**מגישים:**

עידן רזי 204475255

אביב תורג'מן 203959523

**דוח פרויקט מנוע אחזור חלק א':**

1. **עיצוב הפרויקט :**
   1. **מחלקות :** 
      * **Files** – משיכת מידע מתיקיית הcorpus
        + Getfile : מושך קובץ מהדיסק
        + Getdocument : בניית מערך מסמכים מתוך קובץ ע"פ התגיות <doc>
        + Getfilecount – כמות הקבצים במאגר
        + Getdoccount – כמות המסמכים במאגר
        + Getstopwords – קריאת stopwords וטעינה למאגר
      * **Parse** – ביצוע פירסור לכל המילים במסמך ע"פ החוקים שהוגדרו + 2 חוקים משלנו (סעיף ד') **נספח עץ גזירה.**
        + parseDoc- עובר על כל מסמך בקובץ ושולח את המילים במסמך לפרסור
        + Parse- בונה את המילה החדשה תו אחרי תו תוך שמירת מידע אודות המילה הנבנית
        + cutWord- חותך את המילה שנבנתה תוך עיבוד המידע שנשמר עבורה (עלולה לקבל משמעות חדשה)
        + add- הוספת המילה אל המאגר
        + setMonth-מאתחלת מבנה פנימי של המחלקה לשם השוואת חודשים
        + setStopWords- מאתחל את המידע שנשמר על מילה מסויימת (ראה עץ גזירה)
        + remove- מבצעת stmming למילים (במידת הצורך) בנוסף יוצרת א מילון המסמכים.
      * **Stemmer** – ביצוע סטמינג למילה ע"פ אלגוריתם .Porterstemmer
      * **Indexer** – ביצוע את התהליך האינדוקס למאגר:
        + **Posting** : פונקציה אשר קבלת רשימת המילים במסמך לאחר parse+stem , בניית רשימת post ע"פ הפורמט הבא :

"=>"+docid:data "#" term.

כאשר "#" משמש כדלימטר בין המסמכים ו "=>" משמש כדלימטר בין terms. לדג'

father=>doc0: data # doc1 :data

dog=>doc0: data # doc1 :data

ושמירת הרשימה במילון **זמני** postlist(string,string). המילון עוזר לצבור רשימותterms לכל קבוצת קבצים שעובדו בשלבים. גודל הקבוצה נקבע ע"פ זמני ריצה ואילוצי הפרויקט שנתנו, ויפורט בהמשך.

בנוסף הפונקציה סופרת מופעי idf , max tf ומעדכנת במילון הסופי.

* + - * **Write\_to\_post :** כתיבת קובץ posting זמני ממוין ע"פ סדר אלפביתי בתור קובץ בינארי שמכיל אובייקטי string. כל string מכיל רשימת פוסטינג חלקי של terms .
      * **Sortpostlist :** פונקציית עזר ממיינת מילון posting לרשימה מסודרת של postlist לפי סדר אלפבתי.
      * **Sortidf:** פונקציית עזר אשר ממיינת את המילון הסופי לפי DF.
      * **Sortitf:** פונקציית עזר אשר ממיינת את המילון הסופי לפי sum tf (כמות מופעים) לכל מילה .
      * **Build\_max\_idf\_tf:** פונקציית עזר שמחשבת את הדברים הבאים:
        1. 10000 מילים בעלות DF הכי גובה – בשביל בניית זיכרון המטמון
        2. 10 מילים בעלות DF הכי שכיחות – בשביל הדוח הסופי.
        3. 10 מילים בעלות tf הכי שכיחות – בשביל הדוח הסופי.
      * **SaveDictionary :** שמירת המילון הסופי בזיכרון כקובץ בינארי.
      * **LoadDictionary:** טעינת המילון הסופי לזיכרון.
      * **SaveCache :** שמירת זיכרון המטמון הסופי בזיכרון כקובץ בינארי.
      * **LoadCache:** טעינת זיכרון המטמון לזיכרון.
      * **Fileopen/close:** פונקציות עזר לפתיחת וסגירת קבצים בינארים.
      * **Read\_from\_post :** מאפשרת קריאת שורת term מקובץ בינארי.
      * **Get\_term\_frompost :** מאשרת קריאת term מקובץ הpost הסופי . לדג' קבלת המילה father תחזיר את שורת הpostlist הסופית של המילה father.
      * **Merge : פסודו – קוד בנספחים** ממיינת את כל קבצי הפוסטינג הזמנים במיון external merge. בנוסף מעדכנת במילון את המיקום של כל רשימת פוסט של כל term בקובץ הסופי ומעדכנת את מילון המטמון בהתאם למילים שבחרנו ומעדכנת מצביע במילון הראשי שהמילה הזו בזיכרון המטמון ולא בדיסק.
      * **פונקציות הדפסה print dic/cache :**  בשביל להדפיס את המילון והcache למסך הconsole
    - **Info:** מחשבת את הmetadata של הפרוייקט : גודל מילון,פוסט,קאש,כמות מילים,כמות קבצים ומסכים.

