**מנוע חיפוש חלק א'- דו"ח**

**שאלה 1:**תיאור התוכנית:

התוכנית שבנינו יוצרת אינדקס הופכי בעבור מאגר מסמכים, מה שמאפשר אחזור מידע ומענה יעיל על שאילתות.

המחלקות:

1. **מחלקת ReadFile:**

הפרמטרים:

* pathSource- הנתיב שבו נמצא מאגר המסמכים.
* pathForPosting- הנתיב בו יכתבו בסופה של התוכנית האינדקס ההופכי של המאגר.

השדות:

* do\_Stemming: מחזיק את בחירת המשתמש לגבי ביצוע Stemming או לא.
* pathToRead: הכתובת בה נמצא מאגר המסמכים.
* textCount: משתנה ששומר את כמות המסמכים במאגר.
* Indexer: משתנה מהטיפוס Indexer.
* fileIndex: מילון המסמכים.
* citiesList: רשימת כל הערים מהמאגר.
* langList: רשימת כל השפות מהמאגר.
* stop\_words: Set המכיל את כל מילות ה- stop words שעלינו להתעלם מהן.
* parser: משתנה מהטיפוס Parse.

המטודות:

* **readAllFiles():** זו הפונקציה המרכזית שתפקידה הינה מעבר על כלל הקבצים במאגר, קריאתם ופירוקם למסמכים השונים, שליחת כל מסמך ל- Parse ומתחיל בפעולת האינדוקס (בניית אינדקס ה- Terms, המסמכים והערים).
* **BreakToPart(file, openLable, closeLable, subLabel, languge, subCity):**

הפרמטרים:

file- הקובץ הנוכחי.

openLable- תגית המסמלת על פתיחת הקטע הרצוי לקריאה (<TEXT>) .

closeLable- תגית המסמלת על סגירת הקטע הרצוי לקריאה (</TEXT>).

subLabel – קריאת חלק משורה שנמצא אחרי אותו משתנה באותה השורה (לדוגמא: קריאת שם המסמך בשורה שבה מופיעה התגית <DOCNO>) .

Languge- התגית שבעזרתה נקראת השפה של המסמך (<F P=105>).

subCity- התגית שבעזרתה נקראת העיר של המסמך (<F P=104>).

תפקידה של פונקציה זו הוא לחתוך את החלקים הרצויים (שם, שפה, עיר והתוכן של מסמך מסוים) מקובץ בעל טקסטים רבים. הפונקציה פועלת בלולאה על כל שורות הקובץ ושומרת כל תוכן של מסמך במשתנה אותו היא שולחת לParser ולאחר מכן לאינדוקס ראשוני, בנוסף היא שומרת בעבור כל מסמך את הפרטים הרלוונטים עליו.

* **readAllCities(file, subCity):**

הפרמטרים:

file: הקובץ הנוכחי

subCity: התגית שבעזרתה נקראת העיר של המסמך (<F P=104>).

תפקידה של הפונקציה הוא מעבר על כל מאגר המסמכים ויצירת רשימה של כלל הערים שלהן יש מסמכים במאגר.

* **writeAllDictToDisk():**

פונקציה זו נקראת בסיום תהליך האינדוקס ותפקידה הינה כתיבת שלושת האינדקסים מזיכרון התוכנית לדיסק (מילון ה-Terms, מילון המסמכים ומילון הערים).

1. **מחלקת Parse:**

הפרמטר: stopWords אותו מקבלים מהמחלקה ReadFileמכיל Set של כל מילות ה- stop words הנשמר כשדה במחלקה.

הפונקציות:

* parseText(text):

הפרמטר text הינו משתנה מסוג String המכיל את הטקסט עליו צריך לבצע את ה-parse.

בתחילת הפונקציה נמחקים כל הסימנים ב- String בהם אין צורך (כמו: \*~@#^&;:!?\',(){} ועוד), לאחר מכן מופרדים כל הביטויים ב- String ע"פ " " (רווח) ו- \n (ירידת שורה). בלולאה על כל הביטויים ברשימה נשמרים ה- Tokens ע"פ החוקים הנתונים בעבודה ברשימה נוספת החוזרת בסוף הפונקציה.

