תיאור העבודה אופן פעולת התכנית:

* עיקר התכנית בנוי משני חלקים אשר יחד יוצרים את קבצי הPOSTING השונים והמילון בהתאמה. קבצי הפוסטינג והמילון מהווים יחד את הINVERTED INDEX אשר יעזור בהמשך לאחזר את קבצי המאגר לפי השאילתות שיזין המשתמש.

חלק א- שלבי הקריאה, TEXT OPERATION, פרסור ויצירת הקבצים הזמניים.

כל הפעולות אשר מבוצעות בחלק זה הינן מופרדות אחת מהשניה והינן בלתי תלויות אחת בשנייה. תוצר החלק הנ"ל הינו קבצי פוסטינג זמניים אשר יהוו איחד מהיר ונוח לחלק הבא. המכנה המשותף לכל המחלקות הנ"ל היא שיטת התקשורת בין השלבים, תקשורת Producer Consumer . מחלקות אשר מהותן PRODUCER(יצירת אובייקטים) יקבלו תור אשר אליו יכניסו את האובייקט אותם הם מייצרות ומחלקות אשר מהותן CONSUMER(צריכת אובייקטים) יקחו את האובייקטים מאותו התור. חלק מהשלווים מתנהגים כצרכן עבור השלב הקודם להם וכיצרן לשלב הבא. לצורך חסכון בזמן הריצה וייעול חלק א ככלל כל שלב מתופעל ע"י תרדים (THREADS) הממקבלים את השלבים של חלק זה. על מנת לשלוט על התרדים, לקרוא לכל המחלקות באותו הזמן, לסדר אותם בסדר הנכון ולתת להם את התור הרלוונטי ישנה מחלקת master(תיאור בהמשך) אשר כל יעוד הוא ליצור ולהרוג את השלבים השונים בחלק זה. בכדי לשלוט בכמות המידע הנכנס משלב "קריאת הקבצים" נתנו לכל שלב שני SEMAPHORES אשר מגיבילים את קצב הצריכה מצד אחד(ובנוסף מורידים Busy waiting ) וקצב העברת האובייקטים לשלב הבא מצד שני. מאחר והשתמשנו בthreads נאלצנו להתמודד עם מרוצים בין תרדים השונים. לשם כך, השתמשנו בשיטת הpoisoned pill: כאשר מחלקת המסטר רואה כי היצרן סיים את עבודתו הוא שם בתור אשר קיים אצל הצרכן הרלוונטי אובייקט אשר יש הסכמה עליו כי יפסיק את תהליך הצריכה. באמצעות שיטה זאת מחלקות הצרכן לא יירדמו כאשר ינסו לקחת אובייקטים אשר כבר לא מיוצרים ולא היווצרו מרוצים בין התרדים השונים.

וכעת נתבונן בשלבי חלק א' ומחלקותיהן.

שלב קריאת הקבצים

מחלקת Document Reader

מחלקה זו היא מחלקה מסוג יצרן (Producer) והיא מממשת את ממשק הrunnable בכדי שתהיה אפשרות למקבל אותה. מטרת המחלקה הינה לקחת את הנתיב אשר היא מקבלת, לקרוא את קובץ מהנתיב ולחלק את הקבצים למסמכים. את המסמכים היא שמה בתור לצרכן הרלוונטי, קוראת עוד קובץ וחזור חלילה. המחלקה מסיימת את עבודתה כאשר לא נותרו קבצים לקרוא

שיטות:

**public void** run()

שיטה זו קוראת את הקובץ מהנתיב(התקבל בבנאי) מחלקת את הקובץ למסמכים בעזרת מחלקת XMLREADER, יוצרת אובייקט מסוג DOCUMENT ושמה אותו בתור.

מחלקת XML Reader

מחלקה זו עוזרת למחלקת Document Reader לפצל את המסמך למסמכים לפי תגיות המסמך.

שיטות:

**public** TextOperations.Document getNextDocument()

שיטה זו לוקחת את הקובץ (מקבלת בבנאי) ובעזרת האובייקט ELEMENT מהתוספת JSOUP מגדירה את המסמכים בקובץ. השיטה מפצלת כל אלמנט לפי התגיות של המסמך על מנת לקבל את הטקסט של המסמך, המדינה בה הטקסט נכתב וכו'.. השיטה מחזירה אובייקט מסוג DOCUMENT עם פרטי המסמך.