המחלקה רושמת קובץ לוג לדיסק , בנוסף בעלת פונקציה של כתיבת מילים לקובץ

CSVלטובת ניתוח כמו ZIPF LAW וכדומה.

* + - **Priority :** מימוש של תור עדיפויות מהאינטרנט.
    - **Term:** אובייקט שנשמר במילון הסופי : שם , idf , מיקום בקובץ הפוסטינג , ערך BOOL האם המילה בזיכרון המטמון, שכיחות המילה ואפשרות להוספת מידע נוסף בעתיד.
    - **Control –** מחלקת gui+ viewcontrol אשר מנהלת את תהליך יצירת המאגר.
      * שימוש בthreads על מנת לייעל זמני ריצה.
      * run\_Click : event אשר מפעיל את הפונקציה compute
      * browseCurpus\_Click : event שפותח בחירת נתיב של תיקיה עבור המאגר וה-stop words
      * browsePost\_Click: event שפותח בחירת נתיב של תיקיה עבור קבצי הפוסט
      * checkBox1\_CheckedChanged – event המגדיר האם לבצע stemming
      * saveCache\_Click: event שפותח בחירת נתיב של תיקיה ושומר בה את המילון ואת הקאש
      * loadCache\_Click: event שפותח בחירת נתיב של תיקיה וטוען ממנה את המילון ואת הקאש
      * diplayDictionary\_Click: event המציג את המילון שנמצא בזכרון
      * diplayCache\_Click: event המציג את הקאש שנמצא בזכרון
      * clear\_Click: event שמנקה את הזכרון ואת הקבצים שנשמרו עבור הריצה הבאה
      * newshow: event המציג נתונים כללים עבור הריצה האחרונה
      * compute – פעולה אשר קוראת לכל הפעולות שהמנוע מבצע: קריאת קבצים, parsing, הסרת stopwords, stemming , indexing, merging ועיבודי מידע שונים. הפונקציה מיעלת את הריצה בערת שימוש במקביליות (threads) תוך מניעת מירוצים
      * thread1: thread שאחראי על parsing ו-indexing
      * thread2: thread שאחראי על merging
  1. **ניהול אילוצים :**
     + עובדה על גבי 10% מגודל המאגר:

גודל המאגר כ1800 קבצים , החלטנו לעבד כל 4 קבצים שונים ולשמור בקובץ posting זמני בדיסק , בסיום עיבוד כל קבוצה ננקה את מבני הנתונים הזמניים מהזיכרון הראשי . החלטה נבעה בגלל יעילות זמן ריצה (מבדיקת זמנים בהרצות עם קבוצות קבצים בגדלים שונים) וגם כדי לעמוד באילוץ של 10% מכלל המאגר.

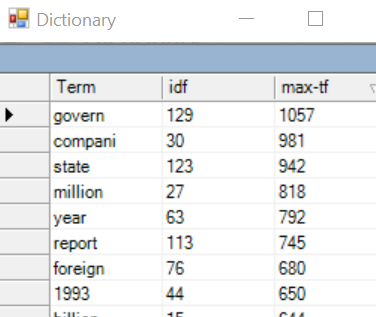
* + - עבודה עם קבצים בינארים:

הוחלט לעבוד עם קבצים בינארים אשר שומרים אובייקטים כמו שהם מופיעים בזיכרון ולא בצורה טקסטואלית , קבצים בינארים מאפשרים שמירת אובייקטים בצורה יעילה (קריאה וכתיבה מהירים מכיוון שהמידע הרצוי נטען ישרות לזיכרון הראשי מבלי ביצוע עיבוד נוסף כמו שצריך בטקסט).