* monthToNum(month):

פונקציה המקבלת שם של חודש מסוים (month) ומחזירה את מספר החודש המתאים.

1. **מחלקת Indexer:**

הפרמטרים:

* pathToWrite: הכתובת אליה לכתוב את האינדקס.

השדות:

* baseDict: המילון של כל ה- Terms במאגר. מילון זה מחזיק בעבור כל ביטוי רשימה מהסוג [offset, size, tf]

Offset- המיקום ממנו להתחיל לקרוא את המידע על הביטוי בקובץ ה- Posting.

Size- מחזיק את גודל המידע בנמצא בעבור הביטוי.

tf- כמות המופעים של הביטוי בכל המאגר.

* tmpDict: רשימה זמנית של מילונים אליה מתווספים מילונים של מסמכים בעבור כל קובץ במאגר.
* dictOfFiles: רשימה זמנית של מילונים אליה מתווספים מילונים של כמה קבצים עד לביצוע bigMerge.
* tokenList: רשימה של כל הביטויים שמגיעים מה-Parser.
* numCount: מילון שמחזיק בעבור כל ביטוי מרשימת הביטויים הנוכחים את כמות הפעמים שהביטוי מופיע בה. (TF במסמך כלשהו)
* citiesIndex: מילון הערים.
* maxTF: משתנה המחזיק את המילה המופיעה הכי הרבה פעמים בטקסט הנוכחי.

המטודות:

* **add(fileName):**

הפרמטר:

fileName: שם המסמך שהגיע מה- parser.

הפונקציה עוברת על כל הביטויים שהגיעו מה- parser ושומרת אותם במילון baseDict בצורה הנכונה בעבור כל המאגר (מבחינת אותיות גדולות וקטנות) ובמילון זמני עבור המסמך הנוכחי, ואת המילון הזה מוסיפה לרשימת המילונים הזמניים tmpDict.

* **merge():**

הפונקציה נקראת בסוף כל קובץ ומאחדת את כל המילונים של המסמכים השונים הנמצאים באותו קובץ (ב- tmpDict) למילון אחד מעודכן ואותו מוסיפה ל- dictOfFiles.

* **bigMerge(num):**

הפרמטר:

num: מספר המייצג את כמות הקבצים שנקראו עד כה מהמאגר.

הפונקציה נקראת כל כמות מסוימת של קבצים (אצלנו כל 600 קבצים). במהלך הפונקציה מתאחדים המילונים של כל הקבצים למילון אחד מעודכן, ולאחר מכן מתבצע מעבר על כל המילון שנוצר ומיון המפתחות לפי סדר לקסיקוגרפי (A-Z ו- 0-9) ובעבור כל אות/ספרה נכתב לדיסק קובץ posting מסוג Ujson המכיל את כל הביטויים והמידע עליהם המופיעים במילון.

* **finalMerge():**

לאחר הביצוע האחרון של הפונקציה bigMerge,נקראת הפונקציה הזו ובה נוצרת רשימה ממוינת של כל הקבצים הנמצאים בתיקיה אליה נכתבים קבצי ה- Postings. בעבור כל הקבצים הנמצאים ברשימה ושמם מתחיל בתחילית מסוימת (A-Z או- 0-9) הפונקציה מאחדת אותם למילון אחד מעודכן ומוחקת אותם מהתיקייה. המילון שנוצר נכתב אל הדיסק כקובץ מסוג Ujson.

* **indexMaker():**

לאחר הפונקציה finalMerge ולאחר שבתיקייה של קבצי ה- Postings ישנם קבצים בעבור כל אות/ספרה תחילית של ביטוי הנמצא במאגר פונקציה זו כותבת את התוכן של כל הקבצים הללו אל קובץ טקסט (finalIndex) ומעדכנת במילון ה- baseDict במקום המתאים את ה- Offset וה- Size של הביטוי הנוכחי בהתאם למיקום שבו נכתב המידע בקובץ ה- Posting הסופי (finalIndex) וגודל המידע שנכתב.

* **readFromFile(token):**

הפרמטר:

token: הביטוי עליו נרצה לקרוא את המידע מקובץ ה- Posting הסופי (finalIndex).