**public boolean** hasNext()

שיטה המחזירה ערך אמת אם קיים מסמך נוסף בקובץ, אחרת שקר.

מחלקת Document

מחלקה זו מייצגת את המסמך בעזרת רשימות של STRING לפי פיצול התגיות.

שיטות:

**public** String getPath()

שיטה זו מחזירה את נתיב המסמך

**public** String getID()

שיטה זו מחזירה את קוד הזיהוי של המסמך

**public** String getHeader()

שיטה זו מחזירה את הכותרת של המסמך

**public** String getDate()

שיטה זו מחזירה את תאריך יצירת המסמך

**public** String getText()

שיטה זו מחזירה את תוכן המסמך

שלב ה TEXT OPERATION

מחלקת Text Operation

מחלקה זו הינה מסוג צרכן של אובייקטים מסוג Documents. בנוסף לכך, המחלקה הינה מסוג יצרן של אובייקטים מסוג Tokenized Document. המחלקה מממשת את ממשק הrunnable בכדי שתהיה אפשרות למקבל אותה. מטרת המחלקה היא להוריד את חלק מה STOP WORDS וליצור TOKEN מכל מילה.

שיטות:

**public void** run()

השיטה לוקחת אובייקטים מסוג document מהתור , מעבירה את האובייקט דרך פילטר אשר מוריד מילים לא נחוצות. לאחר מכן השיטה יוצרת אובייקט מסוג Tokenized Document ודוחפת אותו לתור.

שלב הפירסור

מחלקת Master Parser

מחלקה זו הינה מסוג צרכן של אובייקטים מסוג Tokenized Document וגם מסוג יצרן של אובייקטים מסוג HashMap. המחלקה מממשת את ממשק הrunnable בכדי שתהיה אפשרות למקבל אותה. המחלקה יוצרת מפה של<String,TermDocumentInfo> כאשר המפתח הינו שם הTerm והערך הינו המידע על הTerm במסמך יחיד. המחלקה קוראת לכל הפרסרים השונים ונותנת להם לפרסר לפי חוקיהם ולהכניס את הערכים לאותה המפה.

שיטות:

**public void** run()

השיטה לוקחת אובייקטים מסוג Tokenized Document מהתור, יוצרת מפה ושולחת את שני האובייקטים הנ"ל למחלקות פרסור. בסוף התהליך השיטה דוחפת את המפה לתור.

ממשק Info

ממשק להחזרת מידע בצורת String

שיטות:

String toString()

השיטה מחזירה את המידע הרלוונטי על האובייקט בצורת String

ממשק ItermDocumentInfo

ממשק המתאר פונקציונליות של מידע על Term במסמך Document מסויים.

שיטות:

**int** getFrequency()

מחזיר את התדירות של המילה במסמך

Term getTerm()

מחזיר את הTerm

String getDocumentID()

מחזיר את השם המזהה של המסמך

**void** addToFrequency(**int** frequency)

השיטה מוסיפה לערך התדירות של המילה את המספר שקיבלה

**void** setFrequency(**int** frequency)

השיטה קובעת את ערך התדירות של המילה לפי המספר שקיבלה

מחלקת Abstract Term Document Info

המחלקה מממשת את הממשק ItermDocumentInfo. זוהי מחלקה מופשטת אשר מייצגת מייד על Term

מחלקת Term Document Info

המחלקה יורשת ממחלקת Abstract Term Document Info.

מחלקת Ciyt TDI

המחלקה יורשת ממחלקת Abstract Term Document Info. המחלקה מייצגת מידע עבור term המייצג עיר.

שיטות

**void** addLocation(**int** location)

השיטה מוסיפה מיקום של המילה

מחלקת Abstract Parser

המחלקה מייצגת אובייקט המסוגל לפרסר אובייקטים מסוג TokenizedDocument על פי חוקים כלשהם.

שיטות:

Token get(**int** i)

השיטה מחזירה את הערך ה-i ברשימת הטוקנים

**abstract void** manipulate()

השיטה הינה שיטה מופשטת מאחר וכל Parser היורש ממחלקה זו יממש את הפרסור בצורה שונה עם חוקים שונים. השיטה תעבור על רשימת הטוקנים ותכניס את המילים (term) המתאימים לחוקי הפרסר למפה.