* + - קריאת 3 שורות מכל קובץ זמני – משתמשים בexternal sort , קוראים שורה אחת כל פעם מהקובץ עם הterm המינימלי (ע"פ סדר אלפביתי) . **פסודו קוד למיון בנספחים.**
  1. **מבנה המילון :** המילון מוחזק באובייקט מסוג Dictionary<string,Term> בצורה הבאה:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Value: Trem(object that contain data of term) | | | | Key: Term(string) |
| Cache(bool) האם נמצא במטמון | **Postion –** מצביע לpostlist | **Toatal -tf** | **Idf** |
| true | 233200 | 17 | 6 | Cat |
| false | 233445 | 4 | 5 | Dog |

\*קובץ המילון הסופי הוא בינארי – ניתן לראות את המילון ב-gui בצורת טבלה



בנוסף נשמור גם מילון מסמכים שיכיל: שם המסמך, שם הקובץ, שכיחות המילה הנפוצה ביותר, וכמות המילים במסמך. על מנת לבחון קשר בין מסמכים המאוגדים תחת אותו קובץ

* 1. **בניית זיכרון הCache:** החלטנו לבחור את 10000 המילים הנפוצות ביותר במאגר מבחינת מדד DF , מתוך ההנחה שהמילים הכי נפוצות במאגר (מבחינת הופעות במסמכים שונים) כנראה גם יהיו נפוצות בשאילתות המשתמשים ולכן נרצה לקרוא אותם במהירות ולשמור גישה אליהם בזיכרון.

אופן בניית הזיכרון: זיכרון המטמון הוא בעצם מילון שמחזיק לכל מונח את רשימת הpost שלו.

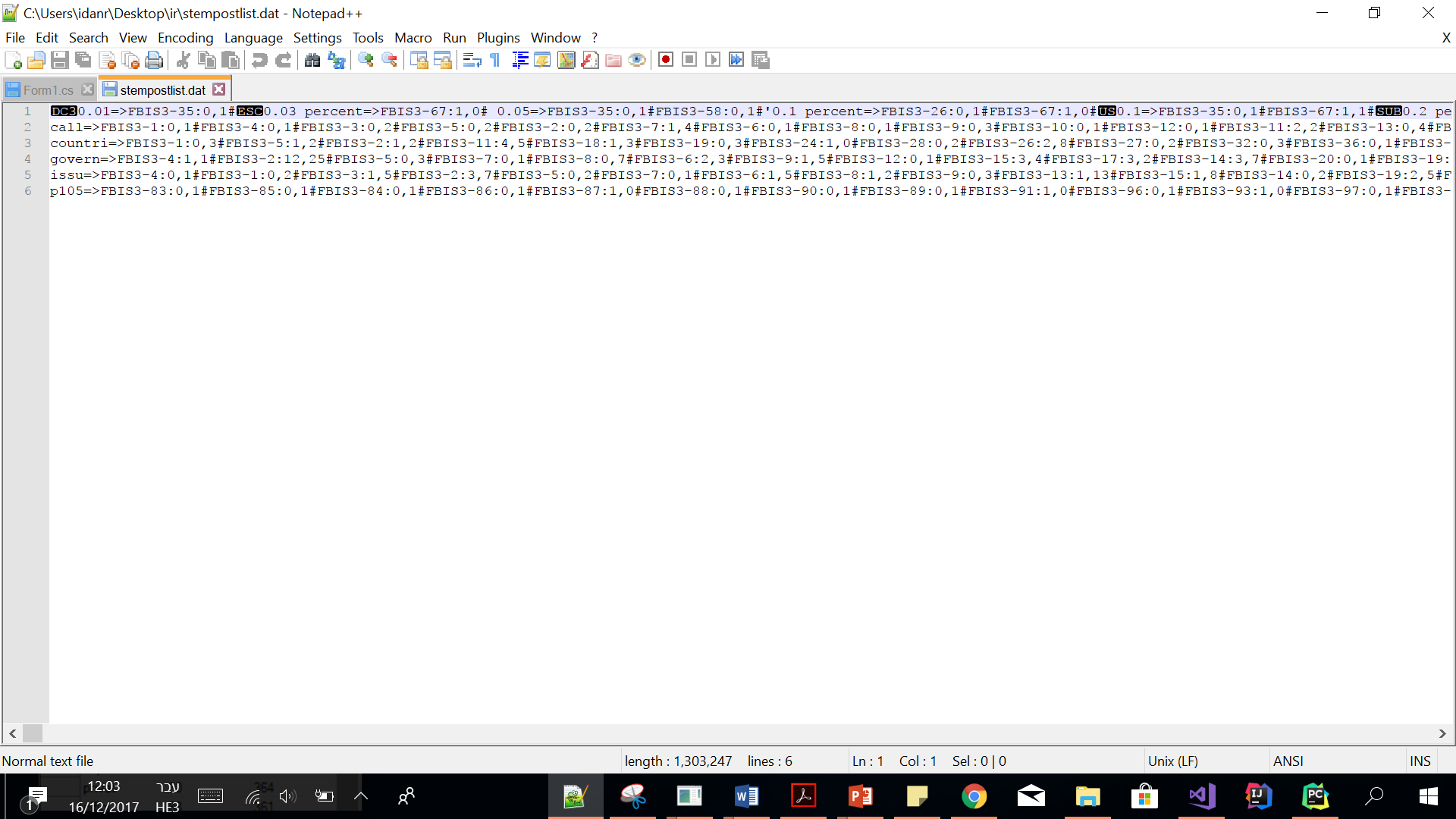
לאחר שלב בניית הposting הזמניים , אנחנו מחזיקים במילון הסופי שלנו 10,000 מילים (ללא שורת הpost הסופי שיקבע לאחר המיון מיזוג) ולכן בשלב זה אנחנו מבצעים מיון ע"פ ערך הdf של כל term ומכניסים למילון רק את 10000 המילים הכי נפוצות. בשלב המיזוג נעדכן את מילון המטמון ונכניס גם את רשימת הפוסט של כל term. בנוסף נעדכן בעזרת ערך בוליאני במילון הראשי שהterm הנוכחי נמצא בcache ולא בדיסק

