\frac{|d|}{avgd})} + delta\right) * \log\left(\frac{N+n(q_i)+0.5}{n(q_i)+0.5}\right)$$

כאשר :
f(q,D) - תדירות המילה במסמך (tf של המילה במסמך)
|d|- אורך המסמך
Avgd- אורך ממוצע כלל המסמכים במאגר.
N- כמות המסמכים במאגר.
n(qi)- בכמה מסמכים במאגר המילה מופיעה (df עבורה ה- term)
b,k,delta- קבועים כך שהגדרנו k=2 ,b=0.5, delta =1.
פרמטרים אלו נבחרו לאחר ניסויים רבים שערכנו כיוון שהם נתנו לנו את התוצאות הטובות ביותר.
פונקציות המחלקה הן :
- **double score(double freq, int docSize, double avgDocSize, long N, long n)**
פונקציה המדרגת עבור מילה אחת מהשאילתה את רלוונטיות המסמך. הפונקציה מקבלת את תדירות המילה במסמך, גודל המסמך, האורך הממוצע של המסמכים בקורפוס, כמות המסמכים בקורפוס, וכמות הפעמים שהמילה הופיעה בקורפוס. הפונקציה מדרגת את המסמך לפי הפונקציה לעיל ומחזירה את הדירוג.

- **double rank(int N, int docSize,double avgDocSize, String[] terms, int[] tf, int[] n)**

פונקציה המקבלת שאילתה ומסמך , תדירות המילים במסמך, גודל המסמך, האורך הממוצע של המסמכים בקורפוס, כמות המסמכים בקורפוס, וכמות הפעמים שהמילים בשאילתה הופיעו בקורפוס , שולחת לפונקציה Score כל מילה מהשאילתה עם המסמך והנתונים הרלוונטיים , ומחברת את כל הדירוגים עבור כל מילה שהפונקציה תחזיר. לבסוף נקבל דירוג למסמך לכל השאילתה והפונקציה תחזיר דירוג זה.

4. **מחלקת RelevanceRanker:**

מחלקת Interface לדירוגים המחייבת כל מחלקה שמממשת פונקציית דירוג לממש את הפונקציה double rank(int N, int docSize,double avgDocSize, String[] terms, int[] tf, int[] n)

- **להלן השינויים שביצענו במחלקות מחלק א':**

1. **מחלקת Parser:**

במחלקה זו הוספנו עוד פונקציה הזה לחלוטין לפונקציית ה-parse רק במקום לשלוח ל-indexer את המילים המפורסרות הפונקציה מקבלת שאילתה והאם צריך לעשות לה Stemming ויוצרת שאילתה חדשה עם המילים המפורסרות ולבסוף מחזירה אותה.

String parseQuery(String text,boolean isStem)

2. **מחלקת Document:**

במחלקה זו שינינו את פרמטר האורך. כלומר כעת , במקום לשמור את כמות המילים הייחודיות במסמך, המחלקה שומרת את כמות המילים כולל כפילויות. ביצענו שינוי זה כיוון שראינו שזה נתון נחוץ לחישוב דירוג ה-BM25 לכן אנו צריכים לשמור אותו עבור כל מסמך בקבצי הפוסטינג.

3. **מחלקת Indexer:**

במחלקה זו הוספנו מילון אשר מחזיק עבור כל מסמך את אורכו. הוספנו נתון זה כיוון שאנו צריכים לחשב את האורך הממוצע של כל המסמכים בקורפוס לאחר שפרפסנו אותם. את הממוצע נשמור בתוך תיקיית קבצי הפוסטינג , תהיה תיקייה avg שזה יהיה קובץ טקסט שכתוב בו הממוצע. לאחר מכן בשלב החיפוש נוכל להוציא משם את המידע על הממוצע בשביל ה-BM25.

בנוסף , נשמור מילון שיחזיק עבור כל מסמך את 5 הישויות הכי נפוצות בו וכמות הפעמים שהן הופיעו במסמך. נכתוב את כל הרשימה הזו בתור קובץ פוסטינג נוסף בדיסק כדי להשתמש במידע זה לאחר מכן בחלק ב'. באופן הזה , יהיה לנו את הנתון הזה בצורה הקלה ביותר ומהירה ביותר במקום לשמור את זה בקובץ הפוסטינג המקורי של המסמכים, כיוון שישויות מזוהות רק לאחר שעברנו על כל הקורפוס. לכן העדפנו לשמור את זה בתוך קובץ פוסטינג נוסף במקום לראות איפה שמרנו כל מסמך ואז לגשת לדיסק ולעדכן את הנתון הזה.

