דו"ח מנוע חיפוש

סעיף 1 a

הסבר על אופן פעולת התכנית

התכנית מורכבת מ-3 תיקיות:view, model, view-model ומחלקת main המתפעלת הכל.

1. **ה-view** אחראי על הצגת הנתונים ל-GUI ומקשר בין ה-fxml והפונקציות אליהן הוא מקושר.
2. **ה-model** אחראי על הלוגיקה, הכולל בניית המילון ואינדוקס.
3. **ה-view-model** מקשרת בין שכבת ה-view ושכבת ה-model.
4. **המain** אחראי על הפעלת החלון הראשון של ה-GUI בו יוכל המשתמש לבחור מה לעשות.

view

מכילה את המחלקות הבאות:

1. קובץ fxml בשם **myView2** המחובר למחלקת myController
2. **מחלקת myController** - מוגדרת כ"צופה" על מחלקת view-model. מפעילה את המתודות לאחר שנלחץ הכפתור המתאים ב-GUI. המחלקה מכילה את השיטות הבאות:
   a. שיטות set-רים:
      i. שיטת setViewModel - מאתחל את שדה myViewModel המקשר אותנו למחלקה view-model.
      ii. שיטת setStage - מגדיר את ה-stage שקיבלנו מה-main.
      iii. שיטת setPathToRead - מעדכן את הנתיב לקריאה לפי הטקסט שהכניס המשתמש (במידה ובחר הכנסה ידנית לתיבת טקסט).
      iv. שיטת setPathToWrite - מעדכן את הנתיב לכתיבה לפי הטקסט שהכניס המשתמש (במידה ובחר הכנסה ידנית לתיבת טקסט).
      v. שיטת setToStem - מעדכן את השדה הבוליאני toStem לפי ה-checkBox לפי בחירת המשתמש.
   b. שיטת update - כחלק מהיותה תיקייה "צופה", מימשנו את המחלקה הזו שנועדה לעדכן פרטים חשובים מהשכבה הלוגית. אנו מעדכנים את שדה ה-dictionary כאשר הוא משתנה בשכבת ה-model. מדובר במילון שנמצא בזיכרון ram.
   c. שיטת loadPathToRead\loadPathToWrite - בעת לחיצה על כפתור ה-"browse" ונבחרת תיקיה המיועדת לקריאה/כתיבה של קבצים, השיטה שומרת את הנתיבים שהוכנסו בתוך שדות המחלקה. השיטה כתובה פעמיים, פעם אחת עבור נתיב לקריאת קבצים ופעם שניה עבור נתיב לכתיבת קבצים.

d. שיטת setDictionary - השיטה שקוראת לשכבה הלוגית לבנות את המילון. מתרחשת כאשר המשתמש לוחץ על כפתור "build Dictionary". השיטה בודקת תחילה האם הנתיב שהוכנס הוא נתיב טוב (כלומר נתיב חוקי המהווה תיקיה), אם לא מציג הודעת שגיאה. לאחר מכן מפעיל את השיטה בשכבת view model שמפעיל את השיטה בשכבת model. בסוף השיטה נפעיל את 3 הכפתורים האחרים של ה-GUI האחראים על מחיקת מידע, טעינת מילון, הצגת מילון.
e. שיטת deleteData -אחראית על מחיקת כל הנתונים שהוספנו לדיסק הקשיח וגם את המילון מה-ram. קוראת לשיטה המיועדת מ-viewModel ומפועלת בעת לחיצת הכפתור הרלוונטי ב-GUI.
f. שיטת showDictionary - נועדה להציג את המילון למשתמש. קוראת לשיטה המיועדת מ-view-model.
g. שיטת loadDictionary - מופעל בעת לחיצת הכפתור המתאים ב-GUI, אחראית על טעינת מילון מהדיסק. קוראת לשיטה המתאימה ב-viewModel.

viewModel

מכילה את המחלקה viewModel.

1. **מחלקת viewModel** - מוגדרת כ"צופה" על מחלקת model וכ-"נצפת" ע"י מחלקת view. מכילה את השיטות הבאות:
   a. שיטות set-רים:
      i. שיטות setPathToRead\setPathToWrite - מאתחלת את השדות המייצגים נתיבים לקריאה/כתיבה אשר קיבלת משכבת ה-view.
      ii. שיטת viewModel - בנאי המאתחל את המחלקה עם השדה model אותו קיבל מה-view.
      iii. שיטת setToStem - מעדכן את שדה toStem לפי מה שקיבל משכבת view.
   b. שיטת getDictionary - מחזירה את המילון השמור במחלקה זו.
   c. שיטת showDictionary - שיטת ביניים בין 2 השכבות. קוראת לשיטה המתאימה ממחלקת model.
   d. שיטת update - כחלק מהיותה תיקייה "צופה", מימשנו את המחלקה הזו שנועדה לעדכן פרטים חשובים מהשכבה הלוגית. אנו מעדכנים את שדה ה-dictionary כאשר הוא משתנה בשכבת ה-model. מדובר במילון שנמצא בזיכרון ram.
   e. שיטת setDictionary - שיטה המקשרת בין בחירת המשתמש ב-GUI וב-view לבנות מילון לבניית המילון ב-model. השיטה קוראת לשיטה המתאימה בשכבת ה-model.
   f. שיטת clearData - שיטה המקשרת בין בחירת המשתמש ב-GUI וב-view למחיקת כל הנתונים והמילון לבין השיטה שעושה זאת בפועל ב-model. השיטה קוראת לשיטה המתאימה בשכבת ה-model.
   g. שיטת loadDictionary - שיטה המקשרת בין בחירת המשתמש ב-GUI וב-view לטעינת המילון מה-GUI לבין השיטה שעושה זאת בפועל ב-model. השיטה קוראת לשיטה המתאימה בשכבת ה-model.