* 1. **מבנה קובץ הפוסטינג :** "=>"+docid:data "#" term. **סוג הקובץ - בינארי**

כאשר "#" משמש כדלימטר בין המסמכים ו "=>" משמש כדלימטר בין terms. לדג' father=>doc0: data # doc1 :data

הData שנשמור על כל term בpostlist הוא: שם המסמך, כמות המופעים שמחולקת לשני מספרים: הראשון יהוו את כמות המופעים בתחילת המאגר (30% הראשון של המאגר)

וחלק השני יהוו את כמות המילים בשאר המאגר (70% הנותרים של המאגר) והרעיון הוא לתת חשיבות גדולה יותר למילים שמופיעות בתחילת המאגר שהן כנראה יותר רלוונטיות



* 1. חוקי parse שנוספו:

חוק ראשון: שני מילים צמודות המכילות / ביניהם יופרדו לשתי מילים נפרדות

דוגמא: dog/cat => dog , cat

חוק שני: שני מילים צמודות המכילות - ביניהם יופרדו לשתי מילים נפרדות

דוגמא: dog-cat => dog , cat

חוק שלישי: כל תו שאינו ספרה או אות ואינו מוכל בחוק אחר ימחק

דוגמא: d#og. => dog (1.253 עדיין יהפוך ל-1.26)

1. **פירוט המאגר:**
   1. כמות terms שונים במאגר לפני stem: 2,799,163
   2. כמות terms שונים במאגר אחרי stem: 2,572,100.
   3. כמות terms שהם מספר במאגר : 2,572,100 מילים שונות במאגר.
   4. כמות מסמכים שאנדקסו : 472493
   5. **עשרת הterms השכיחים ביותר :** STEM
      * year,726142
      * cellrul,718012
      * tablecel,578744
      * mr,484413
      * govern,392522
      * state,389849
      * cent,385220
      * compani,371894
      * market,343647
      * time,338508

קשה להסביר מדוע הופיעו כל מילים בגלל הstemer וגם מכיוון שלא קבלנו מידע לגבי נושא הקורפוס אבל מחיפוש בגוגל ומציאת האתר הבא שמדבר על ניתוח שביצעו ב אוקספורד על המילים הנפוצות בשפה האנגלית :

<http://www.businessinsider.com/zipfs-law-and-the-most-common-words-in-english-2013-10>

If we consider "content words" (words with tangible meaning) instead of "function words," the top ten list changes to include: "time," "person," "year," "way," "day," "thing," "man," "world," "life," and "hand."

