**דוח פרוייקט מנוע אחזור - חלק א'**

**מגישים**

**עומר נגר 307937714, אסף זקס 302329693**

**פרק 1 - עיצוב התוכנה**

1. המנוע קורא את קבצי הטקסט במטרה לבנות מילון אחזור. כל טקסט וכותרת עוברים parse ולבחירת המשתמש גם stem. מופעי המילים נשמרים בsegmentFiles בדיסק הקשיח לאורך פעולת המחלקות הללו, כדי למנוע עומס על הזיכרון הראשי. בשלב הבא חבילת האינדקסר פותחת את הקבצים שנוצרו, מרכזת את המידע על כלל מופעי כל מונח שנקרא, ושומרת אותו לקבצי פוסטינג תוך בניית מילון מונחים. פתיחת קבצי הסגמנט ויצירת קבצי הפוסטינג מתקיימים עבור חלק מהמילים בכל פעם.

**Package Parser:**

**Class Master:**

המחלקה מרכזת את עבודת המחלקות readfile, הparser, הstemmer ומרכזת את יצירת קבצי הסגנמט.

**פונקציות המחלקה:**

run –

הפונקציה מנהלת את עבודת המחלקות השונות בחבילה. מפעילה את readFile שקוראת את הטקסטים לתוך docTexts, אותם היא תשלח לparser כדי לקבל docMD עם המידע על המסמך והמילים השונות בו. הפונקציה מקבלת כארגומנט Boolean הבורר בין הפעלת הסטמר לביטולו.

getDocAmount-

הפונקציה מחזירה את מספר הdocMD השמורים בזיכרון.

saveDocMD-

הפונקציה שומרת את מאגר המידע על המסמכים בדיסק הקשיח.

LoadDocMD-

הפונקציה מחזירה את מאגר המידע על המסמכים מהדיסק הקשיח.

**Class ReadFile:**

המחקלה הקוראת את הקבצים הדרושים לבניית קבצי הסגמנט.

**פונקציות המחלקה:**

getAllPaths-

הפונקציה בודקת את תקינות הנתיב שהתקבל עבור הקורפוס, בודקת כי קיימת תיקיית קורפוס וקובץ stopword, שומרת את הנתיב אליהן ומפעילה שיטה לעיבוד כלל הכתובות של הקבצים בקורפוס.

readCorpus-

הפונקציה עוברת על תייקית הקורפוס ושומרת במבנה נתונים את כלל הנתיבים לכלל הקבצים בתיקייה.

readStopWords-

הפונקציה מחזירה hashSet של כל הstopwords מהקובץ.

handleFile-

הפונקציה מטפלת בקובץ html, מאתרת מסמכים,כותרות ומספרי מסמכים ומתייגת אותם במבנה נתונים להמשך.

**Class Parser:**

מייצגת אובייקט מטיפוס parser, מייצרת אובייקטים מטיפוס docMD. כל אובייקט אחראי על פרסור מסמך בודד.

**פונקציות המחלקה:**

setMonths-

הפונקציה בונה עבור האובייקט מילון של שמות חודשים והערך אליו יש להחליף אותם על פי החוקים.

handleDoc-

הפונקציה יוצרת אובייקט מטיפוס docMD, אליו שומרת את כלל המידע על המסמך והמילים הקיימות בו באמצעות קריאה לפונקציות עזר.

calcMaxTF-

הפונקציה מחשבת מהו הערך המקסימלי של הופעות של מילה במסמך.

maxFreqTerm-

הפונקציה מחזירה את המילה הראשונה בעלת TFMax.

parse-

הפונקציה המבצעת את פעולת הparsing בפועל, עוברת על טקסט, מפרקת אותו לשורות אותן תפרק למילים ותחפש תבניות על פי חוקי הפרוייקט וכללים שהוספנו שיפורטו בהמשך.

wordToNum-

הפונקציה ממירה מילה לערך מטיפוס double.

finalEdit-

הפונקציה מבצעת אופרציות טקסט אחרונות על מילים שלא טופלו ע"י אף חוק, מוחקת תווים מיותרים שאינם טקסטואלים, ומפעילה stem במידה ונבחר.

addToWords-

הפונקציה מוסיפה את המילה לרשימת המילים של המסמך.

numToString-

הפונקציה מבצעת אופרציות טקסט על מספרים, מורידה דיוק של מספר לא שלם לעד 3 ספרות אחרי הנקודה.

handleNumber-

הפונקציה מטפלת במספרים על פי החוקים בהוראות הפרוייקט ומחזירה true במידה והערך התקבל למילון.