**void** putInMap(String s)

השיטה לוקחת את הString שיש להכניס למילון ובודקת האם הוא כבר קיים. אם קיים, המפה תעדכן את הערכים של AbstractTermDocumentInfo אחרת תיצור AbstractTermDocumentInfo חדש ותכניס את הערכים למפה כך שs הוא המפתח.

**int** getTxtSize()

השיטה מחזירה את גודל רשימת הטוקנים

**boolean** mapContains(String s)

השיטה מחזירה ערך אמת אם ערך המילה s קיים במילון, אחרת שקר.

**void** mapRemove(String s)

השיטה מעבירה את ערכי s.toUpperCase() לערך s ומוחקת את ערכי s.toUpperCase() מהמפה.

מחלקת CountryParser

המחלקה יורשת ממחלקת Abstract Parser . המחלקה מפרסרת לפי החוק הבא: עבור כל עיר המופיעה בטקסט וגם בתגית FP 104 יש להוסיף 1 לאובייקט. אם לא קיים האובייקט, צור אותו.

שיטות:

CountryInfo getCountryInfo(String CapitalName)

השיטה מחזירה עבור שם העיר את נתוני העיר מאתר ה restful api.

מחלקת Dates And Range Parser

המחלקה יורשת ממחלקת Abstract Parser . המחלקה מפרסרת לפי החוק הבא: אם המחלקה מזהה מילים מסוג תהליך או טווחים מסויים בטקס היא מוסיפה אותם כ Term למפה.

שיטות:

String convertToYear(String string)

השיטה ממירה String המייצג שנה לפורמט YYYY .

Double isNumber(Token token)

השיטה מנסה להמיר את המילה במשתנה token לdouble אם הצליחה, שולחת משתנה מסוג Double אחרת שולחת null

מחלקת Number Parser

המחלקה יורשת ממחלקת Abstract Parser . המחלקה מפרסרת לפי החוק הבא: אם המחלקה מזהה מילים מסוג מספר היא מפרסרת אותו לפי חוקי הפרסור ומוסיפה אותה כ Term למפה.

Double isNumber(Token token)

השיטה מנסה להמיר את המילה במשתנה token לdouble אם הצליחה, שולחת משתנה מסוג Double אחרת שולחת null

**boolean** isFraction(String string)

השיטה מחזירה ערך אמת אם המילה בפורמט שבר לדוגמא: 3\5 ,אחרת שקר.

String convertNumber(String string, Boolean isTrillion)

השיטה לוקחת מספר בפורמט String וממירה אותו לפי חוקי הפרסור.

מחלקת Percent and Price Parser

המחלקה יורשת ממחלקת Abstract Parser . המחלקה מפרסרת לפי החוק הבא: אם המחלקה מזהה מילים המייצגים אחוז או מחיר, היא תוסיף את המילה כTerm למפה.

**boolean** isFraction(String string)

השיטה מחזירה ערך אמת אם המילה בפורמט שבר לדוגמא: 3\5 ,אחרת שקר.

String convertNumber(String string, Boolean isTrillion)

השיטה לוקחת מספר בפורמט String וממירה אותו לפי חוקי הפרסור.

String convertNumberLessthanMill(String tokenStr)

השיטה מחזירה ערך מספרי מפורסר לפי חוקי הפרסור.

Double isNumber(Token token)

השיטה מנסה להמיר את המילה במשתנה token לdouble אם הצליחה, שולחת משתנה מסוג Double אחרת שולחת null

מחלקת Word Parser

המחלקה יורשת ממחלקת Abstract Parser . המחלקה מפרסרת לפי החוק הבא: תוסיף כל מילה שלא נתפסה בפרסורים הקודמים.

שלב הקבצים הזמניים

מחלקת Segment File

המחלקה הינה מחלקה מופשטת המייצגת מסמך פוסטינג זמני.

שיטות:

**abstract void** add(String key,Info item)

שיטה מופשטת המוסיפה Info לTerm מסויים.

**void** write()

שיטה הקוראת למחלקת הכתיבה האישית של הסגמנט. ((strategy deign

List<String> read()

שיטה הקוראת למחלקת הקריאה האישית של הסגמנט(strategy deign) ומחזירה רשימת סטרינגים.

String getPath()

השיטה מחזירה את הנתיב בו ישמר הקובץ הזמני(הסגמנט)

מחלקת Term Segment File

המחלקה יורשת ממחלקת Segment File.

מחלקת City Segment File

המחלקה יורשת ממחלקת Segment File.