ניתן לראות שהניתוח שלנו מצא את המילים year וtime ב10 המילים הכי שכיחות. תיאור זמנים הוא משהו שמאוד נפוץ בשיח יומיומי ולכן כנראה גם במאמרים השימוש בזמנים נפוץ בשל כך הגיוני שיופיע בטופ 10.

בנוסף המילים market , government , company מעיד שהקורפוס מכיל ידיעות פוליטיות/כלכליות וכלן הגיוני שיופיעו במילים הנפוצות.

* 1. **עשרת הterms הכי פחות נפוצים:** STEM
     + pretoria england,1
     + anthony ridd,1
     + bradman but,1
     + bradman th,1
     + pele don,1
     + yes concord,1
     + lara thei,1
     + legge braik,1
     + braik,1
     + proctor perhap,1

יש עוד המון מילים שמופיעות פעם אחת בלבד (563968) . כל המילים פה הן מילים שחוברו יחדיו בגלל שהופיעו באות גודלה במקור ולכן זה ביטויים שהיו במאגר והגיוני שיופיעו פעם אחת.

* 1. **עשרת הterm הכי נפוצים מבחינת DF:** STEM
     + year,241117
     + time,166007
     + state,141789
     + govern,139815
     + make,135979
     + report,126718
     + nation,121283
     + peopl,121066
     + compani,120882
     + mr,119993

ניתן לראות התאמה גבוהה עם עשרת המילים השכיחות ביותר . אבל גם המילה people הופיע שמתקשרת למילים שהופיעו במאמר שציינתי מקודם כמו : person , man. בנוסף המילה nation מתקשרת יחסית למילה world שהופיעה במאמר גם.

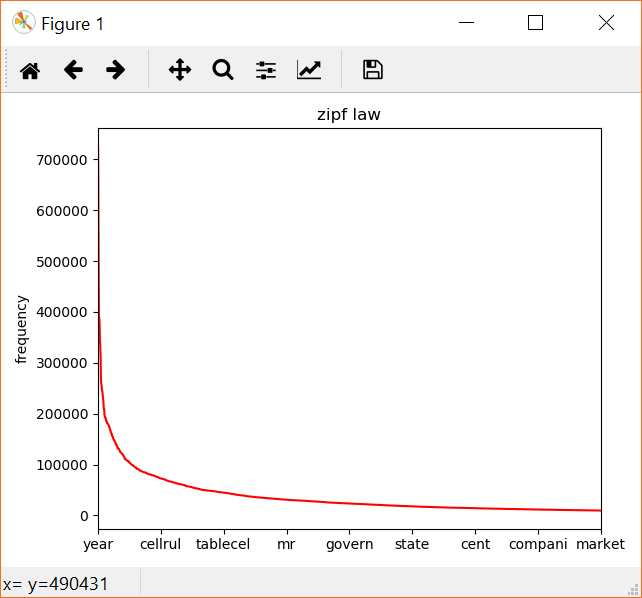
בנוסף המילה time זינקה למקום השני, דבר שמתחבר למאמר שציינתי מקודם שהמילה time נפוצה בשפה האנגלית ע"פ ממד zipf law.

* 1. **עשרת המילים הפחות נפוצות מבחינת df:**
     + martin stanbridg,1
     + mr bellfield,1
     + bellfield,1
     + birmingham unlik,1
     + 99775,1
     + 42915m
     + 43446m
     + englanders w,1
     + 39798,1
     + 34.m1

