**דוח מנוע חיפוש:**

1. **עיצוב התוכנה:**

בחרנו לעבוד בארכיטקטורת MVVM לכן פתחנו את התיקיות הבאות: MODEL, VIEWMODEL, VIEW.

\*מחלקת MyViewModel אחראית על הקשר בין ה Model ל GUI והעברת פקודות המשתמש.

במחלקה זו מימשנו את הפונקציות הבאות:

1. Void startIndexing(bool stemming,string corpus,string posting)

פונקציה זו מעבירה ל Model את הפקודה להתחיל בתהליך האינדוקס.

1. Void resetIndex(string postingPath)

פונקציה זו מעבירה למודל את הפקודה לאפס את האינדקס.

1. Bool loadTermFreqDict(string postingPath,boolstemming)

פונקציה זו מעבירה למודל את הפקודה לטעון את המילון לזיכרון ומחזירה true או false תלוי אם המילון קיים בתיקייה על הדיסק.

\*מחלקת MyModel אחראית על ביצוע הפעולות הלוגיות של המנוע, דיווח(event) והעברת המידע הנדרש ל ViewModel וממנו ל View.

את הפעולות שניתן לעשות על הדיסק(קריאת המילון מהדיסק) ביצענו במחלקה זו.

במחלקה זו מימשנו את הפונקציות הבאות:

1. Void StartIndexing(bool stemming,string corpus,string posting)

פונקציה זו יוצרת מופע חדש של ReadFile ומתחילה את תהליך האינדוקס.

1. Void resetIndex(string postingPath)

פונקציה זו מוחקת את כל הקבצים בתיקיית ה Posting כולל קבצי ה Posting של ה stemming במידה ויש.

1. bool loadFreqTermDict(string postingPath,bool stemming)

פונקציה זו טוענת את המילון המתאים בהתאם לערך ה stemming(סומן או לא).

היא נעזרת בפונקציית העזר loadDictionary(סעיף ד').

1. Bool loadDictionary(string postingPath)

פונקציה זו בודקת אם המילון קייםעל הדיסק, במידה ולא, תקפוץ הודעה למסך והפונקציה תחזיר false אחרת המילון יטען מהקובץ בדיסק ויוחזר true.

\*מחלקת DocInfo שומרת את הפרטים על המסמכים כמו: Terms יחודיים , Term הכי נפוץ ושפת המסמך.

\*מחלקת TermInfo שומרת לכל term מידע כללי כמו: df- כמות המסמכים בהם הוא הופיע, ומילון השומר מידע על מספר ההופעות שלו בכל מסמך והמיקומים בו.

במחלקה זו מימשנו את הפונקציות הבאות:

1. Void addTermInstance(string docId,int pos)

פונקציה זו מוסיפה מופע חדש של מסמך למילון המסמכים ומעדכנת את הפרטים בו.

1. פונקצית עזר: string buuildDictString()

פונקציה זו עוזרת לנו לתרגם את המילון ל string על מנת שנוכל לשמור את המילון על הדיסק.

\*מחלקת TermDocInfo שומרת את המידע על כל Term ב- Doc : מיקומים ומספר הופעות.

א. void updateTermInfo(int pos)

פונקציה זו מוסיפה מיקום חדש של ה Term במסמך ומעדכנת את ה tf.

ב.פונקציית עזר: string buildPosString()

פונקציה זו עוזרת לנו לתרגם את רשימת המיקומים למחרוזת אשר תיכתב לדיסק.

מבנה התכנית:

* על מנת שהתכנית תעבוד מהר ולא תכביד על המחשב נקטנו במספר צעדים:
* בתחילת התכנית טענו את שמות הקבצים מה corpus לתור.
* יצרנו ThreadPool אשר מוגבל למספר קבוע של Threads (8) כאשר כל Thread לוקח שם של קובץ מהתור , מפריד את המסמכים לפי התגית DOC ולאחר מכן שולח כל DOC ל Parse. לאחר שסיים לבצע Parse לכל המסמכים בקובץ , שולח ל- Indexer שבו נוצר קובץ פוסטינג ואותו כותב לדיסק.

העבודה עם Threads הופכת את הפעולות למקביליות ובכך משפרת את זמן הריצה של התכנית.