**Class Stemmer:**

**Open source. URL :** [**https://tartarus.org/martin/PorterStemmer**](https://tartarus.org/martin/PorterStemmer)

המחלקה מבצעת stem למילה ע"פ חוקי הPorter Stemmer. תיאור האלגוריתם והמימוש בקישור המצורף.

**Class DocMD:**

מחלקה המייצגת מידע על מסמך. מחזיקה מצביע למאגר המושגים שנמצאו במסמך לזמן קצר לטובת יצירת קבצי הסגמנט.

**פונקציות המחלקה:**

toString -

הפונקציה משרשרת את המידע השמור על המסמך באמצעות פסיק ומחזירה את השרשור.

**Class DocText:**

מחלקה המייצגת מסמך גולמי. מחזיקה מזהה מסמך, כותרת וטקסט גולמי.

**פונקציות המחלקה:**

getHeader –

הפונקציה מחזירה את כותרת המסמך.

getDocno –

הפונקציה מחזירה את מזהה המסמך.

getInnerText –

הפונקציה מחזירה את הטקסט הרשום במסמך.

**Package Indexer:**

**Class IndexDictionary:**

המחלקה אחראית על יצירת האינדקס - המילון וקבצי הפוסטינג.

**פונקציות המחלקה:**

createIndexer -

הפונקציה מקבלת טבלת האש המכילה מונחים, עוברת על כל ערכי הטבלה, יוצרת קבצי פוסטינג למונחים ומוסיפה את המונח למילון, לאחר שהגיעו 200 מונחים, תדפיס את קובץ הפוסטינג לזיכרון המשני ותפנה את הזיכרון הראשי.

outputPostList -

הפונקציה מקבלת נתיב לשמירת קובץ הפוסטינג, פותחת קובץ חדש בנתיב זה, ומדפיסה לקובץ את כל ערכי הפוסטינג על כל מונח שיצרנו עבורו פוסטינג.

getIndexerPrint –

הפונקציה עוברת על כל המילון ויוצרת מחרוזת ארוכה שמכילה צמדים של מילה וכמות המופעים הכולל שלה, לטובת ההדפסה למשתמש.

saveToDisk –

הפונקציה שומרת את המילון לזיכרון בצורת קובץ טקסט.

loadDictionary –

הפונקציה טוענת את מילון מהזיכרון, מקבלת נתיב למיקום הפוסטינג, וערך בוליאני האם בוצע סטמינג או לא.

getNumOfUniqueTerms –

הפונקציה מחזירה את גודל המילון = כמות המילים השונות במילון.

Comp –

קומפרטור עבור המבנה של המילון – מפת עץ, ממיין לקסיקוגרפית כאשר אין חשיבות לcase.

**Class Posting:**

המחלקה מחזיקה מידע לטובת posting של term בודד.

**פונקציות המחלקה:**

addToPostingList –

הפונקציה מעדכנת עבור מונח את הכתובת שבה צריך להשמר, ומחזירה את המונח.

getPosting –

הפונקציה מחזירה רשימה משורשרת של כל הפרטים על המופעים של המונח.

getMetaOfTerm –

הפונקציה מחזירה את הפרטים על המונח, שישמרו עבור כל מונח בקובץ הפוסטינג שלו.

getPath –

הפונקציה מחזירה את הנתיב של קובץ הפוסטינג שהמונח שמור בו.

**Class SegmentProcesses:**

המחלקה מטפלת בקבצי סגמנט ויוצרת מהם מונחים.