model

התיקיה האחראית על כל הלוגיקה של בניית מנוע החיפוש. במחלקה זו נקרא את הקורפוס, נפרסר אותו לפי החוקים הנתונים לנו ולאחר מכן נבנה inverted index. מי שמנהל את התקשורת עם שכבות ה-view וה-view model היא מחלקת myModel אשר "נצפת" ע"י viewModel. לשם בניית המילון, פרסור ואינדוקס, המחלקה myModel קוראת ל-indexer שעושה את העבודה בפועל. השכבה מכילה את המחלקות הבאות:

1. **מחלקת termData** - המחלקה הזו שומרת נתונים חשובים אודות ה-term, כאלה אשר לא דורשים מקום רב של אחסון. המחלקה מכילה את השדות הבאים:
   a. שדה numOfDoc - כמות המסמכים בהם מופיע המילה.
   b. שדה pointerLine - מספר השורה שבה מופיע ה-term ב-posting file.
   c. שדה termName - שם ה-term.
   d. שדה totalAppearance - סה"כ כמות המופעים של המילה בקורפוס.

   *המחלקה מכילה בנאי, שיטות set-רים ושיטות get-רים.
2. **מחלקת DocDetails** - כחלק מתהליך הפרסור אנחנו בונים גם מבנה נתונים ששומר מידע על המסמכים עצמם, כגון כמות מילים יחודיות במסמך ומספר המופעים של המילה הנפוצה ביותר. המחלקה הזו מייצגת אובייקט המחזיק מידה אודות מסמך. מכילה את השדות:
   a. שדה Max_tf - מייצג את מספר המופעים של המילה השכיחה ביותר במסמך.
   b. שדה uniqeWords - מייצג את כמות מילים יחודיות במסמך.
   c. שדה docno - מייצג את שם המסמך.
   d. שדה docSize - מייצג את אורך המסמך
   e. שדה YeshutNumber - מייצג את מספר הישויות במסמך.

   *המחלקה מכילה בנאי, שיטות set-רים ושיטות get-רים.
3. **ממשק IModel** - ממשק המכיל את כל השיטות של מחלקת myModel.
4. **מחלקת ReadFile** - המחלקת מקבלת את הנתיב לקורפוס, מפרקת אותו לפי תגיות ומכניסה למבנה נתונים של hashMap. מבנה נתונים מורכב ממפתח שהוא מחרוזת המייצג את שם המסמך וערך שמיוצג ע"י מחלקת DocDetails שהוצגה קודם לכן. קריאת המסמך נעשית על כמות מסמכים שנגדיר מראש. הסיבה לכך היא שנרצה לחלק את הקורפוס לקטעים (קטע = כמות מסמכים מסויימת) כאשר כל קטע נקרא, נפרסר ונאנדקס. שיטות המחלקה:
   a. בנאי המקבל נתיב שאפשר לקרוא ממנו את הקבצים ומאתחל שדה מסוג file המייצג את התיקיה של הקורפוס.
   b. שיטת getSizeOfFolder - מחזיר את כמות המסמכים בקורפוס.

c. שיטת ReadFile - השיטה מקבלת מספר התחלתי ומספר סופי. תיקיית הקורפוס נכנסת למערך מסוג File והמספרים מייצגים את מספרי התיקיות עליהן נרוץ במערך. כל קובץ טקסט שנמצא בתיקייה נשלח לשיטת ReadOneFile.

d. שיטת ReadOneFile - מקבלת כקלט נתיב לקובץ טקסט ו-hashMap אליו נכניס את המידע שנקרא. השיטה עוברת על קובץ הטקסט ומפרקת אותו לפי תגיות של שם המסמך, כותרת, תאריך והטקסט שהמסמך מכיל. כל אלו נכנסים ל-hashMap שקיבלנו כקלט תחת האובייקט docDetails, כאשר המפתח הוא שם המסמך.