מחלקת Document Segment File

המחלקה יורשת ממחלקת Segment File.

מחלקת Term Segment Files

מחלקה זו הינה מסוג צרכן של אובייקטים מסוג HashMap<String,AbstractTermDocumentInfo>. בנוסף לכך, המחלקה הינה מסוג יצרן של אובייקטים מסוג Segment File. המחלקה מממשת את ממשק הrunnable בכדי שתהיה אפשרות למקבל אותה. מטרת המחלקה היא לאסוף מסמכים מהפרסר, לקבץ אותם יחד ולשלוח לכתיבה לקובץ זמני.

שיטות:

**void** run()

השיטה מושכת את האובייקטים מהתור עד שהגיעה למספר אובייקטים אשר מוגדר לה מראש. השיטה מקבצת את אוסף האובייקטים לאובייקט אחד ושולחת לתור הכתיבה.

מחלקת Segment Files Writer

מחלקה זו הינה צרכן של אובייקטים מסוג Segment File. המחלקה מממשת את ממשק הrunnable בכדי שתהיה אפשרות למקבל אותה. מטרת המחלקה היא לרשום את האובייקטים.

שיטות:

**void** run()

השיטה לוקחת מהתור את אובייקטי הסגמנט ורושמת אותם לקבצים בנתיב הנתון.

ממשק Segment Writer

מטרת הממשק היא לתת פונקציונליות כתיבה למחלקה אשר תממש אותה.

המחלקות: Segment Term Writer, Segment City Writer, Segment Document Writer מממשות את הממשק Segment Writer. מטרת המחלקות היא לכתוב את האובייקט המתאים לה.

ממשק Segment Reader

מטרת הממשק היא לתת פונקציונליות קריאה למחלקה אשר תממש אותה.

המחלקות: Segment Term Reader, Segment City Reader, Segment Document Reader מממשות את הממשק Segment Reader. מטרת המחלקות היא לקרוא את האובייקט המתאים לה.

חלק ב'-שלבי הקריאת קבצים זמניים, מיזוג, שמירת קבצי פוסטינג ומילון.

חלק זה הינו המשך ישיר לחלק א'. כעת, לאחר שכל הקבצים הזמניים נשמרו בדיסק(כל מקבץ קבצים נשמר ב55 קבצים זמניים לפי התו ההתחלתי שלו) ניתן לקרוא חלקים מהקבצים הזמניים ולאחד אותם למילון אחד גדול ולשמור אותם בקבצי פוסטינג סופיים. חלק זה אינו מקבילי מכיוון שבחרנו שלא להעמיס על הראש קורא כותב של הדיסק. לאחר בדיקות וניסויים ראינו כי חלק זה מהיר דיו גם אם ירוץ בצורה רגילה.

וכעת נעבור למחלקות של השלבים בחלק ב'.

מחלקת Indexer

מחלקה זו אחראית על כל שלב האיחוד והאינדוקס של הקבצים הזמניים, במחלקה זו נוצרים קבצי הפוסטניג והמילון בהתאמה.

שיטות:

**void** CreatePostFiles(String segmentLocation)

שיטה זו מקבלת נתיב אל התקייה בה נמצאים הקבצים הזמניים. המחלקה עוברת לפי התווים הנתונים לה ומחפשת את הקבצים הזמניים בהם המילים מתחילות בתו הראשון של התו הנתון. לאחר מכן עבור כל קובץ זמני ניקראת פעולת מיזוג המכניסה את הערכים החדשים למילון הזמני ומעדכנת את הערכים הישנים. בסוף כל איטרציה נכתבים הנתונים לקובץ פוסטינג המחולק גם הוא ל55 קבצים (לפי התווים הנתונים).

**void** deleteAndMerge(String term, String s,HashMap<String,String> chunkTermIndex)

שיטה זו מקבלת ערך מילה ומילון זמני ומוחקת את מופע המילה באותיות גדולות במילון זה. לאחר מכן השיטה קוראת לאיחוד המילה במופע של אותיות קטנות ועדכון המידע הרלוונטי.

**void** merge(String term, String s,HashMap<String,String> chunkTermIndex)

שיטה זו מאחדת את המידע של מילה term במילון ומעדכנת את נתוניה.