הפונקציות שנוספו הן :

- **addEntToEntDocPosting(Term ent, String doc)**

הפונקציה מקבלת ישות ומסמך , בודקת האם הישות היא בין 5 הישויות הכי נפוצות במסמך, ואם כן , מעדכנת את מבנה הנתונים בהתאם.

- **write()**

הפונקציה מופעלת בכל איטרציה (כלומר בכל 10000 מסמכים) , הפונקציה כותבת את כל הנתונים שקיימים בפוסטינג של המסמכים לתיקייה הרלוונטית בדיסק , ואת כל המילים לפוסטינג הזמני לפני המיזוג.

- **writeDocsEnts()**

פונקציה הכותבת לדיסק את המילון בו רשום עבור כל מסמך את 5 הישויות הנפוצות ביותר, פעולה זו לאחר שסיימנו לעבור על כל הקורפוס.

- **writeAverage(String p)**

פונקציה המקבלת נתיב שאליו נרצה לכתוב את הנתון על הממוצע. הפונקציה תחשב לפי רשימת אורכי המסמכים את אורך המסמכים הממוצע בקורפוס (סכום כל האורכים חלקי גודל הרשימה) ותכתוב את הנתון הזה לנתיב שהיא קיבלה בדיסק.

- **HashMap<String,String> getTermDicWithoutUpload()**

פונקציה המחזירה את המילון של המילים לאחר שהוא הועלה כבר לזיכרון.

4. **מחלקת Model:**

הפונקציות שנוספו הן:

- **HashMap<String, HashMap<String, LinkedHashMap<String, Double>>> getAnswers()**

פונקציית מעטפת למתודה Search ב-Searcher

- **writeAns()**

פונקציית מעטפת למתודה writeQueriesResults ב-Searcher

5. **מחלקת Readfile:**

הפונקציות שנוספו הן:

- **Parse getParser()**

פונקציה המחזירה את אובייקט ה-Parse של המחלקה.

6. **מחלקת mainMenuController:**

הפונקציות שנוספו הן:

- **setQueryPane()**

פונקציה המעבירה למסך איפה שניתן להריץ שאילתה. אם המילון לא הועלה לפני שמנסים לעבור למסך הנ"ל תוצג הודעה בהתאם.

- **setQueryPaneBack()**

פונקציה המחזירה את מסך השאילתה חזרה במידה ואנו רוצים להזין שאילתה נוספת.

- **setOptionsPane()**

פונקציה המעבירה למסך הבחירה שבו יש אופציה לראות את התשובות לשאילתה, או לחזור ולהזין שאילתה חדשה.

- **setOptionsPaneBack()**

פונקציה המחזירה את מסך הבחירה

- **setP_Answers()**

פונקציה המעבירה למסך בו ניתן לראות את התשובות לשאילתה שהזנו

- **startSearch()**

פונקציה הנקראת ברגע שלוחצים על כפתור "Run". הפונקציה תבדוק האם הוזנה שאילתה מתוך קובץ טקסט , או בכתיבה חופשית ובהתאם לכך תפעיל את הפונקציה searchQuery במחלקת ViewModel. לאחר מכן יוחלף המסך למסך הבחירות.

- **showAnswers()**

פונקציה התציג תשובות לשאילתה. הפונקציה רואה איזו שאילתה המשתמש בחר מתוך מקבץ השאילתות ולפי זה מראה את התוצאה המתאימה. הפונקציה תראה את 50 המסמכים הרלוונטיים, ביחד עם 5 ישויות והדירוג שלהם עבור כל מסמך.

- **browseQuery()**

פונקציה הנותנת למשתמש לבחור נתיב שממנו הוא יעלה את קובץ השאילתות. הפונקציה תביא להכניס את הנתיב הנבחר רק בתנאי שהקובץ הנבחר הוא קובץ ולא תיקייה.