5. **מחלקת Parse** - תפקיד המחלקה היא לבנות מילון גלובלי המכיל את כל המילים ופרטים אודות המילים (אובייקט של termData). המחלקה מכילה את השיטות הבאות:

a. שיטת buildDictionary - השיטה מקבלת את ה-hashMap עם הפרטים של המסמכים ממחלקת ReadFile, מבנה נתונים לוקלי שבו נשמור את המידע שיכתב לדיסק הקשיח וערך בוליאני עבור ביצוע steamingת לאחר מכן השיטה עוברת על כל המילים של המסמך. כל מילה במסמך מסווגת לפי החוקים שניתנו לנו בעבודה. בונה את המילון הגלובלי שנשמר ב-ram. השיטה עוברת על כל מסמך שהתקבל כתוצר מה-ReadFile, מפצלת אותו לפי סימני פיסוק שונים ומסווגת את כל המילים בו לפי חוקים מסויימים (כל חוק מטופל בשיטות השונות המוצגות מטה). כמו כן, בסוף כל מסמך נשמר ב-hashMap ייעודי עבור מילון מסמכים, פרטים אודות המסמך (מספר מילים ייחודיות, מספר פעמים שמופיעה המילה הכי נפוצה). hashMap זה אינו מאד גדול ולכן נשאיר אותו ב-ram מבלי לכתוב ל-posting. רצף מילים שגילינו כחשוד לישות נעביר למבנה נתונים מיוחד גלובלי, שם הם ישארו עד שנמצא ביטוי נוסף שיאמת לנו את העובדה שמדובר בישות.
לאחר מעבר על כל המילים במסמך מסויים נעתיק את המידע למבנה נתונים לוקלי, יחד עם מספר הופעות של המילה ובאיזה מסמכים המילה הופיעה.
***כל שאר השיטות במחלקה תומכות בשיטה זו.**

b. שיטת initializeMonthList - כחלק מהחוקים של הפרסר עלינו לזהות אם ביטוי מסויים מייצג תאריך. כדי לבדוק זאת נבדוק האם אחת המילים מייצגת חודש, לשם כך אנו מאתחלות מבנה נתונים מסוג hash map שהמפתח שלו הוא מחרוזת המייצגת את שם החודש ב-3 וריאציות: אותיות גדולות, רק אות ראשונה גדולה, אותיות גדולות מקוצרות). שיטה זו מאתחלת את מבנה הנתונים עם החודשים.

c. שיטת insertStopWords - בעבודה ניתנה לנו רשימה של stop words מהן נוכל להתעלם ולא להכניסן למילון. השיטה עוברת על קובץ הטקסט של stop words ומכניסה את המילים למבנה נתונים של hash set.

d. שיטת addMonthNumbesFirst - שיטה זו מקבלת זוג מילים בהן זוהה שהמילה הראשונה מכילה מספר והמילה השניה מכילה חודש. השיטה בודקת האם המספר מייצגת תאריך או שנה ומחליפה את 2 המילים לפורמט שניתן בעבודה. בסיום ההחלפה הביטוי מוכנס למילון.

e. שיטת addMonth_MonthFirst - דומה לשיטה הקודמת רק שכאן זוהה כי המילה הראשונה מכילה חודש והמילה השניה מכילה מספר. פרט לכך, הכל כמו בשיטה הקודמת.
f. שיטת checkIfFreiction - מקבלת מילה ובודקת האם היא מייצגת שבר ממספרי. מחזירה ביטוי בוליאני.
g. שיטת addNumberToDictionary - השיטה מקבלת מילה שזוהתה כמספר ומתקנת אותה לפי החוקים שניתנו בעבודה. מסווגת גודל מספר (קטן מאלף, אלפים, מיליונים ומיליארדים) וכמו כן האם מדובר במספר עם אחוז.
h. שיטת wordWithoutPsik - שיטת עזר לשיטה addNumberToDictionary. מקבלת מילה שהיא מספר והופכת אותה למספר ללא פסיקים.
i. שיטת wordWithoutPsikANDDot - שיטת עזר לשיטה addNumberToDictionary. מקבלת מילה שהיא מספר והופכת אותה למספר ללא נקודות ופסיקים.
j. שיטת addPrice - מקבלת 4 מילים החשודות ככאלה המייצגות מחיר, מסווגת אותן לפי גודלן (גדול מאלף או קטן מאלף) והופכת לביטוי המתאים לפי הוראות העבודה.
k. שיטת addUpperLowerLetter - השיטה מקבלת מילה שהיא לא מספר ומנסה להבין האם המילה קיימת בצורה אחרת במילון, לדוגמה קיבלנו מילה עם אותיות קטנות וקיימת במילון אותה מילה עם אותיות גדולות. המילה מוכנסת למילון לפי החוקים הנתונים בעבודה.
l. שיטת checkDotPsikHyphemInTheEnd - השיטה מקבלת את המילה בשלבים הראשונים של הפרסור ובודקת האם בסוף המילה יש פסיק, נקודה או מקף. אם כן, המילה מורידה את התו הלא רלוונטי ומחזירה מילה מתוקנת.
m. שיטת checkTermWithBETWEENandAdd - אחד החוקים בעבודה מבקשת לתמוך בביטויים המסמנים טווח כלשהו בהם מופיעה המילה "between". השיטה מקבלת רצף של 4 מילים ובודקת האם מדובר במקרה הנ"ל. אם כן, היא מכניסה את הביטוי למילון בצורה שבה התבקשנו בעבודה.
n. שיטת howMuchSuspected - שיטת עזר למציאת ישויות. השיטה מקבלת רצף מילים ומחזירה מספר המייצג כמה מהמילים הן עם אות גדולה (ברצף). לדוגמה, אם השיטה תחזיר 3, נקבל ש-3 המילים חשודות כישות ונטפל בהן בנפרד.
o. שיטת addToDictionary - השיטה הזו נקראת כמעט מכל הפונקציות הרשומות מעלה ותפקידה להכניס את המילה למילון השמור ב-ram ושינוי ה-termData(מס' מסמכים ומס' הופעות) עבור המילה הזאת במילון.
p. שיטת YeshutAddToDictinary - שיטה זו אחראית על הכנסה של ישות למילון הגלובלי אחרי שמצאנו כי המילה החשודה היא אכן ישות. עשינו שיטה נפרדת להכנסה למילון של ישויות כיוון שהטיפול בהן הוא שונה מעט ובכל הכנסה אנחנו מכניסים בעצם 2 ביטויים שאנחנו מאחדים.
q. שיטת getYeshutGlobalMap - כחלק מהחוקים למציאת ישות עלינו לבדוק האם ביטוי שמצאנו כחשוד לישות מופיע במסמכים אחרים. כיוון שתהליך ה-parsing נעשה עבור כמות