התמודדות עם מגבלות הזכרון

אחד הקשיים הגדולים של העבודה היה מגבלת הזכרון. ביצענו בדיקות רבות ושונות על חלוקת זיכרון בין השלבים. צורת הבנייה של המחלקות והתקשורת בינהם איפשרה לנו יכולת שליטה גבוהה בכמות המידע הנכנס למחלקות ויוצא מהמחלקות. שיטת הProducer Consumer אפשרה לנו להגביל את הזכרון מבחינת כמות המידע העובר בין המחלקות. החלטנו בעבודה כי מחלקות הקריאה צריכות להיות מוגבלות יותר מאשר מחלקות הParse וה Text Operation מאחר וקריאת הקבצים הייתה משמעותית מהירה יותר. יתר על כן, חילקנו יותר Threads למחלקות האיטיות יותר בכדי שיעבירו את המידע הלאה לשלבים הבאים. כל זה נועד בכדי שהמידע ירשם מהר לדיסק ולא יהיה בזכרון. ובעזרת כל הכלים הללו צריכת הזכרון הייתה יחסית נמוכה וכל השלבים עבדו מהר.

שמירת קבצי הפוסטינג

שמירת קבצי הפוסטינג מחולקת ל55 קבצים. כל קובץ מייצג תו. תו זה הינו התו הראשון של המילה. חילקנו את קבצי הפוסטינג בצורה זו על מנת להקל על שמירת הנתונים לאחר הקבצים הזמניים. בזמן האינדוקס של קבצי הפוסטינג, בכל איטרציה, שלפנו את המילים לפי התו הראשון לכן בכל עת בזכרון המכנה המשותף של המילים הנימצאים בזכרון היה התו הראשון. שמירת מילים אלו בקובץ אחד הגביר את יעילות הריצה מכיוון שחסכנו בכתיבות ומיונים מיותרים בין המילים בעלות תו שונה.

שמירת הקבצים הזמניים

כפי שצויין, לאחר בדיקות שונות החלטנו על כמות המידע הנכנסת למחלקות ויוצאת ממחלקות. מחלקת הSegment Files הייתה המחלקה האחראית לאגור מקבץ של מסמכים ולשלוח מקבץ זה לכתיבה של קבצים זמניים. במחלקת הSegment Files ישנו משתנה int numOfDocs משתנה זה מחליט על כמות הכתיבה בכל עת. מניסויים רבים, השתכנענו כי רישום של 10,000 מסמכים יחדיו מקנים יעילות מבחינת ריצת והתכנית וגם מבחינת חיסכון בזיכרון התוכנית.

יצירת המילון

יצירת המילון נעשה בתהליך האינדוקס. לאחר שליפה של קבצים זמניים לפי תו, נעשה מעקב של המילים שהתקבלו מהקבצים הזמניים. עבור כל מילה בדקנו האם היא נמצאת במילון, אם כן עודכנו נתוני הערך במילון עד שלב הכתיבה. בשלב הכתיבה ערך המילה נכתב לפוסטינג ונמחק מהזכרון. מילים שלא הופיעו במילו התווספו כמפתח חדש. בשלב זה וידאנו בנוסף, האם המילה צריכה להישמר באותיות גדולות או קטנות. בסוף שמירת קבצי הפוסטינג, בהכרח נשמר המילון עבור כל מילים במאגר. ערכי המילה במילון הינם: מצביע(מספר קובץ ומספר שורה, מספר החזרות של המילה במאגר, מספר המסמכים בהם הוא מופיע.

פרטי האינפורמציה שהוספנו הם:

* אורך המסמך לפי הTerms . מידע זה יסייע לנו בהמשך לחשב את מידת הרלוונטיות של המסמך ביחס למסמכים אחרים בהינתן שאילתא מתאימה.

החוקים החדשים אותם הוספנו:

* From number and number לאחר מעבר על חלק מהמסמכים זיהינו כי ההגדרה של טווח מצורה זו חוזר מספר רב של פעמים. לכן, נישמור את רצף המילים הללו כחוק פארס אחד על מנ לאחזר שאילתות טווחים בצורה מהירה.
* צורת השייכות S' . צורת כתיבה זו נפוצה בשפה האנגלית. סיומת זו מייצגת שייכות העצם לעצם אחר. מאחר וסיומת זו אינה מוסיפה ערך למילה אליה היא נוספת, נבחר להוריד אותה על מנת ליצור אחזור מדיוק יותר. אם המילה תשאר אם הסיומת במילון, ערכה יהיה שונה מאותה המילה ללא הסיומת. דבר זה גורם לאחזורים לא איכותיים.