7. **מחלקת ViewModel:**

הפונקציות שנוספו הן:

- **searchQuery(String query,Boolean isPath, Boolean isStem, Boolean isSemantic,String postPath)**

הפונקציה מקבלת שאילתה, האם היא מתוך קובץ או שלא, האם הופעל Stemming , האם רוצים אחזור סמנטי, והנתיב לקבצי הפוסטינג. אם השאילתה היא מקובץ אז בעצם מה שאנו מקבלים ב-query הוא נתיב, ולכן הפונקציה תשלח את הנתיב למתודה getQueries , שתחזיר את כל השאילתות הקיימות בקובץ. אם השאילתה היא מכתיבה חופשית הפונקציה

נותנת לשאילתה מספר מזהה של "000". לאחר מכן בהתאם לזה תופעל המתודה search של מחלקת Model.

- LinkedHashMap<String,String> getQueries(String path)
הפונקציה מקבלת נתיב לשאילתות, קוראת אותם ועבור כל שאילתה מחברת את השאילתה עצמה מהתגית title לתיאור שלה בתגית description , ואת המספר מזהה שלה מהתגית num. לאחר מכן היא תיצור מילון שהמפתח יהיה המספר המזהה של השאילתה , והערך יהיה השאילתה שהורכבה מהשאילתה עצמה והתיאור.
- HashMap<String, HashMap<String, LinkedHashMap<String,Double>>> getAnswers()
הפונקציה ממירה את התוצאות שמגיעות מה-model כך שהמפתח בהם הוא מספר המזהה של השאילתה, לאותה מילון רק שהמפתח יהיה השאילתה עצמה ותחזיר את המילון הזה.
- writeAns()
פונקציית מעטפת לפונקציה writeAns של המחלקה Model.
- LinkedHashMap<String,String> getQueriesWithNoDesc()
פונקציה המחזירה את השאילתות ללא התיאור שלהם כלומר, רק את הtitle. המפתח במילון שהוחזר תהיה השאילתה ללא תיאור, והערך עם התיאור.

- **<u>אלגוריתמים בהם השתמשנו:</u>**

- <u>אלגוריתם הדירוג:</u>

על מנת להגיע לתוצאות המיטביות ניסינו מספר דברים:

  - ניסינו להשתמש באלגוריתם המקורי הנלמד בכיתה tf*idf כאשר tf אנו היינו מנרמלים עם כמות המופעים של המילה הכי נפוצה במסמך, אך התוצאות היו רחוקות ממשביעות רצון.
  - החלטנו להשתמש בBM25 שלקחו מקוד פתוח מ - https://github.com/haifengl/smile/blob/master/nlp/src/main/java/smile/nlp/relevance/BM25.java בגרסת סמייל. לאחר המון ריצות גילינו כי הפרמטרים המיטביים עבור אלגוריתם זה הם – a=2, b=0.5,delta=1.0.
בנוסף, ניסינו לנרמל בצורות שונות את ה-tf בנוסחה אך התוצאות כמעט ולא השתנו לכן השארנו את הנוסחה בצורתה המקורית.

  - אופן פעולת האלגוריתם:
לכל שאילתה שקיבלנו, ניתחנו אותה ואת הטקסט המתקבל מה- description של השאילתה. כך הרחבנו את השאילתה שלנו לשאילתה שמכילה גם את המילים הכוללות גם בתיאור. כך קיבלנו תוצאות מדויקות יותר גם ללא טיפול סמנטי כיוון שהמילים בתיאור מכוונות למה הייתה כוונת השאילתה. לאחר שהוספנו (במידה וקיים תיאור), נשלח את השאילתה לחיפוש כאשר הניתוח המתבצע על כל מילה בשאילתה (כולל טיפול סמנטי) מתבצע גם על התיאור. לאחר שנדרג את המסמכים הרלוונטיים לפי הנוסחה שתיארנו למעלה, נקבל את 50 המסמכים הכי רלוונטיים לאותה שאילתה.