הפונקציה קוראת את ה- Offset וה- Size של הביטוי ממילון ה- baseDict ובעזרתם קוראת מ- finalIndex מהמיקום שהוא ה- Offset (בעזרת פונקציית seek(offset)) Size בייטים. בעזרת פונקציית הקריאה אשר קוראת כמות בייטים מסוימת (read(size)).

* **createCityDict(citiesList):**

הפרמטר:

citiesList: רשימה של כל הערים הנמצאות במאגר.

בראשית הפונקציה נשלחת בקשה לשאילתה מה- API שניתן בהגדרת העבודה, לאחר מכן נשמר מילון ובו כל הערים מה-API כאשר המפתחות הן הערים והערכים הם רשימה שבה ישנם: שם המדינה בה נמצאת העיר, המטבע של המדינה והכמות האוכלוסייה בעיר. בעבור כל עיר ברשימת הערים נשמר במילון הערים המידע שמגיע מה- API ובנוסף רשימה של כל המסמכים בהם מופיעה העיר, כמות הפעמים שהעיר מופיע במסמך והמיקומים במסמך.

1. **engineGUI:**

מחלקת ה- GUI של המנוע, מפעילה ומציגה את ממשק המשתמש בעבור מנוע החיפוש עם כלל הכפתורים והדרישות מהגדרת העבודה.

התמודדות עם מגבלת הזיכרון:

בתהליך האינדוקס קבענו שלאחר קריאת כמות מסוימת של קבצים נעצור את הקריאה ונשמור את כל המידע שאגרנו במילונים עד כה בדיסק בצורה של קבצי Ujson הממוינים לפי תחיליות הביטויים (A-Z או- 0-9). בכך אין אנו מחזיקים באף שלב של התוכנית את כל המידע על כל המאגר בזיכרון התוכנית אלא רק חלקים שלו. בסוף קריאת כל המסמכים ועיבודם אנו קוראים בכל פעם קובץ Ujson המכיל מילון של תחילית מסוימת ואותו כותבים לקובץ ה- Posting הסופי שהינו קובץ TXT. בעת שנרצה לקרוא מידע על ביטוי מסוים נשתמש בנתונים (offset ו-size) ששמרנו על אותו ביטוי ממילון ה- baseDict ונקרא ממיקום מסוים offset בקובץ מספר נתון size של בייטים בלבד, ובכך לא נצטרך לקרוא את כל הקובץ אל הזיכרון של התוכנית ורק אז לחפש את המידע הרצוי.

פעולות שנקטנו ע"מ להגיע לזמן ריצה מיטבי:

בראשית, הבנו שחלק הארי בזמן הריצה הוא בגישות לדיסק (קריאה וכתיבה) ולכן הפחתנו בפתיחה וסגירה של קבצים כמה פעמים. בנוסף במחלקת **ReadFile** קראנו על כל קובץ בצורה רציפה ותוך כדי הקריאה שלחנו אל המחלקות האחרות (Parse ו – indexer) את החלקים הרצויים בלבד לאותה מחלקה, כך אין אנו שומרים מבני נתונים גדולים וארוכים של טקסט.

במחלקה **Parse** המעטנו שימוש בפונקציות שאינן inPlace בכל הנוגע לתמרון של strings לדוגמא replace, קבענו את סדר התנאים לפי תדירות התאמתם לביטויים השונים ובכך צמצמנו את כמות הבדיקות של תנאים במחלקה זו.

במחלקה **Indexer** בה נעשות רוב הפעולות מול הדיסק שמרנו כמות מסוימת (לא קטנה במיוחד) של מידע בזיכרון ורק לאחר שהגענו לכמות זו כתבנו בצורה מסודרת את הנתונים לדיסק ולא בעבור כל קובץ או ביטוי, מה שצמצם משמעותית את הגישות לדיסק, בנוסף כל הקבצים הזמניים שנשמרו או נקראו מהדיסק היו קבצי UltraJson שהוא אובייקט מהיר ויעיל מאוד לכתיבה וקריאה של אובייקטים כמו מילון. במקום מעבר על רשימות מסוימות במחלקה זו בחרנו לעבור על ה- Set של אותן רשימות.