* השתמשנו בתהליך זה בתור של Parse על מנת לא ליצור מופעים מיותרים אשר יכבידו על ה RAM.(הגבלנו את מספר ה Parse כמספר ה Thread , 8).
* לאחר סיום כל Thread (סיום עבודה על קובץ) ה GC מפנה מהזיכרון את מערך המסמכים וזה מקל על ה RAM.
* את רשימת ה StopWords בשלב ה parsing שמרנו באמצעות מילון מכיוון שמילון הוא מימוש של hash table עם perfect hashing ובאמצעות מבנה נתונים זה אנו יכולים לבצע בדיקה עבור כל token (שייך/לא ל stop words) ב- o(1).
* לאורך כל התכנית העדפנו להשתמש במילונים לצורך שמירת פרטים שונים(כאלה שמבצעים עליהם חיפושים רבים) מכיוון שזהו מבנה נתונים אשר מאפשר גישה וחיפוש ב – o(1).
* בתכנון הראשוני עשינו מילון משותף עבור כל הקבצים אשר היה סטטי וכל מסמך הוסיף את ה Terms שלו למילון. דרך זו הכבידה על ה RAM(היה שימוש של יותר מ 2GB).

לכן בחרנו לממש זאת על ידי מילון לכל קובץ ושמירתו כקובץ פוסטינג, כאשר המילון נמחק מה RAM על ידי ה GC.

* בחרנו להשתמש בכתיבה בינארית לקובץ- כתיבה בצורה זו חוסכת זמן ומקום.
* את תהליך המיזוג ביצענו ב Threads על ידי שימוש בתור משותף של קבצי פוסטינג.
* במיזוג הסופי של שני קבצי הפוסטינג האחרונים, יצרנו גם את המילון הסופי, זאת כדי לחסוך בזמן ומעבר מיותר על קובץ הפוסטינג הסופי.

**אופן שמירת הפוסטינג:**

בחרנו לשמור קובץ פוסטינג עבור כל קובץ ב corpus (492 קבצי פוסטינג ראשוניים-כמספר קבצי ה corpus). לאחר שה- Thread האחרון סיים לעבוד על הקובץ האחרון השיטה merge נקראת וממזגת בין כל שני קבצים. הקבצים נשמרו בסדר ממוין(ה Terms ממוינים לקסיקוגרפית) וזאת כדי להקל על תהליך המיזוג.

כל קובץ מכיל את שם ה Term ,מספר המסמכים בהם הופיע, פירוט על שמות הקבצים בהם הופיע, מספר המופעים שלו בהם והמיקומים שלו בהם.

לאחר מיזוג שני קבצים, נמחקים הקבצים שמוזגו על מנת לפנות מקום על הדיסק.

ביצענו הפרדה בין כל נתון על ידי סימנים מוסכמים שקבענו וזאת על מנת שנוכל לאחד בין Terms זהים בין שני מסמכים שונים.

* קיים trade off בין כמות השימוש ב RAM לבין זמן הריצה של התכנית.trade off זה נקבע על ידי גודל קובץ הפוסטינג הראשוני(במקרה שלנו שמרנו עבור כל קובץ אחד, קובץ פוסטינג).

במידה והיינו מגדילים את מספר הקבצים עבורם היינו שומרים קובץ פוסטינג בודד, השימוש ב RAM היה גדל ובמקרה ההפוך זמן הריצה היה גדל.

לאחר מספר ניסויונות הרצה הגענו לגודל האופטימלי עבור התכנית שלנו שהוא פוסטינג עבור כל קובץ.

* בחרנו לשמור את הפרטים של ה Terms והמסמכים באובייקט נפרד וזאת על מנת לאפשר שינויים מהירים בקוד והוספת פרטים רלוונטים על ה Terms ועל המסמכים.

**עבור כל Term** שמרנו את מספר המסמכים בהם הוא מופיע, כמה פעמים בכל מסמך ובאילו מיקומים הוא מופיע.

**עבור כל Doc** שמרנו את מספר ה terms היחודיים בו, השפה בו כתוב וה- term הכי נפוץ בו.

* החוקים שהוספנו הם:

1. הפרדת term עם / באמצע לשני terms נפרדים.

דוגמה א': בקובץ FB396003 במסמך FBIS3-232 ה-term: Gushiegu/Karaga יהפוך לשני terms והם: Gushiegu ו Karaga.

דוגמה ב': בקובץ FB396019 במסמך FBIS3-4822 ה-term: OANA/BERNAMA יהפוך לשני terms והם: OANA ו BERNAMA.

1. איחוד terms אשר מופיעים בסדר הבא: number to number יהפוך ל number-number.