**פונקציות המחלקה:**

processCorpus –

הפונקציה עוברת בלולאה על כל קבצי הסגמנט שנוצרו ע"י הפרסר, קוראת את הסגמנט מהזיכרון, ושולחת את כל המילים שנשמרו בסגמנט לעיבוד – הפיכה לאובייקט Term ועיבודים נוספים, לאחר מכן שולחת את כל המונחים לאינדקסר ומשם למילון. בסיום שומרת את המילון שנוצר לתיקייה שמכילה את הפוסטינג.

processSegment –

עבור כל מילה שחזרה מהסגמנט, הפונקציה מבצעת את החוק של האותיות הגדולות – ומעדכנת בהתאם את המילה, כמו כן יוצרת אובייקט Term מכל מילה ומוסיפה אותו לרשימה שתלך לאינדקסר. בנוסף גם קוראת לפונקציה המעדכנת את המופעים של מונח במסמכים.

addOccurrenceToTerm –

הפונקציה מוסיפה מופע נוסף למונח אם המונח כבר נצפה, מקבלת את פרטי המופע וקוראת לפונקציה במחלקה Term עם הפרטים כארגומנטים.

updateDocFreq –

הפונקציה מעדכנת את תדירות המופעים במסמכים שונים, קוראת לפונקציה במחלקה Term.

updateCaseToUpper –

הפונקציה מעדכנת את הcase של מונח שמופיע רק עם אות ראשונה גדולה. עבור כל המפתחות בטבלת המונחים שנוצרו, מפעילה על כל מונח מטודה בוליאנית של המחלקה Term, במידה וחזרה תשובה חיובית, מוחקת את המפתח הקודם, ומוסיפה את המונח מחדש עם אותיות גדולות בלבד.

getTheDictionary –

הפונקציה מחזירה את המילון.

**Class Term:**

המחלקה מייצגת מונח במילון.

**פונקציות המחלקה:**

addOccurrence –

הפונקציה מקבלת פרטים על מופע של מונח, ומוסיפה את המופע לרשימת המופעים של כל מונח.

updateDocFq –

הפונקציה מעדכנת את כמות המסמכים השונים שהמונח מופיע בהם.

updateToUpperCase –

הפונקציה מעבירה את המונח לאותיות גדולות אם האות הראשונה הינה אות גדולה, מחזיר אמת אם השתנה.

getOccurrence –

הפונקציה מחזירה את הרשימה המשורשת של כל מופעי המונח.

getDocFq –

הפונקציה מחזירה את כמות המסמכים שהמונח מופיע בהם.

getTotalFq –

הפונקציה מחזירה את כמות המופעים הכוללת של המונח.

addToTopBottom –

הפונקציה מוחשכת (בהערה), כאשר פעילה שומרת את 10 המונחים השכיחים ביותר ו10 המונחים הנדירים ביותר בשני רשימות שונות, משתמשת לצורך כך גם ב2 קומפרטורים.

**Class TermInDoc:**

המחלקה מייצגת את המידע על כלל מופעי המילה במסמך.

**פונקציות המחלקה:**

setTerm –

הפונקציה מקבלת מילה ומעדכנת אותה למונח במסמך.

updateCapsToLower –

הפונקציה מעדכנת את המילה להיות עם אותיות קטנות בלבד.

getDocNo –

הפונקציה מחזירה את שם המסמך בו המונח מופיע.

getTermfq –

הפונקציה מחזירה את תדירות המונח במסמך.

isHeader –

הפונקציה מחזירה אמת אם המונח מופיע בכותרת המסמך.

isEntity –

הפונקציה מחזירה אמת אם המונח הינו יישות במסמך.

getTerm –

הפונקציה מחזירה את המונח במסמך.

**Class TermInDocList:**

המחלקה מייצגת מבנה נתונים לניהול termInDocs.

**פונקציות המחלקה:**

tidToJson –

הפונקציה מקבלת מספר בין 1-20 שתשתמש בו באינקס למיקום שמירת הסגמנט, ועוברת על רשימה של אובייקטי TermInDoc ותוסיף אותם לאובייקט JSON לאחר מכן תדפיס את אובייקט הJSON לזיכרון המשני לפי האינדקס.

JsonToTid –

הפונקציה מקבלת כקלט כתובת, וטוענת מכתובת זו אובייקט JSON. לאחר מכן עוברת על האובייקט יוצרת ממנו אובייקטים של TermInDoc, לבסוף תחזיר רשימה של מונחים במסמך.

getList –

הפונקציה מחזירה רשימה משורשרת של מונחים במסמך.

setList –

הפונקציה מחליפה את הרשימה המשורשת של המונחים במסמך ברשימה חדשה.