אופן שמירת קבצי Postings בדיסק:

לאחר סיום ריצת התוכנית נשמרים בזיכרון:

1. קובץ Posting בשם finalIndex שהוא קובץ טקסט (.txt) המכיל את כל המידע על כל הביטויים הקיימים במאגר המסמכים ברצף, ללא הפרדה בין ביטויים.
2. מילון הביטויים הסופי בשם baseDict שהוא קובץ מסוג Ujson (.ujson) המכיל את הנתונים על המידע והמיקום שלו בקובץ ה- Posting. המילון בנוי כך:

Term: [offset, size, num of occurrences in all the corpus, num of docs]

1. מילון המסמכים בשם fileIndex שהוא קובץ מסוג Ujson (.ujson) המכיל את כל המידע עבור קובץ מסוים.

המילון בנוי כך:

Doc name: [max\_tf, num of distinct terms, city of doc, language]

1. מילון הערים בשם citiesIndex שהוא קובץ מסוג Ujson (.ujson) המכיל את כל המידע על הערים המופיעות במאגר המסמכים. המילון בנוי כך:

City: [Country, currencies, population, docs containing the city, num of occurrences and locations]

בחרנו לעבוד עם קבצי Ujson מכיוון שהם אובייקטים יעילים לכתיבה וקריאה של מבני נתונים (כמו מילון) לדיסק.

פרטי אינפורמציה שהוספנו:

1. בעבור כל מסמך במאגר שמרנו בנוסף לעיר המתאימה גם את השפה המתאימה למסמך בהתאם לתגית השפה. ניעזר בכך לדוגמא ביצירת רשימת כל שפות המסמכים והצגתן לפי דרישת המשתמש (כנדרש בהגדרת העבודה).
2. בעבור כל ביטוי במילון הביטויים (baseDict) שמרנו בנוסף את כמות הפעמים שהביטוי מופיע בכל מאגר המסמכים. ניעזר בכך בהמשך בחישובים סטטיסטיים הקשורים לאחזור מידע בעבור שאילתות לדוגמא במידה ונצטרך לחשב את ההחלקה בשימוש ב- The query likelihood model ולשם כך נצטרך לחשב את (הסתברות הופעת הביטוי במאגר).

חוקים שהוספנו:

1. תמיכה במספרים שלמים שליליים (לדוגמא 1-, 180-). במילון הביטויים ישמרו המספר השלילי וגם המספר ללא סימן המינוס ע"מ להחזיר תוצאות בעבור שני המספרים.

דוגמאות מהמאגר:

1. במסמך מספר: FBIS3-35 נמצאת השורה "-18 degrees C" ובמילון הביטויים נשמרו הביטויים: "18-", "18", "degrees" ו- "C".
2. במסמך מספר: FBIS3-67 נמצא הביטוי "102-" ובמילון נשמרים הביטויים: "102-" ו- "102"
3. תמיכה בביטויי kg שונים. במילון הביטויים יישמרו גם כמספר עצמו, וגם כ-term בפני עצמו מהצורה: "# kg"  
   דוגמאות מהמאגר:  
   א. במסמך מספר: FBIS3-1 נמצא הביטוי "1000 kgs" ובמילון יישמרו הביטויים: "1000" ו-"1000 kg"

מידע נוסף:  
לא השתמשנו בקוד חיצוני מלבד ספריות שונות במהלך התוכנית (לדוגמא: PoerterStemmer, ujson ועוד..)