- <u>אלגוריתם למציאת 5 הישויות הדומיננטיות במסמך:</u>

  - הוספנו קובץ פוסטינג חדש כך שבחלק א' אנו מגלים מה 5 הישויות הדומיננטיות ביותר עבור כל מסמך, וכמות המופעים שלה. כאשר המשתמש ירצה לראות את תוצאות השאילתות ב-gui הוא יבחר בשאילתה הרצויה ויוצג לו 50 המסמכים הכי רלוונטיים לאותה שאילתה ואת 5 הישויות עם דירוגם לכל מסמך.
בחלק א' שמרנו את הנתון עבור כל מסמך של התדירות של המילה הכי נפוצה.
כאמור, נשתמש בנתון זה על מנת לנרמל את דירוג הישות כך שיתקיים:
$$0 < rank \leq 1$$
דוגמאות ליישויות בקבצים:

- FT923-11890 ---> SOUTH OCTOBER 0.1428,SOUTH ATLANTIC 0.1428, FALKLAND ISLANDS 0.2857,
- FBIS4-49987 ---> BFN BEIJING 0.2,CHINESE SPOKESMAN 0.2,BFN MAY 0.2, CHINESE MINISTRY 0.2,CHINESE FOREIGN 0.2

- <u>אלגוריתם לשיפור סמנטי</u>
  - האלגוריתם לשיפור הסמנטי עובד על קוד פתוח https://github.com/medallia/Word2VecJava . אלגוריתם זה הוא אלגוריתם word2Vec medallia שמשתמש במודל שכבר מאומן על מנת למצוא את המילים הכי דומות סמנטית למילים שבשאילתה. האלגוריתם מקבל את השאילתה, ועבור כל מילה בשאילתה יחבר לשאילתה בסופה את המילה הכי דומה לה כאשר זאת לא אותה מילה (כיוון שלרוב המילה שמקבלת את הדירוג הכי גבוה היא המילה עצמה) , וגם דירוג המילה הוא מעל 0.9 .
    לבסוף תוחזר השאילתה בייחד עם כל המילים הדומות סמנטית ועונות על הדרישות שהזכרנו.

    נראה בדוגמא הבאה כיצד האלגוריתם הסמנטי משפר את התוצאות:
    נסתכל על שאילתה מספר 367, נראה כי ללא טיפול סמנטי נקבל:

Precision=14/50
Recall=14/185
Precision@5 = 0.2
Precision@10=0.2
Precision@15=0.3333
Precision@30=0.2333
MAP=0.0248

אך עם טיפול סמנטי נקבל:

Precision=16/50
Recall=16/185
Precision@5=0.2
Precision@10=0.2
Precision@15=0.3333
Precision@30=0.35
MAP=0.0278

כלומר , ללא הטיפול הסמנטי היינו מאבדים את שני המסמכים הנ"ל.

- **<u>הסבר על שימוש בנתוני קבצי הפוסטינג התומכים באלגוריתמים:</u>**

- קובץ הפוסטינג:
  שמרנו עבור כל מילה את כמות הפעמים שהיא מופיע בכל הקורפוס – נשתמש בנתון זה כ-idf בנוסחת ה-BM25 .
  עבור כל מסמך שמרנו את כמות הפעמים שהמילה מופיעה בו – נשתמש בנתון זה כ-tf בנוסחת ה-BM25 .

- קובץ פוסטינג של הישויות:
  שמרנו עבור כל מסמך את 5 הישויות הכי נפוצות שלו וכמות המופעים שלהם – נשתמש בנתון זה כדי להראות בקלות ל-50 מסמכים שנאחזר את הישויות שלהם.

- קובץ פוסטינג המסמכים :
שמרנו עבור כל מסמך את האורך שלו – נשתמש בנתון זה בנוסחת ה-BM25.
שמרנו עבור כל מסמך את תדירות המילה הכי נפוצה בו – נשתמש בנתון זה על מנת לדרג את הישויות.
בנוסף נשתמש בנתון של האורך הממוצע שנשמור בדיסק בנוסחת ה-BM25.