מוגבלת של מסמכים שאחר כך נכתבים ל-posting, פתחנו hashMap המכיל את כל הביטויים שמצאנו כחשודים אך טרם נמצא ביטוי זהה במסמך אחר ולכן לא יכולנו לסווג את הביטוי כישות. השיטה הזו מחזירה את מבנה הנתונים ששומר את הביטויים החשודים.

r. שיטות clearDictionary\clearDocInfo - נועדו למחוק את מבני הנתונים (hash map) כאשר המשתמש לוחץ על הכפתור המיועד לכך במסך הGUI.

s. שיטת getDocInfo - מחזירה את מבנה הנתונים ששומר פרטים אודות המסמכים.

t. שיטת getDocInfoSize - מחזירה את גודל מבנה הנתונים ששומר מידע על המסמכים, כלומר כמות המסמכים שעברנו עליהם בקורפוס. אנחנו משתמשים בו עבור הודעה המופיעה בסוף בניית המילון המעדכנת בין היתר על כמה מסמכים עברנו.

u. שיטת getDictionary - מחזירה את המילון הגלובלי שנבנה מהפרסר.

6. מחלקת Indexer - מחלקה זו מאחדת את פעילות ה-ReadFile וה-Parse. המחלקה קוראת בלולאה כל פעם 8 תיקיות שבהן נמצאים הקבצים, לאחר מכן מפרסרת אותם ומכניסה למבנה נתונים לוקלי את הנתונים אודות המילים אותם פרסרנו ב8 התיקיות הנ"ל. בסוף הפרסור היא כותבת את המילים והמידע עליהם לדיסק הקשיח(ממבנה הנתונים הלוקלי) ולבסוף מאחדת את כל ה-posting files שהתקבלו.

השיטות במחלקה:

a. בנאי המקבל נתיב לקריאה, נתיב לכתיבה וערך בוליאני האומר האם המשתמש ביקש להשתמש ב-stemming או לא.

b. שיטת index - השיטה המרכזית שמפעילה הכל. רצה כל פעם על 8 תיקיות שבהם נמצאים הקבצים והמסמכים, קוראת אותם, מפרסרת ומאנדקסת. בסוף הריצה על כל הקורפוס השיטה דואגת גם לטפל בכל המילים שהיו חשודות כישויות אך לא נמצא להן ביטוי תואם במסמך אחר ולכן הן מופרדות ומוכנסות ל-posting. ולמילון הגלובלית כמו כן השיטה קוראת לשיטה אחרת הממזגת את כל קבצי הפוסטינג שנוצרו לקובץ פוסטינג אחד מאוחד.

c. שיטת loadDictionary - השיטה עוברת על קובץ ה- posting המאוחד ומטעינה את המידע למילון. כלומר מטעינה עבור כל מילה במילון את מספר המסמכים שהמילה הופיעה בהם, את מספר המופעים של המילה באופן כללי ופוינטר לשורה בקובץ הפוסטינג ששם נשמר המידע עבור המילה הזאת, כמו כן היא מפרידה בין מקרים בהם המשתמש מבקש לבצע stemming וכאלה שלא. משתמש שיבקש לטעון למילון קובץ עם stemming אך לא בנה מילון עם stemming לפניי יקבל הודעת שגיאה.

d. שיטת posting - השיטה מקבלת ניתוב לכתוב בו את קובץ הפוסטינג ומבנה נתונים של treeMap שבו שמרנו את כל המידע שנרצה לשמור ב-posting עבור איטרציה של 8 תיקיות. השיטה פותחת כל פעם קובץ בשם IC + counter ואליו היא מכניסה את הנתונים הבאים, מופרדים באמצעות התו '@': שם מילה, כמות מופעים סך הכל, רשימת מסמכים בהם היא הופיעה+כמה פעמים בכל מסמך, מספר מסמכים בהם הופיעה.