**שאלה 2:**  
1. במאגר כולו, לפני ביצוע Stemming קיימים כ- 917,636  
2. במאגר כולו, לאחר ביצוע Stemming קיימים כ- 876,661  
3. מאגר כולו, חישבנו כי קיימים 57,953 מספרים שונים (התייחסנו למספרים "טהורים": שלמים ועשרוניים בלבד)  
4. מספר המדינות שקיימות במאגר: 213 מדינות  
5. מספר הערים הקיימות במאגר: 492 , מתוכן 385 אינן ערי בירה.  
6. הקובץ בו נמצאת המילה הנפוצה ביותר הוא - FB396232 ושמה של העיר הכי נפוצה הינה: MOSCOW  
והיא נמצאת במיקומים (אינדקסים): 18, 37, 68, 98, 181, 717, 746, 782, 814, 815, 931, 1128, 3543, 4430, 6504, 7466, 8140, 8349, 8400, 8511, 8763, 9074, 9079, 10110, 10119, 10445, 10499, 10557, 10567, 10791, 10841, 10890, 10950, 10978, 11074, 11748, 11775, 12062, 12183, 12196, 12205, 12435, 12535, 12992, 13041, 13170, 13485, 13601, 14410, 14699, 14719, 14782, 14930, 15100, 15449, 15601, 15782, 15785, 15823, 15996, 17507, 18295, 18643, 18651, 18654, 18661, 18664, 18675, 18684, 18695, 18704, 18710, 18711, 18719, 18728, 18732, 18751, 19133, 19243, 19543, 19565, 19851, 19903, 20120, 20309, 20474, 20494, 20820, 20845, 21034, 21055, 21061, 21225, 21262, 21506, 21735, 21746, 22049, 22076, 22097, 22157, 22167, 22176, 22188, 22198, 22203, 22226, 22231, 22252, 22257, 23008, 23334, 23582, 29978, 30909, 31677, 34037, 38534, 44823, 45568, 45961, 46218, 46474, 46644, 47029, 47078, 47145, 47515, 47778, 48078, 48183, 48652, 48668, 48681, 48751, 48774, 48798, 48818, 48852, 48910, 48958, 48989, 49006, 49018, 49128, 49619, 49631, 49694, 49848, 49882, 49884, 49887, 49901, 49940, 50051, 50125, 50148, 50223, 50237, 50254, 50293, 50310, 50348, 50500, 50622, 50825, 50863, 50894, 50912, 50931, 50933, 50961, 50976, 51385, 51585, 51590, 51661, 51715, 51737, 51754, 51775, 51789, 51823, 51828, 51883, 51889, 51896, 51983, 52008, 52030, 52048, 52068, 52128, 52300, 52325, 52384, 52410, 52450, 52622, 52724, 52735, 52922, 53010, 53052, 53085, 53133, 53151, 53254, 53346, 53357, 53407, 53734, 53809, 53816, 54010, 54154, 54270, 54342, 54370, 54563, 54598, 54634, 54660, 54690, 54804, 54814, 54872, 54989, 55057, 55099, 55134, 55235, 58002, 61539, 62139, 67237, 68526, 69450, 73299, 73370, 73386, 73415, 73488, 73533, 73721, 73799, 73860, 73908, 74022, 74265, 74303, 74347, 74476, 74546, 74629, 74693, 74805, 74966, 75064, 75518, 75942, 83493, 84207, 94741, 96737, 104826, 105367, 106131, 106182, 106579, 106816, 106829, 106832, 106842, 106845, 106848, 106851, 106859, 106865, 107069, 107293, 107510, 107736, 108164, 108599, 108627, 109183, 109402, 109471, 109629, 109826, 110388, 110720, 110950, 111040

7. המילים הכי נפוצות וכמות המופעים הן:

('year', 514088)

('mr', 466682)

('cent', 364410)

('government', 349738)

('pounds', 309763)

('people', 294218)

('time', 258882)

('CHJ', 252213)

('state', 250026)

('market', 249952)

המילים הכי פחות שכיחות שקיימות במאגר , ומיקומן הן:

('11-87', 1)

('1988-9o', 1)

('dantai', 1)

('199o-92', 1)

('kyoryokukai', 1)

('MINAMIZAKI', 1)

('then-jda', 1)

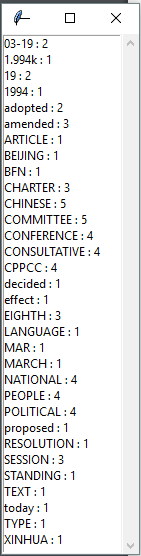
('sobi', 1)

('08-87', 1)

('BENAAN', 1)

8. דיאגמת zip'f law:



  
9. עבור מסמך FBIS3-3366 קיבלנו את רשימת ה-terms הבאה: (term : occurrences number) **סה"כ** 32

10. גודל קובץ ה-posting של המאגר כולו הוא: KB1,370,00 (1.37GB)  
מעבר לכך קיימים 3 קבצי ujson:  
baseDict – 28KB  
citiesIndex – 16.65KB  
fileIndex – 17.5KB