דוגמה א': בקובץ FB396002 במסמך FBIS3-52 ה-terms: 8 to 10 יהפכו ל term אחד: 8-10.

דוגמה ב': בקובץ FB396003 במסמך FBIS3-90 ה-terms: 1000 to 1530 יהפכו ל term אחד: 1000-1530.

* על מנת לצמצם את מספר ה terms הסופיים השתמשנו ב Trim ו Replace עבור סימנים מיוחדים.
* כדי לעבוד ביעילות עם תגיות Html השתמשנו בספרייה HtmlAgilityPack אשר מאפשרת טעינה מקובץ של קוד Html ופיענוחו והפרדתו על פי תגיות בזמן ריצה מהיר. הורדנו את הספרייה דרך ה Visual studio והשתמשנו בשירות זה רק במחלקת ה ReadFile.

**2. ניתוח המאגר:**

**1. לפני stemming ישנם 496776 terms.**

**2. אחרי stemming ישנם 436586 terms.**

**3. - ישנם 17743 terms שהם מספרים (לפני stemming).**

**- ישנם 17745 terms שהם מספרים(אחרי stemming).**

**4.** 10 ה- terms השכיחים ביותר:

government , 145440

people , 118051

text , 117131

state , 112729

minister , 111872

economic , 105468

foreign , 104301

president , 101297

party , 100187

article , 92947

10 ה- terms הכי פחות שכיחים:

$?90 , 1

$]1350000 , 1

$0.50-per , 1

$1-$20 , 1

$1.00-2.00 , 1

$1.036bn , 1

$1.15-billion , 1

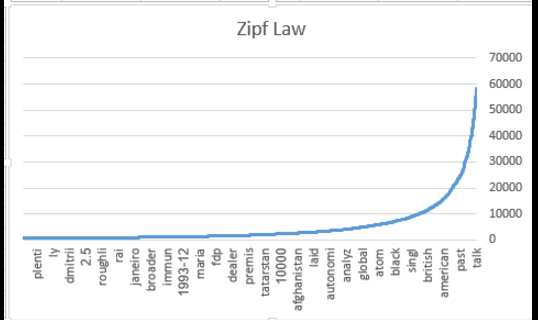
$1.1-billion , 1

$1.2-1.38 , 1

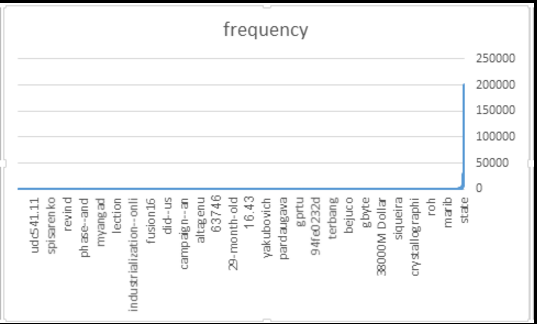
$1.2b , 1

5.

גרף 1:



גרף 2:



בגרף 1 השמטנו את 50 הטרמים הנפוצים ביותר ולקחנו רק את 5000 הטרמים הראשונים. העקומה שיצאה אכן מזכירה את העקומה של חוק זיף. הגרף השני הוא על כל הטרמים במאגר וקשה מאוד לראות צורה שמזכירה את העקומה של חוק זיף. זאת מכיוון שיש הרבה יותר טרמים עם שכיחות נמוכה מאשר טרמים עם שכיחות גבוהה. כמו כן הטרמים הכי שכיחים חוזרים על עצמם הרבה מאוד פעמים וההבדל בין הטרם הכי שכיח הראשון לטרם הכי שכיח השני גדול מאוד לכן הקנה מידה של הגרף השני גדול מידי. את זה תיקנו בגרף 1 על ידי השמטת 50 הטרמים השכיחים ביותר.

6. רשימת ה-terms של המסמך FBIS3-3366:

03-19 , 1

1994-03-19 , 1

adopted , 2

amended , 3

article , 1

beijing , 1

charter , 3

chinese , 5

committee , 5

conference , 4

consultative , 4

cppcc , 4

decided , 1

effect , 1

eighth , 3

language , 1

national , 4

peoples , 4

political , 4

proposed , 1

resolution , 1

session , 3

standing , 1

text , 1

today , 1

typebfn , 1

xinhua , 1

7. נפח האחסון הנדרש על הדיסק לקבצי הפוסטינג(עם stemming ובלי stemming):

855 MB (897,161,481 bytes)