e. שיטת MergePostingFiles - השיטה מקבלת ניתוב לכתיבה ומכניסה את כל קבצי ה-posting לתוך תור. עבור כל 2 מסמכים, היא יוצרת מסמך חדש, אליו היא כותבת את האיחוד של 2 קבצי ה-posting. בסוף האיחוד, השיטה מוחקת את ה-posting files שהשתמשה בהם, ואת הקובץ החדש המאוחד מכניסה לתור. האיחוד האחרון יהווה את קובץ ה-posting הסופי המאוחד. השיטה מבדילה בין בניית posting files עם stemming וללא. קבצים ללא stemming יתחילו באות p ואליהם יצורף counter רץ. קבצים עם stemming יתחילו באות s ואליהם יצורף counter רץ.
f. שיטת getP - מחזירה את אובייקט הפרסר שפתחנו בשיטת ה-index.
g. שיטת getPostingFileName_NoStem - כיוון שעלינו לתמוך בהעלאת מילון מתוך קובץ posting קיים, אין לנו דרך לשמור את שם הקובץ בשום מקום. השיטה מקבלת את הנתיב לכתיבה, עוברת על כל הקבצים בתיקייה, ומחזירה שם קובץ שמתחיל ב-p ומסתיים ב-".txt" (קבצי posting ללא stemming מקבלים אצלנו את השם p + מספר האיחוד האחרון). השיטה מחזירה את שם הקובץ שמצאה.
h. שיטת getPostingFileName_WithStem - בדומה לשיטה הקודמת רק שכאן השיטה מחפשת שם קובץ שמתחיל ב-s ומסתיים ב-".txt" (קבצי posting ללא stemming מקבלים אצלנו את השם p + מספר האיחוד האחרון). השיטה מחזירה את שם הקובץ שמצאה.

7. **מחלקת myModel**

מחלקה מקשרת בין שכבת ה-viewModel ללוגיקה. היא זו בפועל מבצעת את הפעולות שמבקש המשתמש. שיטות המחלקה:

a. שיטת setDictionary - מתפעלת את בניית המילון. השיטה פותחת אובייקט מסוג Indexer ומפעילה את שיטת ה-index. כמו כן היא סופרת את זמן הריצה ומדפיסה הודעה בסוף בניית המילון ובה פרטים על זמני ריצה, כמות מסמכים שעברנו עליהם וכמות מילים ייחודיות.
b. שיטת loadDictionary - קוראת לשיטת loadDictionary מה-Indexer.
c. שיטת clearData - מוחקת את כל הקבצים שפתחנו כחלק מתהליך בניית המילון בדיסק בקשיח וכמו כן קוראת לשיטות במחלקת Indexer שאחראיות על מחיקת מבני הנתונים של ה-ram.
d. שיטת showDictionary - לוקחת את פרטי המילון הקיימים ב-ram ומציגה אותם בטבלה באמצעות מחלקת table view שיבאנו.
e. שיטות set-רים:
   i. שיטת setToStem - מעדכנת שדה בוליאני שאומר האם לבצע stemming.
   ii. שיטות setPathToRead\setPathToWrite - מעדכנת את הנתיבים שקיבלה משכבת viewModel.

סעיף b c d

האופן שבו התמודדנו עם מגבלת זיכרון המחשב היא באמצעות קבצי posting, למעשה כאשר נכנסו לפונקציית build dictionary במחלקת parse וביצענו את הפרסור (עבור המילים של 8 התיקיות שאותם קיבלנו כקלט לפונקצייה), הכנסנו את המילים המפורסרות ואת פרטיהם(מס' מופעים של המילה במסמך ואת המסמכים שבהם הופיעה המילה) למבנה נתונים לוקלי(tree map אשר ממומש עם comperator מיון לפי ה- lower case של אותה מילה), מבנה הנתונים שהשתמשנו בו לצורך החזקת הנתונים הזמנית של 8 התיקיות הוא treemap שזמן הכנסתו למילון הוא log n אשר המילים בו יהיו ממויינות וזה יעזור לנו באיחוד של קבצי הפוסטינג לקובץ אחד.

יש לציין כי בחרנו לעבור על 8 תיקיות בכל איטרציה של פרסור על מנת ליצור מבנה נתונים לוקלי שלא יכיל המון מילים וכך זמן ההכנסה למבנה הנתונים יהיה קטן יותר מאשר מעבר על מספר גדול יותר של תיקיות, כמו כן מבנה נתונים זה לא יתפוס המון מקום ב ram ולא יכביד עליו, וכמובן שנרצה מעט מיזוגים של קבצי פוסטינג ולכן לדעתנו מעבר על 8 תיקיות בכל פעם נותן את האיזון הטוב ביותר, בנוסף לכך ניתן לראות כי קיים איזון בין שמירת נתונים גדולה על הram לעומת כתיבת הנתונים לדיסק שזוהי פעולה יקרה.