Class MyModel:

המחלקה אחראית על חיבור ממשק המשתמש לפונקציונאליות של המנוע.

connectToCorpus – הפונקציה מקבל כתובת לקבצי הקורפוס, וכתובת יעד לקבצי הפוסטינג, עוברת על כל קבצי הקורפוס הנתונים ומייצרת מהם מילון בעזרת המחלקות Master וSegmentProccesses, ושומרת את קבצי הפוסטינג לכתובת הנתונה לדבר, בסיום מדפיסה פרטים על התהליך לחלונית חדשה.

bringUpDictionary – הפונקציה קוראת למטודה loadDictionary ולמטודה loadDocMD ואחראית לטעינת המילון של המונחים והמילון של המסמכים.

changeStemmerMode – הפונקציה אחראית לשנות את המצב של המשתנה הבוליאני stemmer.

forgetDictionary – הפונקציה אחראית לפינוי הזיכרון של המערכת מהמילון שמחזיק

getPostingPa – הפונקציה מחזירה את הכתובת לשמירת הפוסטינג.

getCorpusPa – הפונקציה מחזירה את הכתובת לטעינת הקורפוס.

getDictionaryToPrint – הפונקציה מחזירה סטרינג שמכיל הדפסה של כל המילון.

setCorpusPa – הפונקציה מאפשרת השמה של כתובת חדשה לטעינת הקורפוס.

setPostingPa – הפונקציה מאפשרת השמה של כתובת חדשה לשמירת הפוסטינג.

isStemmer – הפונקציה מחזירה אמת אם מצב הStemmer פועל.

getDictionary – הפונקציה מחזירה את המילון.

getInfoOnRun – הפונקציה מחזירה פרטים על הריצה והיצירה של התהליך.

Class Controller:

המחלקה אחראית על הפונקציונאליות של כל הכפתורים במסך הראשי של הממשק.

pressSet – הפונקציה מקבלת אירוע לחיצה, ופותחת חלונית חדשה שבה ניתן יהיה להזין כתובות למערכת.

pressConnect – הפונקציה מקבלת אירוע לחיצה, ומפעילה את הפונקציה connect במחלקה MyModel.

pressLoad – הפונקציה מקבלת אירוע לחיצה, ומפעילה פונקציה שטוענת את המילון מכתובת הפוסטינג.

pressDisplay – הפונקציה מקבלת אירוע לחיצה, ופותחת חלונית חדשה שבה יוצג המילון.

pressStemming – הפונקציה מקבלת אירוע לחיצה, ומשנה את המצב של המשתנה הבוליאני stemmer.

pressClear – הפונקציה מקבלת אירוע לחיצה, ניגשת לכתובת השמורה של הפוסטינג ומוחקת את כל התוכן בתיקייה.

showAlert – הפונקציה פותחת חלונית אזהרה עם תוכן לבחירה שניתן כקלט.

setModel – הפונקציה מקבלת אובייקט model ומשימה אותו.

Class ConnectController:

המחלקה אחראית על הפונקציונאליות של כל הכפתורים במסך בחירת הכתובות של הממשק.

browseCorpus – הפונקציה מקבלת אירוע לחיצה, פותחת חלונית לבחירת הכתובת של הקורפוס.

browsePosting – הפונקציה מקבלת אירוע לחיצה, פותחת חלונית לבחירת הכתובת לשמירת הפוסטינג.

pressAccept – הפונקציה מקבלת אירוע לחיצה, סוגרת את חלונית בחירת הכתובות.

getModel – הפונקציה מחזירה את אובייקט המודל.

setModel – הפונקציה מאפשר השמה לאובייקט המודל.

Class DisplayController:

המחלקה אחראית על הפונקציונאליות של כל הכפתורים במסך בחירת הכתובות של הממשק.

displayDictionary – הפונקציה מקבלת אירוע לחיצה, ומדפיסה לתצוגה את המילון.

setModel – הפונקציה מאפשר השמה לאובייקט המודל.