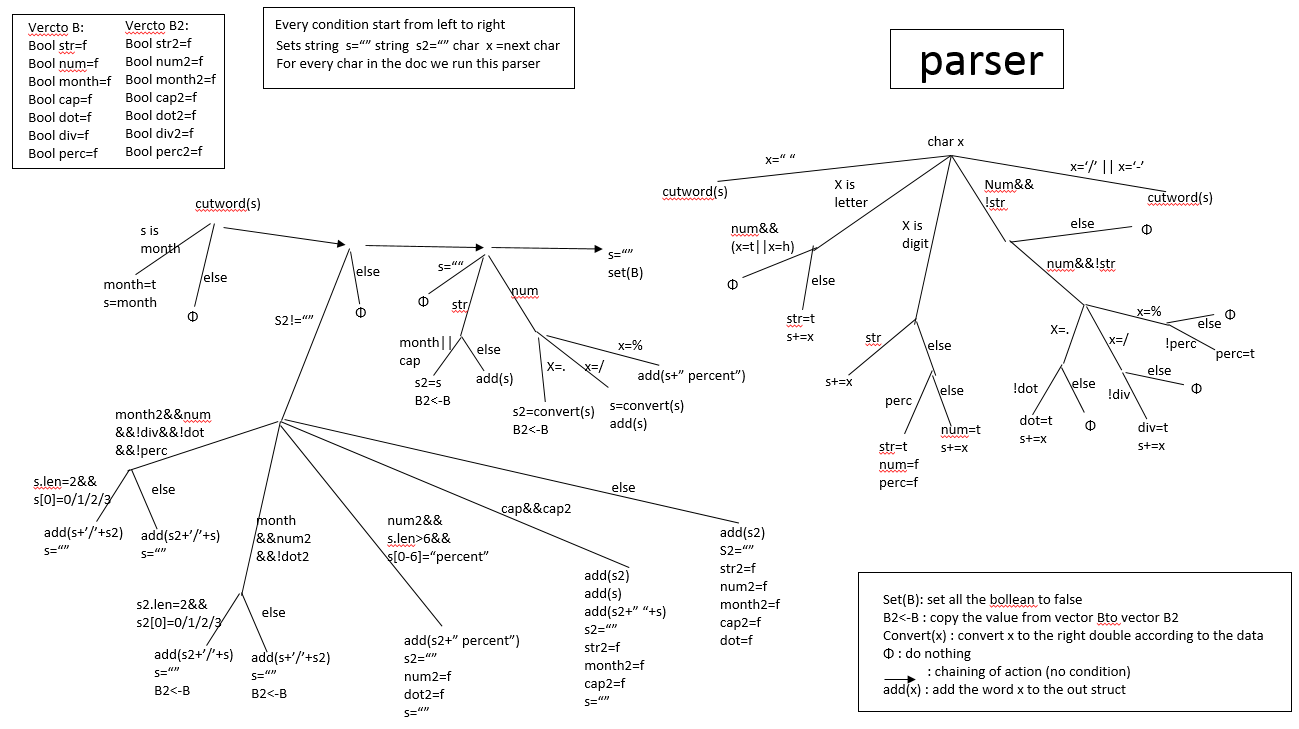
כמו בממד הקודם של tf המון מסמכים הופיעו במסמך אחד בלבד ונבחרה קבוצה באופן שרירותי. ניתן לראות מספרים גדולים וציון m של מטרים . בסה"כ הגיוני שיופיעו פעם אחת.

חיפוש בגוגל של המילה bellfield מוצאת פרמת ביגוד או רוצח סדרתי בשם בלפיד לוי . הגיוני בסה"כ שיופיע פעם אחת בלבד.

* 1. **גרף zipf law:**



* + - אמנם קשה לראות זאת בגרף כיוון שהוא נוצר על כמות מילים גדולה אך גרף זה יצא כמו הגרף הנלמד בשיעור.
    - הגרף חושב על כלל המאגר אבל מציג עד 10 המילים הנפוצות ביותר
  1. **גדלי posting מילון וcache: עם stem**
     + postlist file size :1210626[kb]
     + dictionary file size :152593[kb]
     + cache file size :217612[kb]
  2. **גדלי posting מילון וcache: בלי stem**
     + postlist file size :1307103[kb]
     + dictionary file size :170167[kb]
     + cache file size :219720[kb]

**נספחים:**

1. **פסודו קוד לאלגוריתם מיון:**
   * + - הפונקציה תפתח מצביעים לכל קבצי הposting הזמנים ותקרא 3 שורות מכל קובץ (אילוץ הפרויקט).
       - הפונקציה תכניס את השורות לתור עדיפויות + לכל שורה מאיזה קובץ היא נקראה.
       - S = queue.deque // הterm הקטן ביותר מכל הקבצים
       - כל עוד התור לא ריק:
         1. G = queue.deque
         2. אם g == s : s=s+g
         3. אחרת:

תקרא שורה z מהקובץ של s

כל עוד הz == s

S=s+z

אחרת : queue.add(z) // משמע מצאנו את כל השורות של s

**תכ**תוב את s לקובץ הסופי.

תעדכן את s להיות ל g