לאחר סיום מעבר על 8 התיקיות חזרנו למחלקת indexer שם נכנסו לפונקציית posting אשר כותבת לקובץ טקסט את הנתונים ממבנה הנתונים הלוקלי, שם הקובץ יהיה IC עם counter רץ עם סיומת .txt כלומר סוג קובץ ה-posting הוא קובץ טקסט, בקובץ זה שמרנו מחרוזת אחת ארוכה עבור כל מילה שמופרדת באמצעות התו @, המחרוזת תיראה בצורה הבאה: **מספר מסמכים שבהם הופיעה המילה @ שם מסמך וכמות המופעים שהמילה הופיעה במסמך כך עבור כל המסמכים שבהם הופיעה המילה..@ מספר מופעים כולל של המילה בכל המסמכים@ המילה**

בחרנו לשמור נתונים אלו מכיוון, שכאשר ניגש למילה במילון ונרצה לדעת באיזה מסמכים היא הופיעה וכמה פעמים הופיעה בכל מסמך נוכל לעשות זאת בקלות באמצעות פוינטר למספר שורה בפוסטינג ללא צורך חיפוש בכל ה-corpus כולו.

נשים לב שעבור ה-corpus הניתן לנו, נוצרו 227 קבצי posting file

כך למעשה לא שמרנו את כל הנתונים על המילים שבdataset על ה-ram אלא בדיסק הקשיח, ומה שנשאר על ה-ram הוא המילון עם מעט המידע (termData) שיש עבור כל מילה, כמו כן לאחר סיום המעברים על כל התיקיות ב-corpus וכתיבתן לקובץ posting נכנסו לפונקציית merge במחלקת Indexer, שם נעזרנו בתור עדיפויות כדי לאחר את כל קבצי הפוסטינג לקובץ מאוחד אחד. זה נעשה בדרך הבאה: הכנסת כל קבצי הפוסטינג שנמצאים בתיקייה של הנתיב לכתיבה, לאחר מכן יצירת קובץ פוסטינג חדש בשם p או s (עבור פוסטינג עם steaming), הוצאת שני קבצים מהתור ומעבר על עליהם בצורה ממויינת, לאחר סיום המעבר על שני הקבצים וכתיבת האיחוד ל-posting file חדש, נמחק את הקבצים שהשתמשנו בהם ונכניס את הקובץ החדש לתור, וכך נעשה עד שנישאר עם קובץ אחד מאוחד.

תיעוד קבצי הפוסטינג שנוצרו בעת כתיבתן לדיסק הקשיח :

Name	Date modified	Type	Size
corpus	18/12/2019 21:22	File folder	
IC0	18/12/2019 21:24	Text Document	3,975 KB
IC1	18/12/2019 21:24	Text Document	4,598 KB
IC2	18/12/2019 21:24	Text Document	4,452 KB
IC3	18/12/2019 21:24	Text Document	4,516 KB
IC4	18/12/2019 21:25	Text Document	4,854 KB
IC5	18/12/2019 21:25	Text Document	4,954 KB
IC6	18/12/2019 21:25	Text Document	4,928 KB
IC7	18/12/2019 21:25	Text Document	5,096 KB
IC8	18/12/2019 21:25	Text Document	4,554 KB
IC9	18/12/2019 21:25	Text Document	5,059 KB
IC10	18/12/2019 21:25	Text Document	4,894 KB
IC11	18/12/2019 21:25	Text Document	4,298 KB
IC12	18/12/2019 21:25	Text Document	4,705 KB
IC13	18/12/2019 21:25	Text Document	4,650 KB
IC14	18/12/2019 21:25	Text Document	4,779 KB
IC15	18/12/2019 21:25	Text Document	4,902 KB
IC16	18/12/2019 21:25	Text Document	4,910 KB
IC17	18/12/2019 21:25	Text Document	4,372 KB
IC18	18/12/2019 21:25	Text Document	4,564 KB

סעיף e

פרטי האינפורמציה הנוספים ששמרנו הם עבור מסמך: אורך מסמך על מנת שיעזור לנו בהמשך הפרוייקט בחיפוש שאילתות על הקורפוס בנרמול ערך הtf, פרט אינפורמצייה נוסף שהוספנו הוא מספר הישויות במסמך בו נידרש להיעזר בחלק ב, בו נצטרך לחפש מסמכים שבהם ערך הישויות גדול ממספר מסויים

סעיף f

שני החוקים הנוספים שהוספנו ב-parser הם:

1. מילים שמתחילים במספר ולאחר מכן מופיעים אותיות ייכנסו ב-uppercase

כמו למשל : 0.85kW או 1.999k יהפכו להיות : 1.999K 0.85KW

2. מילים שמתחילים במספר וכוללים מקף ייכנסו גם כ- uppercase

כמו למשל: 70-percent יהפוך להיות: 70-PERCENT

תיעוד מתוך שני מסמכים שונים ב-dataset עבור החוק הראשון

מתוך מסמך שמספרו : FT932-4790

נתון חלק מהטקסט שבו מופיע המילה ששייכת לחוק הראשון

```
end of 1992 it was 11,000MW.
Over the next 17 years the province hopes to increase capacity by a further
69,000MW to 80,000MW. By then it may have achieved installed capacity of 1KW
hour per person, up from its current installed capacity of 0.16KW per
person. Hong Kong has installed capacity per person of 1.42KW, Taiwan
0.85KW, and the US about 2.8KW.
Guangdong's spending on power plants has grown from USDollars 679m (Pounds
441m) in 1989 to Dollars 795m in 1991. Within that total, the share of
imports has risen from Dollars 220m in 1989 to Dollars 430m in 1991. Larger
expenditures are expected in the coming years, making the market for
electrical generation equipment, transmission lines and computer systems to
```

מתוך מסמך שמספרו: FT924-9936

נתון חלק מהטקסט שבו מופיע המילה ששייכת לחוק הראשון

```
For delicate fabrics, Epri has been blowing cooling air into the microwave
drier, where temperatures have not generally exceeded 1100F - appropriate
for delicate fabrics. For normal loads, the shortest drying time could be
achieved by combining heated air with microwaves.
Since the middle of last year, Epri has been testing an experimental unit
built by two Californian companies, Thermo Energy and JG Microwave. Equipped
with eight 0.85kW magnetrons, the unit can supply 6.8kW of microwave power,
enough to dry a seven-pound load in about half the time required by a
conventional electric drier.
Overall the savings on time and energy are impressive, but are complicated
by the probability that any production machine would be a hybrid of
microwave and conventional drying. The reduction in drying times could range
from 25 to 60 per cent, says Kesselring, with the biggest gains in
```

תיעוד מתוך שני מסמכים שונים ב-dataset עבור החוק השני

מתוך מסמך שמספרו :FBIS3-29

נתון חלק מהטקסט שבו מופיע המילה ששייכת לחוק השני

SKODA MLADA BOLESLAV SALES RISE 20 PERCENT: The Skoda automobile plant sold 219,600 cars in the 1993 fiscal year. This information was provided by Detlev Schmidt, chief of the marketing department. Compared to the 1992 results, this is a 20-percent increase. (Prague MLADA FRONTA DNES 10 Jan 94 AU)

DECEMBER 1992-DECEMBER 1993 INFLATION 18.2 PERCENT: Consumer prices rose 18.2 percent in the December 1992-December 1993 period. This information was provided by the Czech Statistical Office. (Prague HOSPODARSKE NOVINY 11 Jan 94 p 1 AU)

FOREIGN TRADE BANK ACQUIRES 70-PERCENT STAKE IN CEDOK: The Ceskoslovenska Obchodni Banka (Czechoslovak Foreign Trade Bank -- CSOB) has become the owner of 70 percent of the Cedok travel agency. This information was confirmed by Milan Tomanek, press spokesman for the bank. (Prague LIDOVE NOVINY 11 Jan 94 p 13 AU)

מתוך מסמך שמספרו :FBIS3-60

נתון חלק מהטקסט שבו מופיע המילה ששייכת לחוק השני

The Ericsson electronics company accounts for the lion's share--60 percent--of Sweden's exports to China, which is well on its way to becoming Ericsson's largest market, according to an article in DAGENS NYHETER on 29 January. Ericsson has been most successful in exporting mobile phones to China, where it holds a 70-percent market share. The article cites Hans Ekstrom, the company's head of operations in China, as saying that "the sale of mobile telephones in China has been a real cash cow for us." Ericsson's sales of these phones--a popular status symbol in China--doubled in 1992 and 1993 and are expected to further increase when China installs a standardized digital system. Ericsson already has an advantage in the vast Chinese market because of its ability to build large-scale

סעיף h

השתמשנו בעבודה בקוד פתוח

במחלקת ReadFile השתמשנו בקוד פתוח מהכתובת הבאה:

https://jsoup.org/download

נעזרנו בקוד זה על מנת לפרק את המסמכים שקיבלנו בקורפוס על פי תגיות כמו : text, docNom וכו'

במחלקת Parser השתמשנו בקוד הפתוח מהכתובת הבאה:

https://jar-download.com/artifact-search/porter-stemmer

נעזרנו בקוד זה על מנת להפוך מילה לשורש על פי האלגוריתם של פורטר

סעיף 2

סעיף a+b+c

גודל המילון-

ללא steaming הוא 1,574,980 מילים

עם steaming הוא 1,565,362 מילים

מספר ה- terms שהם מספרים במאגר הוא: 256,223

סעיף d

עשרת המילים השכיחות ביותר במאגר לפי סדר השכיחות

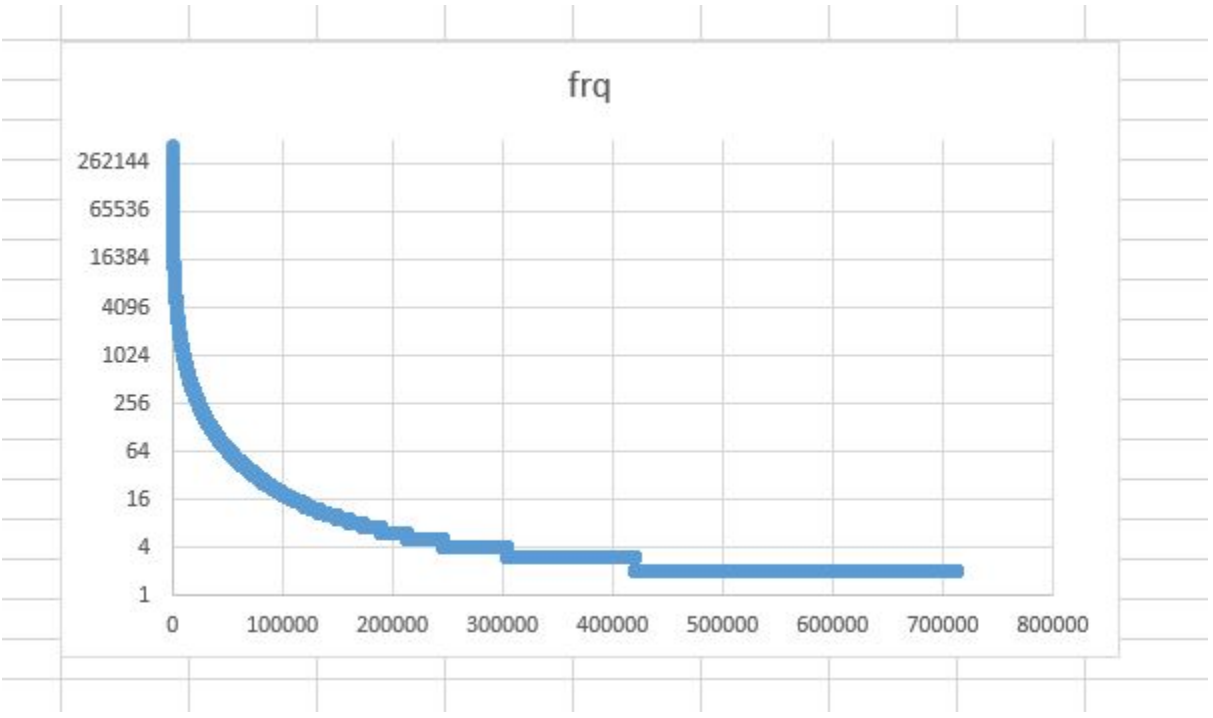
429235	year
363316	cent
315034	government
302151	pounds
286160	people
247661	years
240341	market
237889	time
208298	company
176971	state

עשרת המילים הכי פחות שכיחות ביותר במאגר לפי סדר השכיחות

7	1	OCHIROVA
8	1	3,42.75
9	1	98.69%
0	1	TSCHOELTSCH
1	1	3,042.8M
2	1	152
3	1	3,42.9
4	1	176,120
5	1	DEISHA
6	1	OCHRONY

סעיף e

גרף Zipf's Law



הגרף אכן תואם לגרף zipf's law

סעיף f

1.988 K@1
1.992 K@1
1.993 K@1
10@1
100.0 K
120%@1
15@1
151@1
400.0 K Dollars@2
44@1
60.21 M@1
65 M@1
72 M@1
804 M@1
added@1
addition@1
airport@1
approved@1
areas@1
ARTICLE@1
AUTONOMOUS@1
BFN@1
border@1
BORDERING@1
build@1
built@2
BUIR@1
centers@1
CHINA@2
cities@3
connections@1
contracts@1

cooperation@1
cooperations@1
dollars@1
economic@2
emerged@1
ENGLISH@1
expanded@1
experimental@1
foreign@2
francs@2
funds@1
handled@1
hefty@1
highway@1
HOHHOT@1
HONG@1
HULUN@1
implemented@1
IN@1
including@1
involve@1
JAPAN@1
joint@1
KONG@1
KOREA@1
LANGUAGE@1
LAST@1
leading@1
league@3
MACAO@1
MARCH@1
modern@1

MONGOLIA@2
NEW@1
north@1
number@1
opened@1
OVERSEAS@1
people@1
port@1
railway@1
reform@1
region@3
REPUBLIC@1
rise@1@
routes@1
RUSSIA@2
SINGAPORE@1
STATES@1
structural@1
SWISS@2
SWITZERLAND@1
TAIWAN@1
technological@1
TEXT@1
topped@1
total@1
totalling@1
tourist@1
tourists@1
trade@1
trading@1
turnover@1
TYPE@1

U.S@1
UNITED@1
ventures@1
volume@1
world@1
XINHUA@1
year@1
yuan@2
ZEALAND@1
zone@1

<u>סעיף g</u>

גודל קבצי הposting ללא steaming :

KB 928,614

עם steaming :

957,958 KB

<u>סעיף h</u>

משך הזמן שלקח למנוע החיפוש לבנות את האינדקס על קבצי הקורפוס :