1. התמודדות עם מגבלת הזיכרון:

גודל הקורפוס אינו מאפשר עבודה עם כלל הקבצים בו זמנית. על מנת לשמור על הזיכרון הראשי פנוי לאורך כל ריצת התוכנית, שמרנו בשלב הparse עבור הקבצים כולם את הנתיב עצמו בלבד. בטיפול במסמכי קובץ ספציפי שמרנו את המידע הגולמי של הקובץ רק בזמן עיבודו ודאגנו לשחרר מצביעים לאחר סיום השימוש, לטובת עבודת הgarbage collector. לאחר סיום הparse של כל מסמכי הקובץ, שרשרנו את המונחים השונים שנמצאו במסמכי הקובץ ב20 קבצי segment המנוהלים לפי חישוב ערך הhash של המונח באותיות קטנות. כך הבטחנו שכל מופעי המילה יכנסו לאותו הקובץ עבור כלל המסמכים בקורפוס, ובכך הבטחו שהאינדקסר יעבד רק 1/20 מהמידע בכל שלב, מבלי להחזיק את כל קבצי הsegment (ובשלב האינדקס – הposting) באותו הזמן פתוחים. מצאנו ששמירת כלל מופעי המילים של file מסויים בכל פעם יעילה כמעט כמו צבירת מסמכים ומאפשרת עבודה מסודרת ועל כן בחרנו בה. הבחירה בערך 20 נמצאה כיעילה בזמן העבודה עצמה.

1. קבצי הPosting מכילים את כל המידע שאנו שומרים על המונחים. בנינו אותם בצורה שכל קובץ מכיל מידע על 200 מונחים שונים, אנו שומרים את הקבצים הללו כקבצי txt ויש לנו 11,763 קבצים כאלו, בחרנו דווקא ב200 מכיוון שהבחנו ביעילות בזמן השמירה. עבור כל מילה, אנו שומרים שורה לפרטים שלה, ושורה עבור הפרטים של כל מופע שלה בקורפוס.
2. החלטנו ליצור את הקבצים ההופכיים בצורה הדרגתית מקבצי סגמנטציה. קבצי הסגמנטציה נבנים בצורה הדרגתית בשלב הפירסור, ישנם 20 קבצי סגמנטציה שווים בגודלם, כאשר כל מילה נבחרת לקובץ ע"י ביצוע פונקציית אש, דבר זה מבטיח שכל מילה תופיע ביחד עם כל המופעים שלה באותו קובץ הסגמנטציה. בחרנו במספר 20 מכיוון שמצאנו במספר זה כיעיל מבחינת זכרון וזמני ריצה. אנו עוברים על קובץ סגמנטציה, אוספים את כל הכפילויות בעזרת טבלת אש, ולאחר מכן מעבירים כל רשומה מטבלת האש לעץ מיון, ובכך מחזיקים את המילון ממויין.

קבצי הפוסטינג נכתבים לזיכרון לאחר כל הכנסה של 200 מילים למילון.

1. המידע הנוסף אותו בחרנו לשמור:

* עבור כל מופע של מונח במסמך תיעדנו האם הוא מכותרת המסמך או מהתוכן עצמו, מתוך מחשבה שלהופעת מילה בכותרת משקל רב יותר מבטקסט עצמו, שכן תפקידה לתת את עיקרי הדברים.
* עבור כל מסמך שמרנו את המילה בו שחזרה הכי הרבה פעמים שאינה stopword, שכן יכולה להיות מוטיב במסמך ולתת לו זיקה למילה.

1. חוקים שנוספו לparser:

* עבור number grams נשמור number GR, וכן עבור number kilograms נשמור number\*1000 KG

דוגמאות לשימוש בdataset:

* במידה ומילה מסתיימת ב’s נגזור את הסיומת.

דוגמאות לשימוש בdataset:

1. לאורך העבודה על הparser מצאנו כי קיימים מונחים רבים המכילים סימנים במקום רווחים בניהם או סימני הפרדה בניהם. הוספנו תנאים על מנת להתגבר על תוספות אלו במטרה להפריד את המילים שנמצאות בניהן ולאנדקס גם אותן.

תוסיף משהו אם אתה מוצא לנכון

1. שימוש בקוד פתוח וחבילות חיצוניות:

* Porter Stemmer

URL: [**https://tartarus.org/martin/PorterStemmer**](https://tartarus.org/martin/PorterStemmer)

הקוד מהאתר הועתק (המחלקה stemmer), לטובת ביצוע stemming.

* JSON jar

URL: <https://jar-download.com/artifacts/org.json>

השתמשנו במחלקה לטובת יצירת אובייקט Json לשמירה לזיכרון.

* JSON Simple jar

URL: <https://jar-download.com/artifacts/com.github.cliftonlabs/json-simple/2.3.0/source-code>

השתמשנו במחלקה לטובת יצירת אובייקט Json לשמירה לזיכרון.

עומר הסבר על jsop jar