מרינה בוטביניק 317522605 , דור אלקבץ 2040

- **הרצה וביצועי המנוע:**

אחזור ללא stemming ללא שיפור סמנטי:

MAP	Time in Millis	percision@50	percision@30	percision@15	percision@10	percision@5	Recall	Precision	מילות השאילתה	מספר השאילתה
0.1857	4873	22/50	0.4	0.4	0.2	0.2	22/48	22/50	Falkland petroleum exploration	351
0.0117	1285	9/50	0.2667	0.1333	0.2	0.2	9/246	9/50	British Chunnel impact	352
0.2092	609	24/50	0.4667	0.4667	0.3	0.2	24/51	24/50	blood-alcohol fatalities	358
0.0311	898	6/50	0.1667	0.0667	0.0	0.0	6/28	6/50	mutual fund predictors	359
0.0649	145	9/50	0.2	0.2	0.3	0.4	9/39	9/50	human smuggling	362
0.0248	838	14/50	0.2333	0.3333	0.2	0.2	14/185	14/50	piracy	367
0.0463	1198	4/50	0.1333	0.2	0.1	0.0	4/16	4/50	encryption equipment export	373
0.0130	946	9/50	0.2	0.1333	0.2	0.4	9/204	9/50	Nobel prize winners	374
0.0695	601	12/50	0.2333	0.0667	0.1	0.0	12/36	12/50	cigar smoking	377
0.1683	372	6/50	0.1667	0.2	0.1	0.0	6/7	6/50	obesity medical treatment	380
0.0698	904	13/50	0.2	0.2667	0.4	0.2	13/51	13/50	space station moon	384
0.0286	932	12/50	0.2	0.0667	0.0	0.0	12/85	12/50	hybrid fuel cars	385
0.0517	701	13/50	0.3	0.2667	0.2	0.4	13/73	13/50	radioactive waste	387
0.0911	725	10/50	0.3	0.4	0.4	0.6	10/50	10/50	organic soil enhancement	388
0.0526	1065	14/50	0.3333	0.4	0.3	0.6	14/122	14/50	orphan drugs	390
0.0746	**16.09 sec**	**177 /750**	**0.2533**	**0.24**	**0.2**	**0.2267**	**177/1241**	**177/750**		**סה"כ**

מרינה בוטביניק 317522605 , דור אלקבץ 2040

אחזור ללא stemming ועם שיפור סמנטי:

MAP	Time in Millis	percision@50	percision@30	percision@15	percision@10	percision@5	Recall	Precision	מילות השאילתה	מספר השאילתה
0.1703	4620	19/50	0.5	0.4667	0.3	0.2	19/48	19/50	Falkland petroleum exploration	351
0.0082	2643	8/50	0.2333	0.1333	0.2	0.2	8/246	8/50	British Chunnel impact	352
0.2115	1161	24/50	0.4667	0.4	0.2	0.4	24/51	24/50	blood-alcohol fatalities	358
0.0034	1495	2/50	0	0	0	0	2/28	2/50	mutual fund predictors	359
0.0247	1162	7/50	0.1333	0.0667	0.1	0	7/39	7/50	human smuggling	362
0.0278	1974	16/50	0.3	0.3333	0.2	0.2	16/185	16/50	piracy	367
0.0729	2243	3/50	0.1	0.2	0.3	0.2	3/16	3/50	encryption equipment export	373
0.0216	1468	14/50	0.3333	0.2	0.3	0.4	14/204	14/50	Nobel prize winners	374
0.0756	1131	12/50	0.2	0.1333	0.2	0	12/36	12/50	cigar smoking	377
0.0042	1499	1/50	0	0	0	0	1/7	1/50	obesity medical treatment	380
0.0582	1660	9/50	0.1333	0.2667	0.3	0.4	9/51	9/50	space station moon	384
0.0326	2139	9/50	0.2	0.2667	0.4	0.4	9/85	9/50	hybrid fuel cars	385
0.0602	1505	14/50	0.3	0.2667	0.2	0.4	14/73	14/50	radioactive waste	387
0.0505	1532	6/50	0.1667	0.2667	0.3	0.4	6/50	6/50	organic soil enhancement	388
0.0376	2961	11/50	0.3	0.3333	0.4	0.4	11/50	11/122	orphan drugs	390
0.0573	**29.19 sec**	**155/750**	**0.2244**	**0.2222**	**0.2267**	**0.24**	**155/1241**	**155/750**		**סה"כ**

מרינה בוטביניק 317522605 , דור אלקבץ 2040

אחזור עם stemming וללא שיפור סמנטי:

MAP	Time in Millis	percision@50	percision@30	percision@15	percision@10	percision@5	Recall	Precision	מילות השאילתה	מספר השאילתה
0.0444	1043	12/50	0.1667	0.0667	0.1	0	12/48	12/50	Falkland petroleum exploration	351
0.0109	1109	10/50	0.3	0.1333	0.2	0.2	10/246	10/50	British Chunnel impact	352
0.1313	639	17/50	0.3333	0.4667	0.4	0.2	17/51	17/50	blood-alcohol fatalities	358
0.0283	937	4/50	0.1333	0.2	0.2	0	4/28	4/50	mutual fund predictors	359
0	113	0/50	0	0	0	0	0/39	0/50	human smuggling	362
0	587	0/50	0	0	0	0	0/39	0/50	piracy	367
0.0089	907	1/50	0.0333	0.0667	0.1	0	1/16	1/50	encryption equipment export	373
0.1215	936	32/50	0.6	0.8667	0.9	1	32/204	32/50	Nobel prize winners	374
0.1512	556	18/50	0.3	0.2	0.3	0	18/36	18/50	cigar smoking	377
0	445	0/50	0	0	0	0	0/7	0/50	obesity medical treatment	380
0.0396	929	9/50	0.1333	0.2	0.3	0.4	9/51	9/50	space station moon	384
0.0219	798	10/50	0.2	0.0667	0	0	10/85	10/50	hybrid fuel cars	385
0.0066	576	5/50	0.0333	0.0667	0.1	0	5/73	5/50	radioactive waste	387
0.0051	506	2/50	0.0667	0.0667	0.1	0	2/50	2/50	organic soil enhancement	388
0.0137	867	7/50	0.2	0.2667	0.2	0.2	7/122	7/50	orphan drugs	390
0.0389	**10.94 sec**	**127/750**	**0.1667**	**0.1778**	**0.1933**	**0.1333**	**127/1241**	**127/750**		**סה"כ**

מרינה בוטביניק 317522605 , דור אלקבץ 2040

אחזור עם Stemming ועם טיפול סמנטי

MAP	Time in Millis	percision@50	percision@30	percision@15	percision@10	percision@5	Recall	Precision	מילות השאילתה	מספר השאילתה
0.1083	4055	19/50	0.2667	0.0667	0.1	0	19/48	19/50	Falkland petroleum exploration	351
0.0091	2473	9/50	0.2667	0.1333	0.2	0.2	9/246	9/50	British Chunnel impact	352
0.1335	1228	17/50	0.3333	0.4667	0.4	0.2	17/51	17/50	blood-alcohol fatalities	358
0.0071	1592	1/50	0.0333	0.0667	0.1	0.2	1/28	1/50	mutual fund predictors	359
0	1171	0/50	0	0	0	0	0/39	0/50	human smuggling	362
0	1411	0/50	0	0	0	0	0/185	0/50	piracy	367
0.0052	2042	1/50	0.0333	0.0667	0	0	1/16	1/50	encryption equipment export	373
0.1296	1502	33/50	0.7	0.8667	0.9	1	33/204	33/50	Nobel prize winners	374
0.1385	1138	17/50	0.2667	0.2	0.3	0.2	17/36	17/50	cigar smoking	377
0	1772	0/50	0	0	0	0	0/7	0/50	obesity medical treatment	380
0.0285	1841	8/50	0.1333	0.2	0.2	0.2	8/51	8/50	space station moon	384
0.0208	2006	9/50	0.2	0.2	0.2	0	9/85	9/50	hybrid fuel cars	385
0.0060	1091	5/50	0.0333	0.0667	0.1	0	5/73	5/50	radioactive waste	387
0.0055	1209	2/50	0.0667	0.1333	0.1	0	2/50	2/50	organic soil enhancement	388
0.0089	2398	5/50	0.1667	0.2	0.1	0.2	5/122	5/50	orphan drugs	390
0.0402	**26.92 sec**	**126/750**	**0.1667**	**0.1778**	**0.18**	**0.1467**	**126/1241**	**126/750**		**סה"כ**

סיכום

לסיכום, הפרויקט דרש מאיתנו השקעה רבה וזמן רב , אך ללמדנו ממנו רבות, ושיפרנו את כישורי התכנות שלנו.

הבעיות שנתקלנו בהם במהלך ביצוע הפרויקט:

אחת הבעיות המרכזיות שנתקלנו בהם בחלק א' היא אופן השמירה של קבצי הפוסטינג. תחילה, חשבנו לשמור את כל הנתונים עבור כל אחת מהמילים בעזרת תגיות ספריית JSOUP. חשבנו , שבאופן כזה יהיה לנו הרבה יותר קל בחלק ב' לגשת לכל term ולקרוא את הנתונים ששמרנו עליו. באופן הזה , לא נצטרך לעבור שורה שורה ולחפש את המילה (גם אם נקבל את הנתיב לתיקייה המסויימת בקובץ הפוסטינג בה נמצאת המילה) אלא נקבל ישר את המידע. בשלב מסויים, גילינו כי על אף הצורה הנוחה שבה קבצי הפוסטינג שלנו נכתבים ישנן 2 בעיות בצורת שמירה זו :

1. קבצי הפוסטינג יוצאים מאוד גדולים (פי 9) מגודל הקורפוס הרגיל ולכן כאשר התוכנה מנסה לעלות קובץ כזה , היא לא מעלה לזכרון שורה שורה אלא מעלה את כל הקובץ ישר על מנת לפרסר אותו לפי תגיות , וגודל הקובץ יותר גדול מגודל ה-heap והתוכנה קורסת.
2. לפרסר קובץ פוסטינג לפי תגיות ולאחזר תגית אחת לוקח הרבה יותר זמן מאשר לכתוב קובץ פוסטינג בצורת שורות טקסט רגילות ולקרוא אותם שורה שורה.

לכן , החלפנו את כל אופן שמירת קבצי הפוסטינג שלנו כך שישמרו בצורת קובץ טקסט רגיל כך שבכל שורה מופיעה מילה חדשה והנתונים שאנו שומרים עליה.

באופן כללי , גילינו כי שימוש נכון במבני נתונים, ושמירת קבצים בצורה יעילה משפיע המון על זמני הריצה. הגשנו את חלק א' כאשר הגענו לזמן ריצה רק של שעה ועשרים שזהו זמן ריצה ארוך מאוד. לאחר מכן אחרי ראינו שהבעיה הייתה באופן המיזוג שעשינו ובאופן השימוש בתכנות מקבילי, אחרי שתיקנו פונקציה זו, ושינינו את המקביליות לצורה הרבה יותר יעילה **זמן הריצה ירד ל-25 דקות**.

בנוסף, בחלק א' תחילה כתבנו את המילונים של המילים והמסמכים לזיכרון בצורה של אובייקט מסוג ser כיוון שהכתיבה שלנו נעשית בצורה מאוד מהירה. אך לאחר מכן שינינו גם את זה לכתיבה של קובץ טקסט רגיל כיוון שקריאת קובץ טקסט (אפילו של מעל מיליון שורות!) הרבה יותר מהירה מהעלאת קובץ ser לזיכרון.

הבעיה העיקרית שנתקלנו בה בחלק ב' היא עם הטיפול הסמנטי, כיוון שלא ניתן היה לגשת לאינטרנט ולהשתמש במודל מאומן, היינו צריכים למצוא מודל שכבר יש בו מילים מאומנות ולהשתמש בו. בנוסף , לא יכולנו להשתמש בLSI למרות שקיימות ספריות לכך ב-java כיוון שהמטריצה שיצאה לנו שקלה מעל ה-TB400. המודל שהוספנו בסוף מאומן על כמות לא גדולה של מילים אך עובד בצורה טובה ומוצא מילים בעלות אותה משמעות.

לבסוף , היינו מקדישים יותר זמן לתכנון העבודה שלנו, והלבנה למה נחוץ כל דבר שאנו כותבים להמשך העבודה , וכך היינו מייעלים את העבודה שלנו רבות. בנוסף היינו מוסיפים התייחסות למיקומי המילים במסמך ששמרנו (positions) כדי לשפר את איכות אחזור המסמכים, ומוסיפים דירוג משקלים למילים כך שמילים שנמצאות בשאילתה יקבלו דירוג גבוה לעומת מילים בתיאור שיקבלו משקל נמוך יותר.